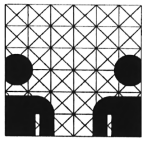
METHODEN DER VORFALLSERKENNUNG UND -ANALYSE

unter besonderer Berücksichtigung komparativ statischer Systemanalysen am Beispiel von Festplattenzuständen

Diplomarbeit

Fachbereich Informatik

Universität Hamburg



Jan Menne

Betreuer:

Prof. Dr. Klaus Brunnstein

Dr. Hans-Joachim Mück

Hamburg, im September 2003

ZUSAMMENFASSUNG

Seit 1988 stellt das CERT/CC jährlich fest, dass sich die Zahl der Sicherheitsvorfälle wieder erhöht und in den letzten Jahren sogar verdoppelt hat. Gleichzeitig wurden immer mehr Schwachstellen bekannt. Beides zusammen macht es erforderlich, dass geeignete Methoden zur Vorfallserkennung und -analyse zur Verfügung stehen und genutzt werden.

In dieser Diplomarbeit wird gezeigt, dass Vorfälle als Anforderungsverletzungen von Vertraulichkeit, Integrität und Verfügbarkeit auf einer IT-System-Schicht eintreten, mit eventuellen Auswirkungen auf die anderen Schichten und Sicherheitsanforderungen. Entsprechend diesem Verständnis und unter Berücksichtigung der verschiedenen Vorfallsprozesssichten lässt sich eine Klassifikation für alle Methoden der Vorfallserkennung und -analyse aufstellen. Ein wichtiger Aspekt in der Klassifikation ist die Unterscheidung, wie die Analysedaten erfasst werden: dynamisch durch Beobachtung der Transitionen während des Vorfalls oder komparativ statisch durch den Vergleich von Systemzuständen. Aus den möglichen Methoden wird der Weg zur Implementation einer komparativ statischen Methode dargestellt, die die Abbildung des IT-System-Zustandes auf der Festplatte fokussiert. Mit der entwickelten Software CompareSys können auf der Treiberschicht Integritätsverletzungen der Festplattendaten festgestellt werden. Die Ergebnisse ermöglichen es einem Anwender, Sicherheitsverletzungen sogar auf anderen Systemschichten zu identifizieren, wie die Versuchsreihe zeigt. Trotz und wegen der Erkenntnisse werden Grenzen der komparativ statischen Systemanalyse am Beispiel von Festplattenzuständen aufgezeigt werden.

ABSTRACT

Since 1988, the annual number of computer security incidents reported to the CERT/CC has increased and in the last years even doubled. At the same time many more vulnerabilities have been published. Both – the number of incidents and the number of vulnerabilities – raise the need for suitable methods for incident detection and analysis.

In this paper, incidents are explained as the violation of security requirements such as confidentiality, integrity and availability. Those violations originate on one layer of the computer architecture – possibly affecting other layers and security requirements. This understanding of incidents under consideration of different views of incident processes leads to a classification of methods for incident detection and analysis. An important aspect for the classification is the difference in how the analysis data is gathered: dynamically by monitoring the transitions during the incident or comparative static by comparing system states. The implementation of a comparative static method is described. It focuses on the representation of the system state on the hard drives. The software CompareSys is capable of detecting integrity violations on hard drive data. Moreover, the results enable a user to identify further security violations even on other layers of an IT system as shown in a series of experiments. In accordance with the results the limits of this comparative static system analysis considering the example of hard disk states are shown and respective conclusions are drawn.

VORWORT

Im Sommersemester 2001 begann ich, mich auf den Schwerpunkt IT-Sicherheit zu konzentrieren. Neben dem Besuch entsprechender Vorlesungen entstand so ebenfalls die im Juni 2001 verfasste Baccalaureats-Arbeit „Intrusion Detection Systeme in Firewalls“, die ich in Zusammenarbeit mit Benjamin Hoherz, Silvio Krüger und Nils Michaelsen erstellt habe. Aufgrund dieser Projektarbeit stellte sich mir erstmals die Frage, wie das Wissen erlangt wird, nach dem Firewalls und Intrusion Detection Systeme konfiguriert werden. Zwar zeigte das Buch „Kuckucksei“ von Clifford Stoll (s. [Stoll 1999]), wie ein Sicherheitsvorfall erkannt und analysiert wurde; die beschriebene Vorgehensweise schien in meinen Augen aber stark vom Zufall und den Ereignissen gesteuert zu sein.

Nach meinem Einstieg in die IT-Sicherheit absolvierte ich zwei Semester im antiViren Test Center (aVTC) der Universität Hamburg. Dort erfuhr ich, wie reaktiv und proaktiv gegen Bedrohungen der Klasse Malware vorgegangen werden kann, und lernte signatur- und heuristikenbasierte Ansätze kennen.

Hier stellte sich mir eine weitere Frage, die mich an mein Diplomarbeitsthema heranführte: wie wird das erste Auftreten einer Malware-Instanz analysiert, um die Software als Malware zu erkennen und eine Signatur erzeugen zu können? Die Antwort fand ich, als am Fachbereich Informatik im Wintersemester 2002 / 2003 ein Incident Response Team (IRT) als Studenten-Projekt gegründet wurde. Auch die anderen Projektteilnehmer suchten, wie ich, nach Lösungsansätzen auf diese Fragen – Fragen, die bis dato nur in wenigen wissenschaftlichen Quellen detailliert behandelt werden. Selbst die Computer Emergency Response Teams (CERTs) beschränken sich größtenteils auf die Weiterleitung von Hinweisen, wie folgende Aussage des CERT-Bund des Bundesamtes für Sicherheit in der Informationstechnik belegt:

> „Aufgrund der hohen Anzahl kann aber leider nicht jeder Vorfall detailliert analysiert werden. Deshalb beschränken sich viele Advisories darauf, den Hersteller-Angaben zu vertrauen und diese bewertet weiter zu geben.“ (vgl. [CERT-Bund 2003])

Laut dieser Aussage führen also primär die Hersteller derartige Analysen durch. Diese Erkenntnis half uns nicht weiter, denn im Gegensatz zu den Herstellern und Malware-Autoren hat das IRT meistens keinen Zugriff auf den Quellcode. Damit entfällt auch die Möglichkeit, den Quellcode per Debugging Schritt für Schritt durchzugehen. Aus diesem Grund beschloss das IRT, sich zunächst den schon bekannten Würmern Nimda und Slapper zu widmen. Unser Ziel war, zu sehen, ob die eigenen Ergebnisse mit den detaillierten Beschreibungen der AntiMalware-Hersteller übereinstimmen. Dies sollte uns als Grundlage zur Bewertung unserer eigenen Ansätze dienen.

Die Erkenntnisse der ersten Untersuchungen und das Ziel, eine umfassende Vorfallsanalyse durchführen zu können, bilden meine Motivation für diese Diplomarbeit. Die Diplomarbeit wurde durch Beiträge in vielen Gesprächen, Diskussionen und Projekten bereichert.

Danken möchte ich deshalb besonders meinen Betreuern:

Professor Doktor Klaus Brunnstein und

Doktor Hans-Joachim Mück

sowie (in alphabetischer Reihenfolge):

Astrid Beger, Christoph Buchwald, Karin Colsman, Benjamin Hoherz, Bastian Koos, Silvio Krüger, Mathias Meyer, Nils Michaelsen, Lars und Tanja Orta, Christoph Rößner, Birgit Schreiber, Kerstin Schwarze, Jan Seedorf, René Soller als auch den anderen ehemaligen sowie jetzigen Mitgliedern des Incident Response Teams der Universität Hamburg und – nicht zu vergessen – meinen Eltern für ihre Unterstützung.

INHALTSVERZEICHNIS

0 Einleitung

Mitte 2002 klingelte mein Telefon und ein Freund sagte: „Sag mal, Du kennst Dich doch aus, mein Computer verhält sich so komisch. Was könnte das sein?“ Nach einigen Fragen, Versuchen und Untersuchungen wussten wir, dass der Wurm W32/CodeRed@MM auf dem Computer aktiv war. Die offensichtliche Frage war danach: wie gelangte CodeRed auf den Computer?

Diese und ähnliche Vorfälle treten täglich auf verschiedensten IT-Systemen aller Welt auf und bei jedem stellt sich die Frage, wie das passieren konnte. Die Untersuchungen solcher Vorfälle werden von so genannten Computer Security Incident Response Teams (CSIRT) vorgenommen. Ein solches CSIRT ist das Computer Emergency Response Team Coordination Center (CERT/CC), an das Vorfälle und Schwachstellen gemeldet werden können. Schwachstellen werden hier zunächst an die Hersteller berichtet und erst später veröffentlicht, um den Herstellern Reaktionen auf die Schwachstellen zu ermöglichen.

Bei den Untersuchungen werden immer wieder neue Software-, aber auch Hardware-Schwachstellen festgestellt. Allein das CERT/CC veröffentlichte im Jahr 2002 4129 gemeldete Schwachstellen. Laut einer Statistik (vgl. [CERT/CC 2003]) ist die Anzahl der an das CERT/CC gemeldeten Schwachstellen seit 1999 jährlich um den Faktor 2 gewachsen (vgl. Abb. 1). Letztes Jahr wurden im Schnitt an jedem Tag 11 neue Schwachstellen gemeldet. Im selben Zeitraum verdoppelte sich ebenfalls jährlich die Anzahl der gemeldeten Vorfälle auf 82.094 im letzten Jahr (vgl. Abb. 1). Das sind 225 gemeldete Vorfälle pro Tag.

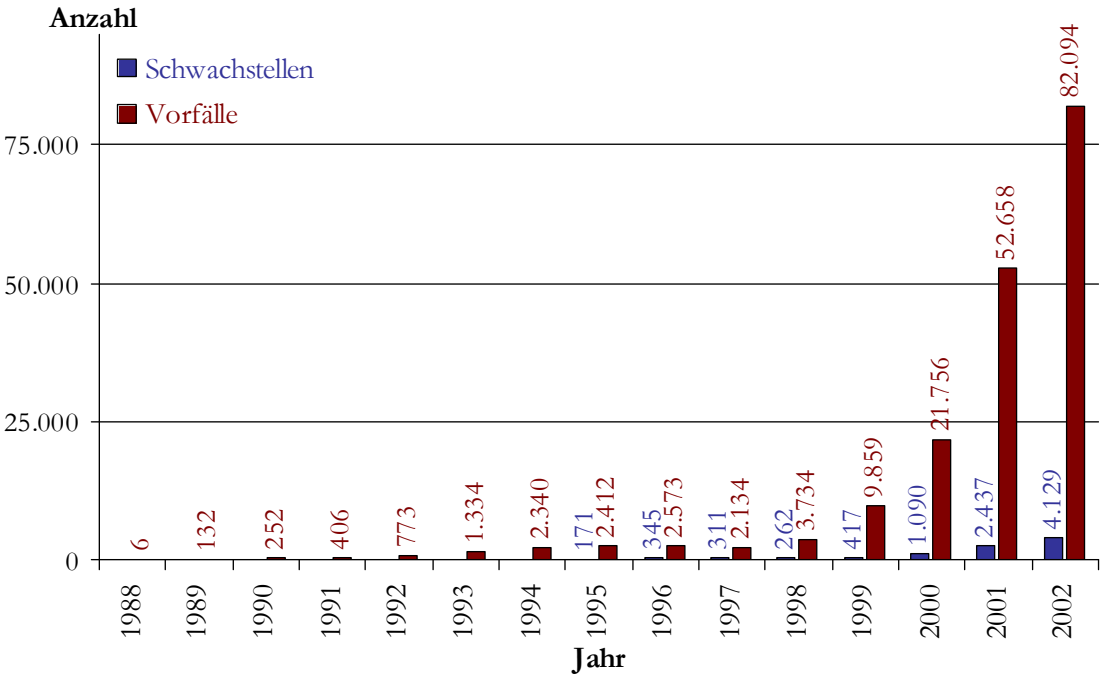


Abb. 1: Anzahl an das CERT/CC gemeldeter Schwachstellen und Vorfälle nach [CERT/CC 2003]

Anders als bei dem Freund, dessen Computer mit den richtigen Tools und wenig Aufwand wieder voll funktionsfähig gemacht werden konnte, entsteht durch solche Vorfälle besonders in der Wirtschaft nennenswerter materieller Schaden. Bei einer Umfrage des Computer Security Institutes (CSI) und der Computer Crime Squad des Federal Bureau of Investigation (FBI) in San

Francisco gaben US-Unternehmen für 2001 einen Schaden von 375 Millionen US-Dollar an (vgl. [CCSS 2001: S.11]). Allerdings konnten oder wollten nur 37 % der 538 befragten Firmen (also etwa 200 Unternehmen) ihren Schaden vorfallsspezifisch quantifizieren (vgl. Abb. 2).

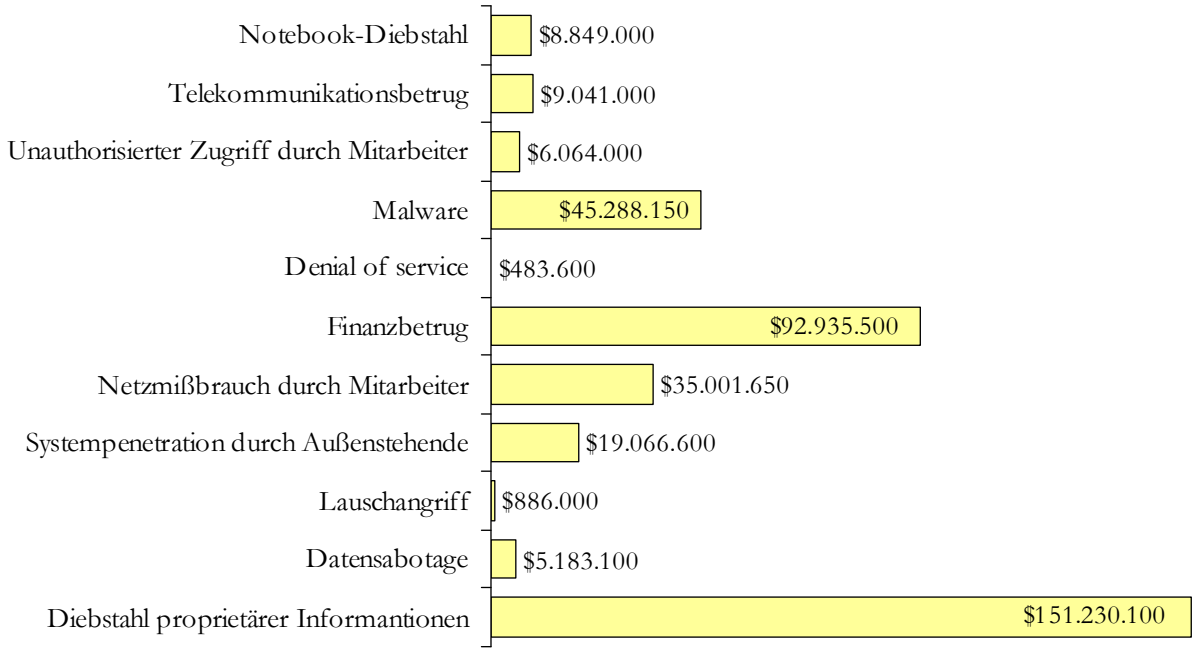


Abb. 2: Quantifizierbare Verluste im Jahr 2001 (USA) nach [CCSS 2001: S.14]

Trotz der Schadenshöhe haben nur 36 % aller Unternehmen die Staatsanwaltschaft eingeschaltet. Häufig sehen die betroffenen Unternehmen von Anzeigen ab, um Imageverlusten durch die Veröffentlichung solcher Vorfälle vorzubeugen. Die aus den nachfolgend dargestellten Analysen hervorgehende Häufigkeit der internen Angriffe, also von Mitarbeitern dieser Unternehmen, trägt meines Erachtens zum Verständnis für diese Handlungsweise bei:

Aus der Umfrage von CSI und FBI geht durch die Verteilung der Vorfälle (vgl. Abb. 3) hervor, dass die Unternehmen nicht nur Opfer externer Angriffe, sondern in hohem Maße auch Opfer interner Angriffe sind. Mit Blick auf die Vorjahre sind die Angriffe von Intern und solche auf die Telekommunikationstechniken konstant bzw. rückläufig, auch wenn etwa 50 % der Unternehmen für 2001 angeben, dass Mitarbeiter unerlaubten Zugriff hatten und sogar bei 91 % der Unternehmen die Mitarbeiter den Netzzugriff missbrauchten.

Überraschend ist, dass 94 % der Unternehmen und damit mehr als je zuvor in der Umfrage Opfer von Malware wurden, obwohl die Abwehrwerkzeuge wie zum Beispiel AntiMalware und Frühwarnsysteme mittlerweile bekannt und im Einsatz sind. Der durch Malware verursachte Schaden beläuft sich bei den rund 200 Unternehmen auf 45 Millionen US-Dollar und liegt damit deutlich unter den 175 Millionen US-Dollar Schaden durch Sabotage, Innovationsdiebstahl und Systempenetrationen.

Wegen des hoch zu beziffernden Schadens und aufgrund des weiteren Schadenspotentials ist es sehr wichtig, die Vorgänge zu verstehen, die zu einem Computersicherheitsvorfall führen. Deshalb wurde zur Analyse von Vorfällen an der Universität Hamburg das Incident Response Team (IRT) des Fachbereichs Informatik gegründet. Aufgabe dieses Teams ist es, Vorfälle zu

erkennen, eine Analyse der Vorfälle vorzunehmen sowie Maßnahmen und Konzepte zur Abwehr von Vorfällen gleichen Typs zu erarbeiten.

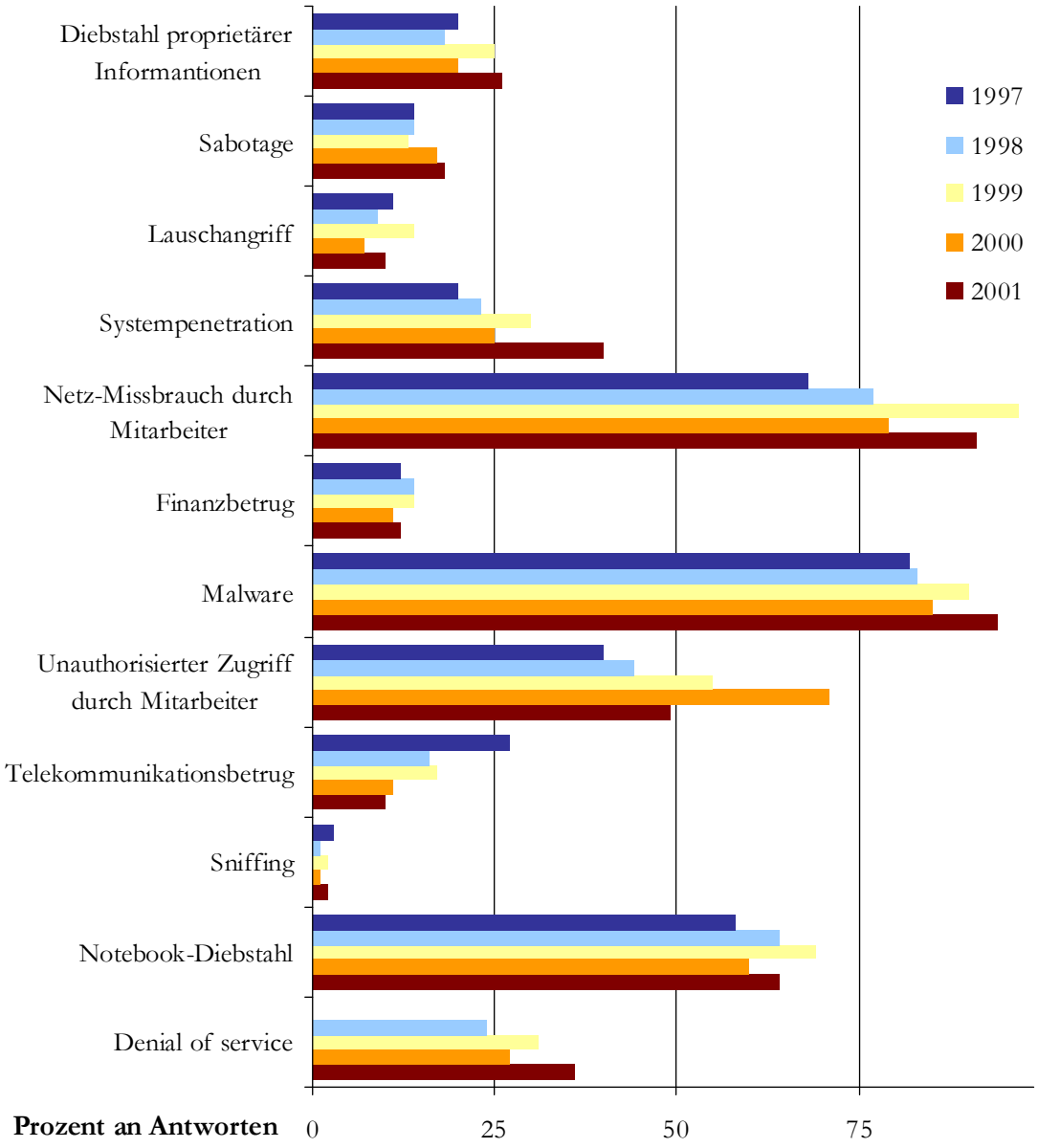


Abb. 3: Arten des Angriffs auf US-Datennetze (in %) nach [CCSS 2001: S.12]

Im Rahmen der Fachgruppe für Microsoft Betriebssysteme des Incident Response Teams entstand diese Diplomarbeit, welche Methoden zur Vorfallserkennung und zur Vorfallsanalyse unter besonderer Betrachtung von komparativ statischen Systemvergleichen vorstellt.

Im anschließenden Kapitel 1 werden dazu zunächst die Grundlagen dieser Thematik erläutert. Auf dieser Basis folgt im Kapitel 2 eine Beschreibung der Eigenschaften von Methoden der Vorfallserkennung und -analyse. Diese Eigenschaften werden genutzt, um eine Klassifikation der Methoden aufzustellen. Das Kapitel 3 beinhaltet weiterführend die Implementation einer hostbasierten, komparativ statischen Methode dargelegt anhand der Unified Modeling Language (UML). Versuchsreihen des IRT mit dieser Implementation werden im Kapitel 4 ausgewertet. Resultierend aus diesen Versuchen werden im Kapitel 5 die Grenzen der implementierten Methode aufgezeigt, bevor im Kapitel 6 ein Ausblick vorgenommen wird.

1 Grundlagen

Wie auf allen Gebieten gibt es auch in der Informatik Fachbegriffe, die in verschiedenen Anwendungskontexten unterschiedlich interpretiert und definiert werden. Um Verwirrungen auszuschließen, werden für die vorliegende Diplomarbeit in diesem Kapitel die verwendeten Begriffe erläutert und in Zusammenhang gebracht. Dazu wird zunächst erklärt, was IT-Systeme sind, bevor dann auf die Sicherheitsanforderungen an die Systeme eingegangen wird.

1.1 IT-Systeme

Der Begriff „System" entstammt dem Griechischen *sýstēma* (griechisch für Zusammenstellung) und steht laut [Meyer 1992: System] im deutschen Sprachraum für „einheitlich geordnetes Ganzes". Schlingloff versteht Systeme als Zusammenstellung von interagierenden Komponenten.

> "A system is something which is composed of several components which interact with one another. The complexity of a system is the number of different components and interaction methods." (vgl. [Schlingloff 2001: Def. 2])

In der Informatik werden Systeme (IT-Systeme) laut [Inf.-Duden 1993: S.717] als Systeme mit speziellen Aufgaben verstanden: „ein [IT-]System löst oder bearbeitet in der Regel ein wohl definiertes Bündel von Aufgaben." In RFC 2828 (vgl. [Shirey 2000]) werden die Aufgaben eines IT-Systems bezüglich der zu gewährleistenden Funktionalität verdeutlicht. Demnach gehören Sammeln, Aufzeichnen, Verarbeiten, Aufbewahren, Transportieren, Abrufen und Anzeigen von Daten zu den Aufgaben eines IT-Systems.

DEFINITION

IT-System	An IT-System is "an organized assembly of resources and procedures – i.e., computing and communications equipment and services […] – that collect, record, process, store, transport, retrieve, or display information to accomplish a specified set of functions."

Quelle: [Shirey 2000]

Um die gewünschten Funktionen realisieren und mit der hohen Komplexität umgehen zu können, wurden Systeme in Abstraktionsebenen unterteilt (siehe z.B. das Architekturmodell der International Organisation for Standards (ISO) für offene Netze (Open System Interconnection, OSI) erläutert in [Kerner 1995: 2.2]). Eine mögliche Unterteilung eines IT-System in Schichten wurde in [GBI 2001: 0.5] vorgenommen (s. Abb. 4).

Die unterste Schicht (l_0) stellt dabei die Hardware dar. Sie speichert, verarbeitet und tauscht die Bits aus. Auf dieser Schicht werden die Daten nicht nur lokal verarbeitet, sondern auch über Netzwerke empfangen und gesendet. In der darüber liegenden Schicht (l_1) der Treiber und Firmware werden Basisbefehle zur Verfügung gestellt, die die Steuerung spezieller Hardware ermöglichen. Diese Basisbefehle werden auf der Schicht der Betriebssysteme (l_2) kombiniert, um den Anwendungsprogrammen als Betriebssystemfunktionen komplexe Basisbefehlskombinationen

zur Verfügung zu stellen. Auf der Schicht der Anwendungsprogramme (l_3) befinden sich diejenigen Programme, die zur Lösung spezieller Aufgabenstellungen von den Anwendern benötigt und verwendet werden. In manchen Darstellungen von Computerarchitekturen wird eine weitere, oberste IT-System-Schicht (l_4) eingeführt. Diese Ebene wird als Präsentationsschicht bezeichnet und vermittelt die Funktionen und Daten der Anwendungsprogramme an die Anwender (l_5).

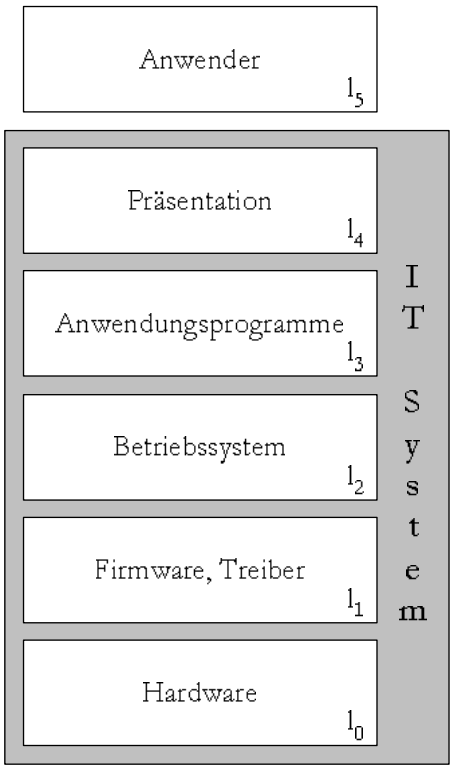


Abb. 4: Grobarchitektur von IT-Systemen nach [GBI 2001: 0.5]

Sun Microsystems sprach bereits 1985 davon, dass Netze ebenfalls als ein Computer-System verstanden werden können: „The Network Is The Computer“ (vgl. [Sun 1985]). Auch die in dieser Diplomarbeit gewählte Definition für IT-Systeme ermöglicht eine derartige Auffassung. Werden Netzwerke und deren Komponenten ebenfalls als IT-System betrachtet, dann können Probleme bei der Anwendung der Grobarchitektur auftreten, da vernetzte Router als Netz nur selten über Betriebssysteme (meist nur Firmware) und nie über Anwendungen verfügen. Befinden sich allerdings Gateways, Proxies und ähnliche Komponenten in einem Netzwerk oder werden Server und Clients zum Netzwerk dazugezählt, dann werden wieder alle Schichten verwendet (ausgenommen die Präsentationsschicht, die üblicherweise nur auf den Clients genutzt wird). Sofern im Folgenden aus dem Kontext nichts Gegenteiliges deutlich wird, wird der Begriff „IT-Systeme“ möglichst allgemein verwendet, also auch dann, wenn ein Gerät nicht über alle Schichten der Grobarchitektur verfügt. Jedoch werden nicht Netzwerke, sondern lediglich die einzelnen Komponenten der Netzwerke, wie zum Beispiel Router und Gateways als IT-System verstanden.

Neben der Zergliederung von IT-Systemen in Schichten besteht bereits eine weitere Aufteilung seit dem Beginn der Computerentwicklung. Sie unterscheidet Programme (bzw. Instruktionen oder Funktionen) und Daten. Damals wie auch heute werden basierend auf Gattern und minimalen Schaltwerken Grundfunktionen wie die Addition, Subtraktion, Multiplikation und Division bereitgestellt. Die Daten werden über Leitungen an die Gatter und Schaltwerke, also in die Funktionen, gebracht (vgl. [Schiffmann & Schmitz 1996: 4 & 6]). Mit John von Neumanns

Computerarchitektur (vgl. [Stallings 2000: 3.1]) wird es ermöglicht, dass Befehle und Daten zwar im selben Speicher aufbewahrt werden, eine Trennung besteht aber immer noch.

Diese Trennung von Daten und Programmen lässt sich vereinfachend auf die Grobarchitektur anwenden: So können wie bei den Schichten des ISO OSI-Modells (vgl. [Kerner 1995]) die Daten und Befehle bzw. Programme einer Schicht als die Daten einer untergeordneten Schicht angesehen werden. Ein derartiges Verständnis von IT-Systemen erleichtert die Zuordnung von Sicherheitsvorfällen zu einer Schicht, in der der Vorfall seinen Ursprung hat.

In der Praxis ist es seit wenigen Jahren allerdings üblich geworden, dass die Daten auch Programme für dieselbe Schicht beinhalten. So können zum Beispiel Texte in Microsoft Word Dokumenten Visual Basic for Applications Code beinhalten (vgl. [aVTC 2003: S.8]). Auch Web-Dokumente werden sehr häufig mit so genannten „aktiven Inhalten" ausgestattet, die beim Lesen der Daten interpretiert und ausgeführt werden (vgl. [Feldmann & Senoucci 2002: S.15]). Somit findet die Trennung von Programmcode und Daten nicht mehr auf Dateiebene, sondern innerhalb von Dateien statt. Damit wird auch eine Zuordnung zu den Schichten erschwert.

1.2 Sicherheit

In der heutigen Welt werden IT-Systeme immer wichtiger. Nicht nur die Wirtschaft, sondern auch Wissenschaft, Militär und andere Institutionen lassen immer mehr Aufgaben von IT-Systemen erledigen. Mit der Anzahl der Aufgaben wächst auch die Relevanz der IT-Systeme, so dass sie im Kontext von Sicherheitspolitiken und Notfallkonzepten so genannte Vermögensgegenstände (engl. *assets*) darstellen (vgl. [Brunnstein 2003]). Auch nach jüngster Gesetzgebung (vgl. [KonTraG 1998]) müssen diese Vermögensgegenstände entsprechend dem ausgesetzten Risiko gesichert werden, wobei Sicherheit hier als Erhaltung des Wertes verstanden wird.

In [Meyer 1992: Sicherheit] wird „Sicherheit" allgemein erklärt als „Zustand des Unbedrohtseins, der sich objektiv im Vorhandensein von Schutz (-einrichtungen) bzw. im Fehlen von Gefahr (-enquellen) darstellt […]".

Wird Sicherheit auf IT-Systeme bezogen, so hat sich in der Literatur und Forschung (vgl. u.a. [Pfleeger 2000: 1.3], [Brömme 2001: 19], [CC 1999: Teil 1: 4.1.1] und [Shirey 2000]) der Gedanke entwickelt, dass Sicherheit sich aus den Eigenschaften Integrität, Vertraulichkeit und Verfügbarkeit zusammensetzt. Brunnstein stellte diese Eigenschaften in eine Orthogonalitätsbeziehung (s. [GBI 2001: 4.1a]) und diskutierte damit die gegenseitige Unabhängigkeit (s. Abb. 5). Die Unabhängigkeit der Aspekte wird in letzter Zeit allerdings wieder relativiert, da zum Beispiel die Verfügbarkeit eines Informationssystems stark eingeschränkt sein kann, wenn die Daten nicht der Integritätsanforderung entsprechen.

Dennoch wird die Sicherheit eines IT-Systems durch Maßnahmen zur Erfüllung von konkreten Anforderungen an die Sicherheitseigenschaften erzielt. Bestehen die Anforderungen an ein IT-System zum Beispiel nur aus der Integrität der Daten, dann kann der Einsatz von Checksummenverfahren ausreichen, um die Sicherheit des Systems zu überwachen. Entsprechend können Firewalls die Vertraulichkeit und Notfallrechensysteme die Verfügbarkeit eines Objektes erhöhen.

Integrität (engl. *integrity*) wird im Zusammenhang mit Sicherheitsanforderungen eines IT-Systems meistens nur auf die Daten bezogen. Entsprechend legt die ISO-Norm 7498 im Teil 2 fest, dass Integrität die Eigenschaft von Informationen ist, die ausschließt, dass die Daten durch einen unzulässigen Vorgang modifiziert oder vernichtet werden.

DEFINITION

Integrität Integrity is "the property that information has not been modified or destroyed in an unauthorized manner."

Quelle: [ISO 7498-2]

Auch die Vertraulichkeit (engl. *confidentiality*) wird bei IT-Systemen meistens nur im Zusammenhang mit den Daten gesehen. Entsprechend definiert die ISO die Vertraulichkeit als Eigenschaft, dass Informationen keinem Unberechtigten (z.B. natürlichen oder juristischen Personen sowie Prozessen) verfügbar oder bekannt gemacht werden dürfen.

DEFINITION

Vertraulichkeit Confidentiality is "the property that information is not made available or disclosed to unauthorized individuals, entities, or processes."

Quelle: [ISO 7498-2]

Die ISO-Norm 7498 definiert die Verfügbarkeit (engl. *availability*) als die Eigenschaft von IT-Systemen oder deren Ressourcen, dass sie zugreifbar und nach Verlangen durch einen Berechtigten nutzbar sind (s. [ISO 7498-2]). In RFC 2828 wird diese Definition dahingehend ergänzt, dass das IT-System nur entsprechend ihrer Performanz-Spezifikation verfügbar sein muss.

DEFINITION

Verfügbarkeit Availability is "the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system."

Quelle: [Shirey 2000]

Üblicherweise werden die Anforderungen an die Verfügbarkeit in Prozent (z.B. benötigte Verfügbarkeitsstunden pro Tag, vgl. [Kefk 2003]), an die Vertraulichkeit entsprechend den Orange Book Kategorien (D, C1, C2, B1, B2, B3 und A1; vgl. [TCSEC 1985]) und die Integrität in „erfüllt" oder „nicht erfüllt" angegeben. Andere Metriken sind aber auch möglich und sinnvoll. So kann die Vertraulichkeit auch in den Evaluation Assurance Leveln (EAL) 1 bis 7 der Common Criteria (vgl. [CC 1999: Teil 3: 6.2) angegeben werden. Bezieht man in die Überlegungen für eine Integritätsmetrik auch noch die Schichten der Computerarchitektur ein, dann kann wie die Verfügbarkeit auch die Integrität in Prozent angegeben werden: Die Bits auf einer Festplatte der Hardwareschicht brauchen nicht integer zu sein, wenn die Werte auf der Applikationsschicht rekonstruiert werden können. Werden zum Beispiel die Binärwerte 0 und 1 der Applikationsschicht auf der Hardwareschicht mit 111 bzw. 000 gespeichert, dann reicht eine Garantie aus, dass maximal eins von drei Bits verfälscht wird, um auf der Applikationsschicht den ursprünglichen Wert durch einen Fehlerkorrektur-Code (s. [T1 1998]) ermitteln zu können. Aufgrund der unterschiedlichen Metriken, die angewandt werden können, werden Sicherheit und ihre Eigenschaften im Folgenden allgemein und nicht auf einen speziellen Maßstab bezogen betrachtet.

Mit den jeweilig gewählten Metriken kann eine Anforderung an eine Sicherheitseigenschaft durch konkrete Werte bestimmt werden. Diese Werte sind kritische Schranken, deren Unterschreitungen für die Sicherheit eines IT-Systems im Sinne einer Sicherheitspolitik nicht tolerierbar sind. Die kritischen Werte der Vertraulichkeit, Integrität und Verfügbarkeit bestimmen dann gemeinsam die minimale Sicherheitsanforderung (s. Abb. 5).

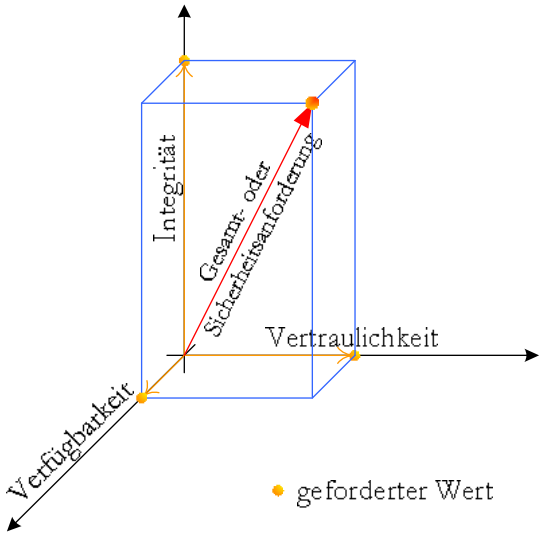


Abb. 5: Sicherheitsanforderungen und Orthogonalität der Sicherheitseigenschaften nach [GBI 2001: 4.1a]

Brunnstein erweiterte die aufgeführten Sicherheitseigenschaften in [GBI 2001: 4.1a] noch um die Aspekte Verlässlichkeit, Persistenz, Funktionalität und Korrektheit, um Anforderungen an beherrschbare IT-Systeme aufzustellen. Nicht selten wird auch Authentizität als Sicherheitseigenschaft angeführt. Diese Aspekte stellen in den meisten Anwendungsfeldern wichtige Anforderungen dar. Sie sind zum einen aber nicht unabhängig von Integrität, Vertraulichkeit und Verfügbarkeit und zum anderen nur selten im für diese Diplomarbeit relevanten Blickfeld von Incident Response Teams.

1.3 Computervorfälle

Howard und Longstaff definieren Computervorfälle (engl. *incidents*) als „eine Gruppe von Angriffen, die dadurch charakterisiert werden können, dass es Unterschiede bei den Angreifern, Angriffen, Motiven, Angriffsstätten und beim Zeitpunkt gibt“ (aus dem Englischen nach [Howard & Longstaff 1998: 5.3], dargestellt in Abb. 6).

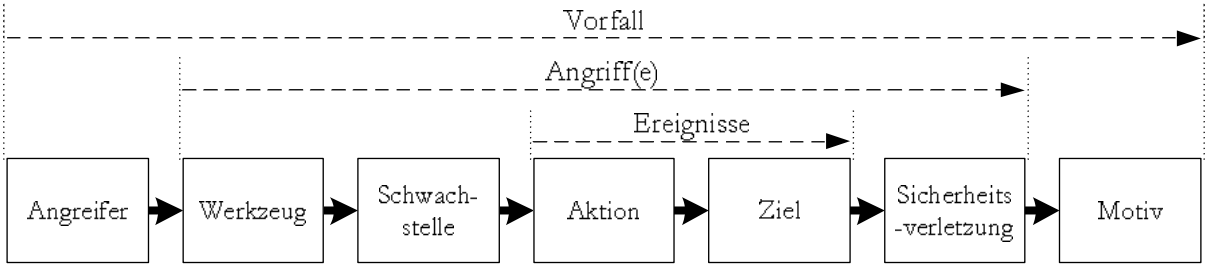


Abb. 6: Vorfallsdefinition nach [Howard et al. 1982: 5.3]

Diese Definition von Vorfällen deckt alle erforderlichen Erkenntnisse einer herkömmlichen, kriminologischen Untersuchung einer Straftat zur Ermittlung einer Kausalkette ab. Sie ist für eine ganzheitliche Betrachtung von Vorfällen bezogen auf IT-Systeme aber zu eng gefasst, da sie vor allem auch Motive berücksichtigt und durch Zufall begründete Vorfälle wie den ersten Wurm im Xerox Palo Alto Research Center (s. [Shoch & Hepps 1982]) ausschließt. In der forensischen Informatik spielt die Definition von Howard und Longstaff jedoch eine sehr wichtige Rolle, da die Forensik im starken Zusammenhang mit der Kriminalistik steht.

Neben dieser Definition wird in der Literatur die Ansicht vertreten, ein Vorfall sei das, was Howard und Longstaff als Angriff (s. Abb. 6) bezeichnen. Ein Vorfall reduziert auf Angriffe im Sinne von [Howard et al. 1982: 5.3] bestände aus der Kette: *Werkzeug* zur Ausnutzung einer *Schwachstelle*, mit der eine oder mehrere *Aktionen zielgerichtet* durchgeführt werden können, um eine *Sicherheitsverletzung* herbeizuführen.

Die kompaktesten Definitionen finden sich in RFC 2828 (vgl. [Shirey 2000]):

> "[a] security incident [is a] security event that involves an act or event that disobeys or otherwise breaches security policy. In other words, a security-relevant system event in which the system's security policy is disobeyed or otherwise breached"

und in [Wack 1991: 1.3]:

> "a computer security incident, [...] is any adverse event whereby some aspect of computer security could be threatened: loss of data confidentiality, disruption of data or system integrity, or disruption or denial of availability"

und reduzieren Sicherheitsvorfälle auf die Kompromittierung von Integrität, Vertraulichkeit oder Verfügbarkeit.

Bezieht man in die Diskussion die Aspekte von geforderten Sicherheitsniveaus (kritische Schranken) ein, dann gelangt man zu folgender Definition, die im Weiteren verwendet werden soll:

DEFINITION

Vorfall	Ein Computer(-sicherheits-)vorfall ist die Unterschreitung eines verlangten Niveaus einer oder mehrerer Sicherheitseigenschaften eines IT-Systems.

Bei einer mathematischen Betrachtung dieser informellen Definition können folgende Aussagen getroffen werden:

Der zu betrachtende Raum für IT-Sicherheit sei SEC und hänge ab von den Sicherheitseigenschaften c für Vertraulichkeit, i für Integrität und a für Verfügbarkeit. Entsprechend der Definition von Sicherheit, sind C, I und A Mengen von geordneten Elementen, die die Niveaus der entsprechenden Eigenschaften repräsentieren. Dann lässt sich SEC formal definieren:

Sicherheitsraum: $$\mathrm{SEC} := \left\{ \langle c, i, a \rangle : \forall c \in C, i \in I, a \in A \right\}$$

Seien c(t), i(t) und a(t) die konkreten Werte von Vertraulichkeit, Integrität und Verfügbarkeit zu einem bestimmten Zeitpunkt t, dann kann der Sicherheitszustand sec(t) abhängig von t definiert werden.

Sicherheitszustand: $\mathrm{sec}(t) := \langle c(t), i(t), a(t) \rangle \in \mathrm{SEC}$

Der Sicherheitsraum besteht also aus der Menge aller möglichen Sicherheitszustände.

Seien c_{req}, i_{req} und a_{req} die kritischen Schranken, also die Werte, die in den Sicherheitsanforderungen an Vertraulichkeit, Integrität und Verfügbarkeit mindestens gefordert werden, dann sei sec_{req} die Gesamtsicherheitsanforderung an ein IT-System:

Sicherheitsanforderung: $\mathrm{sec}_{req} := \langle c_{req}, i_{req}, a_{req} \rangle \in \mathrm{SEC}$

Ferner sei die Relation $<_{sec}$ auf C, I und A wie folgt definiert, sofern x und y aus derselben Menge C, I oder A stammen:

Relation $<_{sec}$ $\quad x <_{sec} y \equiv \begin{cases} \textit{wahr} & \text{wenn y ein höheres Niveau darstellt als x} \\ \textit{falsch} & \text{sonst} \end{cases}$

Unter der Annahme $\mathrm{sec}_a = \langle c_a, i_a, a_a \rangle, \mathrm{sec}_b = \langle c_b, i_b, a_b \rangle \in \mathrm{SEC}$ für $c_a, c_b \in C$, $i_a, i_b \in I$, $a_a, a_b \in A$ sei die Relation $<_{SEC}$ auf der Menge SEC aller möglichen Sicherheitszustände definiert:

Relation $<_{SEC}$ $\quad \mathrm{sec}_a <_{SEC} \mathrm{sec}_b \equiv \begin{cases} \textit{wahr} & \text{wenn } (c_a <_{sec} c_b) \vee (i_a <_{sec} i_b) \vee (a_a <_{sec} a_b) \\ \textit{falsch} & \text{sonst} \end{cases}$

Ein Vorfall liegt im Zeitpunkt t_i genau dann vor, wenn die Vorfallsrelation v den Wert *wahr* annimmt:

Vorfallsrelation: $v(t_i) := \mathrm{sec}(t_i) <_{SEC} \mathrm{sec}_{req}$

und nach Anwendung von $<_{SEC}$ folgt:

$$v(t_i) = (c(t_i) <_{sec} c_{req}) \vee (i(t_i) <_{sec} i_{req}) \vee (a(t_i) <_{sec} a_{req})$$

Bei Betrachtung dieser Vorfallsrelation kann ein Vorfallsraum VR in SEC aufgespannt werden, der durch den Vektor sec_{req} und damit durch c_{req}, i_{req} und a_{req} bestimmt ist (vgl. Abb. 7).

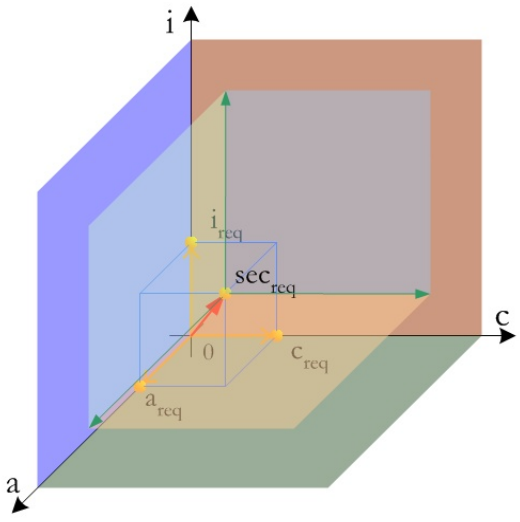


Abb. 7: Vorfallsraum VR

Wenn zum Beispiel ein Dieb ein IT-System entwendet und damit die Verfügbarkeit des Systems den Sicherheitsanforderungen nicht mehr entspricht, dann liegt ein Sicherheitszustand aus dem Vorfallsraum vor.

Vorfälle müssen nicht wie beim genannten Bespiel im gesamten IT-System ursächlich sein. Der Dieb im Beispiel hat ja nicht nur die Hardware (Schicht l_0), sondern mit dem System auch alle in ihm vertretenen Schichten gestohlen. Um die Verfügbarkeit unter eine kritische Marke ($a(t) < a_{req}$) herabzusetzen, hätte es ausgereicht, bestimmte Treiber auf Schicht l_1 zu löschen. Andererseits hätte es bei einem RAID5-System nicht ausgereicht, eine Festplatte zu stehlen, da aufgrund der RAID5-Architektur die Daten der gestohlenen Festplatte im laufenden Betrieb rekonstruiert werden können. Betrachtet man also die Sicherheitsanforderungen eines IT-Systems, dann müssen diese an alle Schichten der Computerarchitektur eventuell mit unterschiedlichen kritischen Werten dargestellt werden.

Ein Sicherheitsvorfall auf einem IT-System könnte dann auch als Vektor v bestehend aus den Werten der drei Sicherheitseigenschaften zum Zeitpunkt t_k (also einem Sicherheitszustand) und einem Vektor L der ursächlich betroffenen Systemschichten definiert werden:

Vorfallsvektor:

$$\vec{v}_k := \left\langle c(t_k), i(t_k), a(t_k), \vec{L} \right\rangle$$

$$\text{mit } \vec{L} = (l_0, l_1, l_2, l_3, l_4, l_5) : l_0, l_1, l_2, l_3, l_4, l_5 \in \{\text{wahr}, \text{falsch}\}$$

$$\text{für } sec(t_k) <_{SEC} sec_{req} \text{ mit } sec(t_k) = \left\langle c(t_k), i(t_k), a(t_k) \right\rangle \text{ und}$$

$$\text{folglich } sec(t_k) \in VR$$

1.3.1 Klassifikation von Computervorfällen

Eine Klassifikation von Vorfällen nach den Architekturschichten, in denen sie ursprünglich begründet sind, scheint nicht zufrieden stellend zu sein. Der Vektor L deutet an, dass es Vorfälle gibt, die gleichzeitig auf verschiedenen Ebenen agieren. Der Wurm W32/Nimda.A@MM (vgl. [Mackie, Roculan, Russell & Van Velzen 2001]) nutzt zum Beispiel sowohl Schwächen auf der Anwendungsschicht (l_4, Email-Client) als auch des Betriebssystems (l_3, offene Netzwerkfreigaben) aus. Damit wäre eine solche Klassifikation nicht zwingend eindeutig.

R. Moses schlägt in [Jackson & Hruska 1992: 21.3.2.2] eine Klassifikation für Sicherheitsbedrohungen und den daraus resultierenden Vorfällen vor, die zunächst zwischen zufälligen und beabsichtigten Vorfällen unterscheidet.

Zu der Gruppe der zufälligen Vorfälle zählen:

- *Ereignisse höherer Gewalt*
  Darunter werden im Allgemeinen Vorgänge verstanden, auf die Menschen keinen oder nur minimalen Einfluss haben. Beispiele für höhere Gewalt sind Blitzschlag, Wassereinbruch, Witterungseinflüsse, Feuer, Ausfall von Mitarbeitern usw.

- *Menschliches Versagen*
  Zu dieser Klasse gehören Fehlspezifikationen, die zu Mängeln in der Software- oder Hardware-Architektur führen, Fehlimplementationen (so genannte Bugs) und Fehlkonfigurationen sowie sonstige, unbeabsichtigte Handlungen.

- *Fehlfunktionen der Ausstattung und Software*
  Elemente dieser Klasse können zum Beispiel durch Produktionsfehler wegen mangelnder Keimfreiheit der Luft verursacht werden oder in Stromschwankungen oder ähnlichem begründet sein. Auch Ausfälle aufgrund von Materialermüdung gehören in diese Klasse.

Die zweite Gruppe enthält laut [Jackson et al. 1992: 21.3.2.2] Vorfälle, die von Menschen vorsätzlich verursacht werden:

- *Systeminfiltration*
  Die Infiltration eines IT-Systems erfolgt immer durch nicht autorisierte Personen und Prozesse, die sich eines Systems oder dessen Ressourcen bemächtigen. Dazu gehören zum Beispiel Datenspionage, -modifikation und -löschung, sowie das Ändern der Datenzugriffsrechte.

- *Missbrauch von Ressourcen*
  Unter Missbrauch von Ressourcen werden Vorfälle verstanden, bei denen Personen und Prozesse, die ein zweck- oder zielgebundenes Recht auf Nutzung einer Ressource haben, diese nicht im Sinne des Zweckes oder Zieles einsetzen. Zum Beispiel wäre der Gebrauch eines IT-Systems für Geschäftsvorgänge in einem Unternehmen legitim, jedoch könnte die Nutzung dieses Computers zum Spielen als Missbrauch angesehen werden.

- *Absichtliche Schädigung*
  Vorfälle dieser Gruppe schließen sowohl physikalische als auch logische Vorfälle ein. Eine absichtliche, physikalische Schädigung war zum Beispiel der erste Computervorfall überhaupt, als ein Student der Texas University auf den Zentralprozessor eines Großrechners mit einem Revolver schoss (vgl. [Brunnstein 2003]). Auf der logischen Ebene gehören viele destruktive Viren und Würmer zu dieser Klasse.

- *Diebstahl*
  Unter Diebstahl wird nicht nur die physikalische Entfernung eines IT-Systems oder deren Komponenten verstanden, sondern auch das Stehlen von Daten und Software, indem sich der Angreifer eine Kopie erstellt.

1.3.2 Angriffstechniken

Die vorangegangene Klassifikation legt den Schwerpunkt auf die Ursachen von Vorfällen und die Ziele bzw. Position der Verursacher. Sie sagt aber nichts über die dahinter stehenden Techniken aus. Zwar können die meisten Angriffstechniken der Gruppe der absichtlichen Vorfälle zugeordnet werden, jedoch können diese Techniken auch zufällig auftreten. IT-Systeme mit einer fehlerhaften Implementation der TCP/IP-Protokollfamilie (vgl. [MS KB 154174]) können mit sehr großen ICMP-Echo-Request-Paketen zum Absturz gebracht werden. Wird ein solches Paket aufgrund eines zufälligen Tippfehlers abgeschickt, dann wird die Verfügbarkeit eines IT-Systems genauso herabgesetzt, wie beim absichtlichen Versand des Pakets zum Zwecke eines so genannten Denial of Service-Angriffs.

Da die Palette der Angriffstechniken sehr groß ist, die bei vielen Vorfällen auch noch kombiniert angewendet werden (vgl. [Hoherz, Krüger, Menne & Michaelsen 2001: 5]), werden im

Folgenden anhand des in [GBI 2001: 3.1] vorgestellten Katalogs nur die am meisten genutzten Angriffstechniken erklärt.

Port Scanning

Laut [Mutlu, Schnell & Yüksel 2001: 2.4.5 & 3.2.1.3] dient Port Scanning bei vielen Angriffen zur Vorbereitung. Auf der Transportschicht wird für die verschiedenen Protokolle (wie etwa TCP und UDP) ermittelt, welche Dienste der höheren Schichten, die mittels Ports über ein Netzwerk adressiert werden können, zur Verfügung stehen. Laut [Metterhausen 2003] kann aufgrund der offenen Ports auch darauf geschlossen werden, welches Betriebssystem auf dem Opfersystem läuft. Ein Angreifer erhält somit einen Überblick, welche Dienste er auf welchem Betriebssystem eventuell für seine Zwecke nutzen kann.

Unter eher unüblichen Umständen kann ein Portscan die Verfügbarkeit eines Systems so sehr herabsetzen, dass es den Sicherheitsanforderungen nicht mehr genügt. Da bei einem Portscan das gescannte System auf den offenen Ports Meldungen erhält, diese verarbeiten und gegebenenfalls beantworten muss, kann ein Portscan bereits als Missbrauch von Ressourcen aufgefasst werden.

Sniffing

Als Sniffing (engl. für schnüffeln) wird das Mitlesen von Daten auf Netzen bezeichnet (vgl. [Shirey 2000] und [GBI 2001: 1.2b]). Dazu muss ein Angreifer Zugriff auf die versendeten Datenpakete erhalten. Den Zugriff kann er dadurch erlangen, dass er die Gewalt über ein IT-System (z.B. ein Gateway) auf der Route zwischen den Kommunikationspartnern erlangt (s. Abb. 8).

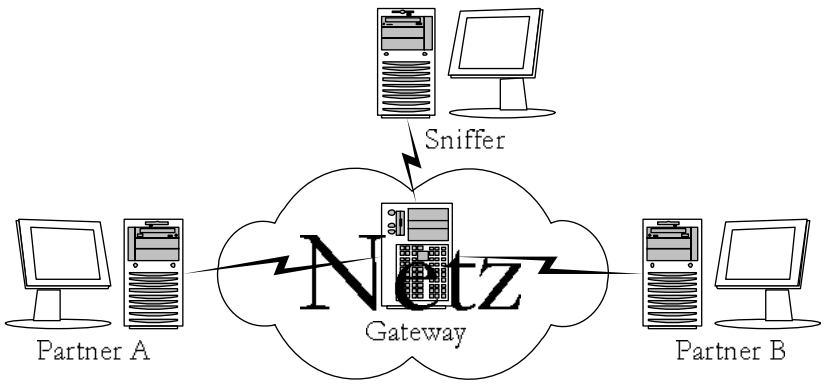


Abb. 8: Position eines Sniffers im Netz

Die Datenpakete kann der Angreifer ebenfalls mitlesen, wenn er Zugriff auf einen Computer hat, der an einem Netzsegment angeschlossen ist, in dem alle Nachrichten an alle verbundenen Systeme gesendet werden (zum Beispiel beim Einsatz von Hubs) und durch das der Verbindungsweg der Kommunikationspartner läuft. Sofern keine weitere Angriffstechnik dazu kommt, kann mit Sniffing „nur“ Datenspionage betrieben werden.

Sniffing wird aber nicht nur als Angriffstechnik eingesetzt. Ursprünglich wurde diese Technik von Administratoren eingeführt, die wissen wollten und durften, was in dem von ihnen betreuten Netzwerk passiert (vgl. z.B. [Stoll 1999]). Die Legitimation wird in solchen Fällen mit dem Verantwortungsbereich der Administratoren begründet, für die dann andere Sicherheitsanforderungen gelten können.

Spoofing

Bei der Adressfälschung (engl. s*poofing*) stehen dem Angreifer verschiedene Möglichkeiten zur Verfügung, um einem Nutzer eine falsche Identität vorzutäuschen (vgl. [Shirey 2000] und [GBI 2001: 1.2c]). Dabei kann er seine Pakete mit einer falschen Absenderadresse versehen, damit der Adressat glaubt, von einem anderen die Informationen zu erhalten. Der Angreifer kann alternativ auch mit etwas Aufwand die Einträge in den DNS-Listen ändern, in denen Domainnamen IP-Adressen zugeordnet werden. Im letzteren Fall wäre dann auch eine bidirektionale Kommunikation möglich. Diese würde bei einer gefälschten Absenderadresse scheitern, da die Antworten des ursprünglichen Empfängers bei demjenigen eingehen würden, dessen Identität der Angreifer vortäuscht. Auch andere Manipulationen der Kommunikationsroute sind bei Spoofing für bidirektionalen Datenaustausch denkbar.

Session Hijacking

Wie beim Sniffing muss ein Angreifer beim Session Hijacking (engl. für Verbindungsentführung) auf eine Komponente des Verbindungsweges zwischen den Kommunikationspartnern Zugriff haben. Wenn die Partner kommunizieren, trennt der Angreifer den Kommunikationsweg und übernimmt die Rolle des jeweils anderen Partners. In dem Glauben, mit dem ursprünglichen Partner zu kommunizieren, nehmen die Opfer dieses Angriffs zum Beispiel Befehle entgegen und geben Informationen an den unbefugten Dritten weiter (vgl. [Mutlu et al. 2001: 2.4.6.6]).

Flooding

Das Fluten (engl. *flooding*) von IT-Systemen (insbesondere von Netzen) erfolgt dadurch, dass man ihnen mehr Daten zusendet, als sie verarbeiten können (vgl. [Shirey 2000]). Bei schlechter Konfiguration oder Implementation des IT-Systems werden dann so viele Ressourcen für die Verarbeitung der zugesandten Daten verbraucht oder reserviert, dass die eigentlichen Aufgaben durch das System nicht mehr wahrgenommen werden können. Das Ergebnis eines solchen Angriffs sind häufig Dienstverweigerungen (engl. *Denial of Service,* DoS), wie sie im Februar 2000 bei Yahoo!, Amazon.com, CNN und eBay auftraten (vgl. [CNN 2000]).

Angriffe auf Authentizität

Die Authentizität eines Anwenders oder Prozesses wird mittels Identifikation und Authentifikation festgestellt. Meldet sich zum Beispiel ein Benutzer an einem System an, so identifiziert er sich mit seinem Benutzernamen und authentifiziert sich mit dem dazugehörigen Passwort. Ein Angriff auf die Authentizität kann zum Beispiel dadurch erfolgen, dass ein fremder Benutzername angegeben und das dazugehörige Passwort erraten wird. Das IT-System geht dann davon aus, dass der eigentliche Inhaber des Benutzernamens sich angemeldet hat (vgl. [GBI 2001: 3.3]).

Bösartige Software

Viren, Würmer und trojanische Pferde sind wohl die bekanntesten Vertreter bösartiger Software. Sie werden unter dem Begriff *Malware* zusammengefasst, der sich aus den englischen Worten *malicious* (bösartig) und *software* zusammensetzt. Malware ist dahingehend bösartig, dass sie Funktionen enthält, deren Ausführung absichtlich die Sicherheit eines IT-Systems kompromittiert. Dies geschieht meist für den Anwender zunächst unerkennbar.

DEFINITION

Malware "A software or module is called 'malicious' ('malware') if it is intentionally dysfunctional, and if there is sufficient evidence (e.g. by observation of behavior at execution time) that dysfunctions may adversely influence the usage or the behavior of the original software."

Quelle: [Brunnstein 1999: Def. 4]

Mit dieser Definition wird Software zur Malware, wenn die Software absichtlich Dysfunktionen enthält – also Funktionen, die in der Spezifikation des Herstellers nicht festgelegt wurden oder vom Anwender nicht erwartet werden müssen und absichtlich implementiert wurden. Solche Dysfunktionen erzeugen bei dem Anwender von Malware meistens einen Schaden im Sinne einer Sicherheitsanforderungsverletzung. Deshalb spricht man bei Dysfunktionen auch von Schadfunktionen oder Payloads (engl. für Ladung).

Die Klassifikation von Malware erfolgt entsprechend der genutzten Verbreitungstechnik, die zusammen mit der Schadfunktion die wesentlichen Charakteristika einer Malware darstellen. Das antiViren Test Center an der Universität Hamburg unterteilt Malware nach der Notwendigkeit der Existenz von Netzen für die Malware-Verbreitung. Des Weiteren erfolgt eine Unterteilung anhand der Fähigkeit der Malware, sich selber zu replizieren (vgl. [aVTC 2003: S.4] und Abb. 9). Unter der Replikationsfähigkeit von Malware wird verstanden, dass mindestens auch noch die zweite selbstständig erzeugte Generation einer Malware-Instanz dieselben Eigenschaften aufweist (vgl. [Brunnstein 1999: Def. 6a]).

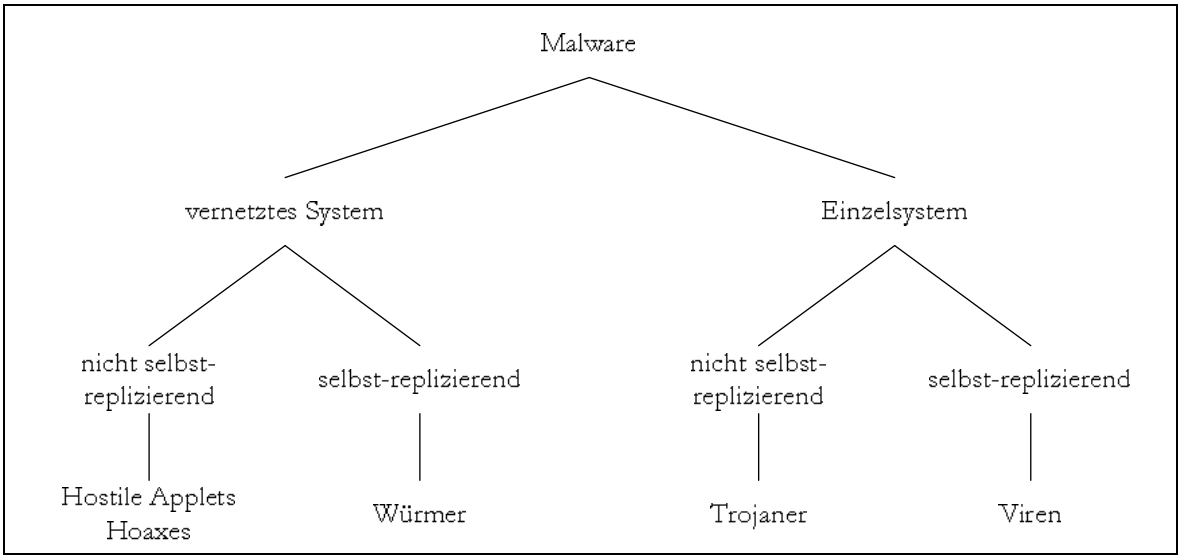


Abb. 9: Klassifikation von Malware aus [aVTC 2003: 4]

Obwohl eine Trennung von Verbreitungstechnik und Schadfunktion theoretisch gemacht werden kann, ist sie in der Praxis nicht immer möglich. Entsteht der Schaden wie beispielsweise beim Wurm W32/Sapphire (vgl. [Hypponen & Erdelyi 2003]) durch die hohe Anzahl der Replikationen, führt dies zu überlasteten Netzen und gefährdet somit die Verfügbarkeit von IT-Systemen. Auch der Schaden überschreibender Viren liegt zumindest teilweise in ihrem Replikationsmechanismus (vgl. [Seedorf 2002: 1.2.3.1]).

Social Engineering

Unter Social Engineering (ursprünglich engl. für angewandte Sozialwissenschaften) wird im Kontext der IT Sicherheit das bösartige Ausnutzen von sozial normalem Verhalten verstanden (vgl. [Granger 2001]). Angriffe dieser Gruppe haben entweder keinen oder einen nur sehr geringen technischen Anteil. Zum Beispiel nutzen so genannte Hoaxes Social Engineering aus, um vermeintlich wichtige Informationen per Mail weiter zu geben. Der Inhalt der Mail, der häufig vor neuen Viren warnt oder zu Spenden für einen leidenden Mitmenschen aufruft, ist dabei frei erfunden, animiert aber den Empfänger einer solchen Mail zur Weiterleitung derselben. In Extremfällen kann das zu Email-Flooding führen. Zumindest handelt es sich jedoch um einen Missbrauch von Ressourcen.

1.4 Incident Response Teams

In den vorangehenden Abschnitten wurde gezeigt, dass an IT-Systeme Sicherheitsanforderungen gestellt und diese durch verschiedene Techniken kompromittiert werden können. Zur Analyse, Bekanntgabe und Entwicklung von Gegenmaßnahmen für solche Vorfälle haben sich seit dem Internetwurm von 1988 mehrere Institutionen herausgebildet.

1.4.1 Computer Emergency Response Team

Im November 1988 startete Robert Morris, Student der Cornell University, ein selbstreplizierendes Programm, das sich im damaligen Internet rasant ausbreitete (vgl. [van Wyk & Forno 2001: S.13]). Obwohl das Programm als so genannter „Internet Worm“ nur etwa 2500 Internetrechner (etwa 4 % des damaligen Internets) aufgrund von UNIX-Softwarefehlern befiel, legte der Wurm aufgrund der Relevanz der befallenen Systeme 70% des Internets für einige Tage „lahm“ (vgl. [Howard 1997: 3.1]). Eine Gruppe von Experten, die unter anderem vom in Boston ansässigen Massachusetts Institute of Technology, der University of California in Berkley und der Purdue University in West Lafayette stammten, nutzten Methoden des Reverse Engineering (s.a. [Janz 2000]), um das Programm zu analysieren und die Softwarefehler von UNIX zu beheben.

Aufgrund dieser Erfahrung beschloss die Defense Advanced Research Projects Agency (DARPA) des Verteidigungsministeriums der USA, eine Art „Feuerwehr für das Internet“ einzurichten (vgl. [Howard 1997: 3.5 & 4.1]). Das Computer Emergency Response Team Coordination Center (CERT/CC) wurde am Software Engineering Institute der Carnegie Mellon University in Pittsburgh noch Ende November 1988 gegründet. Das CERT, wie es häufig auch nur genannt wird, übernahm nach [Kossakowski 1995] die Aufgaben, mit der Internetgemeinschaft zusammen auf Vorfälle bei den Internetrechnern zu reagieren, proaktiv das Bewusstsein für IT Sicherheit zu schärfen und Untersuchungen zur Erhöhung des Sicherheitsniveaus existierender Systeme anzustrengen. Diese Aufgaben nimmt das CERT/CC auch heute noch wahr.

In der Wirtschaft hat der Begriff „Computer Emergency Response Team“ eine weitere Bedeutung erhalten. Innerbetriebliche CERTs übernehmen häufig die Aufgaben der Umsetzung von Notfallplänen und reagieren bei Computervorfällen. Ihnen wird auch die Aufgabe übertragen, Beiträge zur Umsetzung von Sicherheitspolitiken zu leisten. Wenn solche CERTs Vorfälle

analysieren, dann eher um den entstandenen Schaden für das Unternehmen zu ermitteln und die Lücken im betreuten Bereich festzustellen. Sie beziehen ihre Informationen und Kenntnisse meist von Spezialisten wie dem CERT/CC und verteilen diese in ihrem Verantwortungskreis. Ihr Fokus liegt auf den Geschäftsinteressen des Unternehmens, in das sie eingebettet sind, weshalb diese CERTs im Folgenden „Corporate CERTs“ genannt werden.

Andere Namen für Computer Emergency Response Teams sind *Computer Security Incident Response Team* (CSIRT) und *Incident Handling Team* (IHT, vgl. [Kossakowski 2000: Glossar]). Da der Begriff „Computer Emergency Response Team“ als Marke vom CERT/CC registriert wurde, setzt sich CSIRT zurzeit durch.

In einer Arbeitsgruppe der Internet Engineering Task Force (IETF) wurde in den „Guidelines and Recommendations for Incident Processing“ die Aufgaben eines CSIRTs (bzw. eines CERTs oder IHTs) definiert:

> “A Security Incident Response Team should be capable of dealing with incidents that occur within its defined constituency. It should provide a means for reporting suspected incidents and offer technical assistance to help sites handle these incidents. Teams should also disseminate important incident-related information to relevant parties.” (vgl. [Kossakowski 2000: 437])

Die Aufgaben beziehen sich also immer nur auf die Klientel eines CSIRT, wobei im Falle des CERT/CC die Klientel aus allen Internet-Betreibern und -Nutzern besteht und im Falle der Corporate CERTs aus den anderen Abteilungen des Unternehmens.

1.4.2 Das Incident Response Team an der Universität Hamburg

Das Incident Response Team der Universität Hamburg fällt nicht ganz in den Rahmen der eben aufgeführten CSIRTs. Das IRT, wie dieses CSIRT im Folgenden genannt wird, wurde nach der Privatisierung und dem Weggang des CERT des Deutschen Forschungsnetzes (DFN-CERT) von Prof. Dr. Brunnstein am Fachbereich Informatik im Oktober 2002 gegründet. Seitdem erarbeitet das IRT neben der reinen Vorfallsanalyse auch Methoden für Computer Security Incident Response Teams, die anhand im Labor durchgeführter Vorfälle getestet und weiterentwickelt werden.

1.5 Aufgabengebiet des IRTs und anderer CSIRTs

Die Aufgabengebiete von CSIRTs werden häufig sehr individuell festgelegt, wobei die Gemeinsamkeit darin liegt, dass sie sich mit Computervorfällen beschäftigen (vgl. [Kossakowski 2000: S.13]). Das Schaubild der Abb. 10 zeigt, wie sich die Aufgaben des IRT von üblichen Corporate CERTs unterscheiden. In der Grafik wird zunächst eine Kausalkette aufgezeigt:

Ausgegangen wird von einem Risiko (engl. *risk*). Dieses besteht darin, dass IT-Systeme mit Schwachstellen gewissen Bedrohungen gegenübergestellt sind (vgl. [GBI 2001: 4.2a], [Hoherz 2003: 1.1.1] und [Michaelsen 2003: 2.2]).

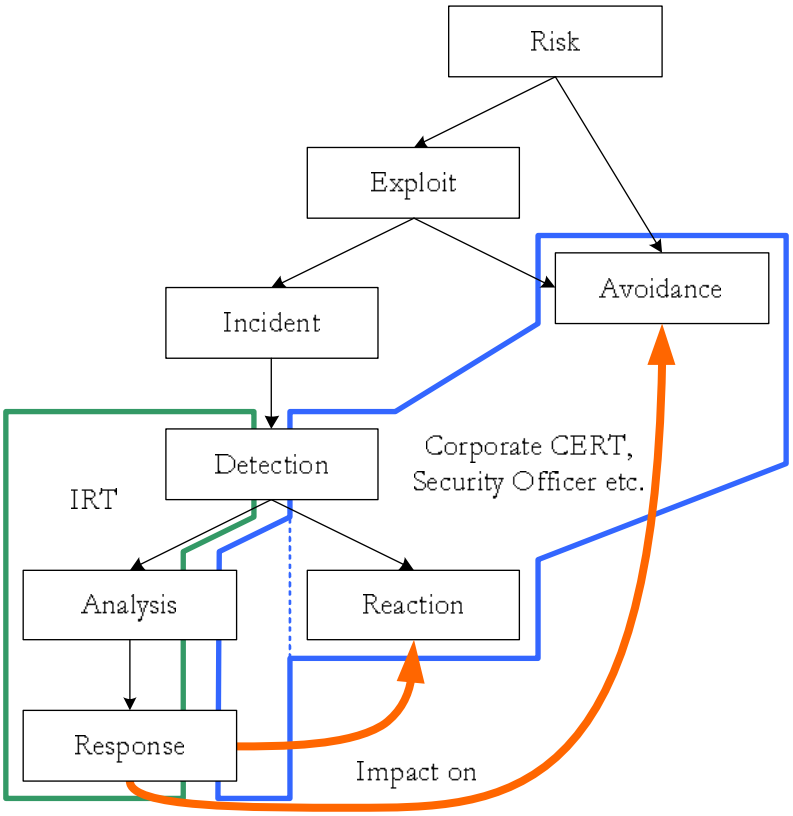


Abb. 10: Aufgabengebiete bei Computervorfällen aufbauend auf [IRT 2003a: F.3]

Auf ein Risiko mit Abwehrmaßnahmen zu reagieren, wird Vermeidung (engl. *avoidance*) genannt. Das kann zum Beispiel ein Web-Server sein, der durch eine Firewall geschützt wird, durch die nur Datenpakete für den HTTP-Port (TCP 80) hindurch gelangen. Auch das Einspielen von Patches und Bug-Fixes, also von Software, die unter anderem Schwachstellen beheben soll, zählt zu den Maßnahmen, die unter Avoidance fallen. Diese Aufgabe kann durch Corporate CERTs oder den für die Umsetzung der Sicherheitspolitik verantwortlichen Security Officer wahrgenommen oder unterstützt werden.

Gibt es zu einer Schwachstelle auch Quellcode oder Software, die die Schwachstelle ausnutzt, dann spricht das IRT von einem Exploit (engl. für Ausnutzung) oder von der „verwirklichten Ausnutzung einer Schwachstelle“. Der Wurm W32/Nimda@MM, der die MIME-Schwächen vom Microsoft Internet Explorer und von Microsoft Outlook ausnutzt (s. [Mackie et al. 2001: 5f.]), stellt somit einen Exploit dar. Diese Definition stimmt überein mit der 2. Definition in [JargonFile 2002: exploit]:

DEFINITION

Exploit	“[…] 1. A vulnerability in software that can be used for breaking security or otherwise attacking an Internet host over the network. […] 2. More grammatically, a program that exploits an exploit in sense 1.”

Quelle: [JargonFile 2002: exploit]

Sobald ein Exploit existiert, können noch einmal gesonderte Abwehrmaßnahmen speziell für diesen Exploit ergriffen werden. AntiMalware-Hersteller bieten zum Beispiel üblicherweise erst Abwehrmaßnahmen an, wenn es bereits einen Exploit gibt, denn Produkte zur Malware-

Bekämpfung (so genannte AntiMalware) erkennen und beseitigen Malware und bessern keine Schwachstellen aus.

Wird ein Exploit auf ein konkretes IT-System angewandt, dann handelt es sich um einen Vorfall (engl. *incident*). Ein Vorfall wäre zum Beispiel die Infektion eines Computers durch ein Exemplar des W32/Nimda.A@MM (einer Variante von W32/Nimda@MM).

Die Erkennung (engl. *detection*) des Vorfalls erfolgt häufig aufgrund von Anwendermeldungen oder speziellen Systemen, wie zum Beispiel Intrusion Detection Systemen, und wird sowohl von Corporate CERTs und anderen Incident Response Teams als auch dem IRT durchgeführt.

Je nach Art des erkannten Vorfalls und der Zielsetzung der Teams folgt auf die Vorfallserkennung eine detaillierte Analyse (engl. *analysis*) und/oder eine Reaktion (engl. *reaction*) zur Sicherung von IT-Systemen und der durch sie repräsentierten Werte. Idealerweise wurden die Reaktion und deren Schritte schon vorausschauend in einem Notfallkonzept hinterlegt.

Wird ein Vorfall analysiert, so wird laut der Definition aus [Elder & Wolfe 2001] versucht, die Komplexität des Vorfalls dadurch aufzulösen, dass der Vorfall in viele nachvollziehbare und bestimmbare Einzelteile und -schritte zerlegt wird, um dann durch das Zusammenfassen der Teilanalyseergebnisse zu einem Gesamtanalyseergebnis und bestenfalls zu einem Gesamtverständnis zu gelangen.

DEFINITION

Analyse	"Analysis: 1. The resolution or breaking up of anything complex into its various simple elements, […]; the exact determination of the elements or components of anything complex […]. 2. A tabular statement or other form embodying the results of the above process; an abridgement exhibiting the essential heads; a synopsis or conspectus."

Quelle: [Elder et al. 2001

Aufgrund des Analyseergebnisses und des Verständnisses der Vorgänge während des Vorfalls können Maßnahmen und Konzepte als Antwort (engl. *response*) auf den Vorfall entwickelt werden. Diese Maßnahmen und Konzepte haben häufig Einfluss darauf, wie auf Vorfälle reagiert wird bzw. welche Abwehrmaßnahmen ergriffen werden. Stellt sich zum Beispiel heraus, dass durch eine gewisse Reihenfolge von HTTP-Paketen die Integrität eines Web-Servers verletzt werden kann, dann ist eine Abwehrmaßnahme, eine vorgelagerte Firewall mit Content-Filtering (vgl. [Hoherz et al. 2001: 3.2]) auf der HTTP-Schicht auszustatten, damit solche Pakete im Bedarfsfall analysiert und geblockt werden können.

Auch die Reaktionen auf Vorfälle ähnlicher Art können durch die Erkenntnisse über einen analysierten Incident anders oder neu geplant werden. So wurden zum Beispiel nach der starken Ausbreitung von W32/Melissa.A@MM viele Mitarbeiter von Unternehmen über die Verbreitungstechnik informiert (vgl. [CA-1999-04]) und gebeten, Makros nur auszuführen, wenn sie die Zusendung von Dateien mit Makros von dem Absender erwarten.

2 Vorfallserkennung und -analyse

Im vorangegangenen Abschnitt wurde erklärt, welche Aufgaben von CSIRTs wahrgenommen werden können. Dazu zählen die Erkennung und Analyse von Computersicherheitsvorfällen. Diese werden im Folgenden genauer erläutert, bevor dann Methoden zur computerunterstützten Erfüllung der Aufgabe betrachtet und klassifiziert werden.

2.1 Vorfallserkennung

Der Begriff der Vorfallserkennung kann unterschiedlich verstanden werden. Entsprechend der Beschreibung in [BBAW 2003] bedeutet „erkennen: 1. etwas […] so deutlich [wahrnehmen], dass man weiß, was […] man vor sich hat […] 2. […] aus gewissen Anzeichen einen Tatbestand feststellen." Beide Bedeutungen können auf die Vorfallserkennung übertragen werden:

1) *Vorfallszuordnung (Identifikation)*
   Diese Begriffsinterpretation versteht Vorfallserkennung als Einordnung eines Vorfalls zum Beispiel in die Vorfallsklassifikation aus Abschnitt 1.3.1 oder in die in Abschnitt 1.3.2 aufgeführten Angriffstechniken. Eine solche Zuordnung kann nur Ergebnis einer Analyse sein. Erst aufgrund einer Vorfallsanalyse kann festgestellt werden, dass beispielsweise eine Verfügbarkeitsverletzung aufgrund absichtlicher Schädigung durch Flooding vorliegt und keine Fehlfunktion des IT-Systems.

2) *Vorfallsverdacht (Detektion)*
   Entsprechend der in Abschnitt 1.5 aufgeführten Begriffsinterpretation ist die Vorfallserkennung der Verdacht oder die Feststellung, dass die Sicherheitsanforderungen unterschritten wurden (vgl. Vorfallsdefinition S. 10). Wie der Vorfall zustande gekommen ist, wird in der anschließenden Analyse geklärt. (vgl. [Mandia & Prosise 2001: S.18])

Das erste Begriffsverständnis fasst die Vorfallserkennung also im Prinzip als Zusammenfassung und weitere Auswertung der Analyseergebnisse auf. Dies ist aber laut [Kossakowski 2000: 3.3.5]) noch Teil der Analyse (vgl. auch 1.5 insbesondere den zweiten Aspekt der Analyse-Definition). Entsprechend wird in dieser Diplomarbeit die Vorfallserkennung gemäß der Detektions-Interpretation verwendet.

Kossakowski, Stikvoort und West-Brown führen im „Handbook for Computer Security Incident Response Teams (CSIRTs)" (vgl. [West-Brown, Stikvoort & Kossakowski 1998: 3.3]) eine Sichtungsfunktion bei den CSIRTs ein. Diese überwacht alle Vorfallsmeldungen, die dem CSIRT zugestellt werden. Dabei weisen die Autoren darauf hin, dass diese Meldungen nicht nur technisch automatisiert (z.B. durch Intrusion Detection Systeme o.ä.), sondern auch per Email, Telefon und Fax abgesetzt werden. Entsprechend der Kommunikation (Mensch oder Maschine) erfolgt auch die Vorfallserkennung. So kann ein Web-Server-Ausfall durch einen Anwender erfolgen, der die Web-Site nicht herunterladen kann, oder die Meldung erfolgt tool-gestützt durch ein Überwachungsprogramm.

Nach Eingang einer Vorfallsmeldung folgt die Vorfallsanalyse (vgl. u.a. Abschnitt 1.5, [West-Brown et al. 1998: 3] und [Mandia et al. 2001: S.80]).

2.2 Vorfallsanalyse

In [West-Brown et al. 1998: 3.4.2] werden Vorfallsanalysen, wie folgt, unterteilt:

- *Intra-Vorfallsanalyse*
  Diese Art der Analyse beschäftigt sich mit einem konkreten Vorfall und besteht aus:
  - Protokolldatei-Analysen
  - Artefakt-Analysen
  - Analysen der Software-Umgebung
  - „Web of Relations"-Analysen
- *Inter-Vorfallsanalyse*
  Hier werden Wechsel- und Zusammenwirkung mit anderen Sicherheitsvorfällen ermittelt. Sie dienen der Herstellung eines holistischen Bildes, der Erkennung von Ähnlichkeiten der Vorfälle und dem Ausfindigmachen des eventuell gemeinsamen Angreifers.

In [West-Brown et al. 1998: 3.4.2] wird, wie oben dargestellt, die Analyse eines konkreten Vorfalls weiter aufgegliedert. Die erste dort erwähnte Analyse basiert auf Protokolldateien (engl. *Log-Files*). Einträge in diesen Dateien werden je nach Konfiguration von Anwendungen, Betriebssystemen und Treibern vorgenommen. Üblicherweise werden dort Programmereignisse dokumentiert, die für jegliche Arten der Analyse interessant sein könnten.

Die zweite genannte Intra-Vorfallsanalyse basiert auf vorgefundenen Artefakten. In [West-Brown et al. 1998: 3.4.2.4] werden alle Spuren, die ein Angreifer bei einem Computersicherheitsvorfall auf einem IT-System hinterlässt, als Artefakte bezeichnet. Kossakowski, Stikvoort und West-Brown nennen als Beispiele die Log-Files von Netzsniffern, Passwortdateien, Trojanern, ausgetauschte Dateien und Programme, Skripte für Exploits sowie Quellcodedateien, die von einem Eindringling hinterlassen werden.

Vor allem Skripte, Quellcode-Dateien und auch ausgetauschte Programme können von einem Hacker nur unter bestimmten Voraussetzungen genutzt werden. Diese werden in der (dritten) Analyse der Software-Umgebung untersucht. Laut [West-Brown et al. 1998: 3.4.2.5] ist diese Analyse wichtig, da Quellcode-Dateien, Programme und Skripte nur mit den entsprechenden Compilern, Programmbibliotheken und Interpretern auf einem IT-System genutzt werden können. Ein Visual Basic Script Artefakt kann auf einem IT-System mit Microsoft Windows beispielsweise nicht genutzt werden, wenn der Windows Scripting Host (WSH) nicht installiert ist (vgl. [Feldmann et al. 2001]).

Als vierte und letzte Intra-Vorfallsanalyse führen Kossakowski, Stikvoort und West-Brown die Untersuchung des „Web of Relations" (engl. für Beziehungsnetz) an. Während die Inter-Vorfallsanalyse Beziehungen zwischen verschiedenen Computersicherheitsvorfällen aufdeckt, wird bei der Untersuchung des „Web of Relations" die Beziehungen innerhalb eines Vorfalls ermittelt. Insbesondere wird untersucht, in welcher zeitlichen Reihenfolge die gefundenen Artefakte und Log-File-Eintragungen stehen. Aufgrund der Analyseergebnisse des Beziehungsnetzes können die einzelnen Schritte eines Angreifers nachvollzogen werden. In einem konkreten Vorfall kann so beispielsweise geklärt werden, ob der Hacker zunächst Zugriff auf das IT-System durch ein geratenes Passwort erlangte oder ob der erste Zugriff schon wegen des vorgefundenen Passwortspionageprogrammes gelingen konnte.

Alle Analysen zusammen sollen zu einem holistischen Vorfallsverständnis führen. Das Ergebnis der Analyse kann des Weiteren die Benennung der Angriffstechniken und die Einstufung des Vorfalls entsprechend der Vorfallsklassifikation umfassen.

2.3 Methoden der Vorfallserkennung und -analyse

Bei den Aufgaben der Vorfallserkennung und -analyse werden die CSIRTs üblicherweise von IT-Systemen unterstützt. Entsprechend gibt es eine Fülle von CSIRT-Werkzeugen. In diesem Kapitel werden die Aspekte der dahinter stehenden Methoden erläutert und klassifiziert.

In einer Umfrage ermittelte insecure.org die beliebtesten 75 Sicherheitswerkzeuge (vgl. [Fyodor 2003]). Unter diesen befinden sich auch zahlreiche Implementationen von Vorfallserkennungs- und -analysemethoden. Obwohl die Beschreibungen einen Überblick über den Software-Funktionsumfang beinhalten, können die Tools nicht immer offensichtlich differenziert werden. Aus diesem Grund werden im Folgenden mögliche Klassifikationen diskutiert.

In Abschnitt 2.2 wurden Analysen unterschieden nach Intra- und Intervorfall. Eine entsprechende Einteilung von Methoden gelingt zwar auch, scheitert aber in der weiteren Differenzierung der Intra-Vorfallsanalyse. So sind die Untersuchungsobjekte bei Protokolldatei- und Artefaktanalysen unterschiedlich, beide Untersuchungen erfordern aber eine Dateiinhaltsanalyse, die durch dieselben Methoden erledigt werden können.

In Abschnitt 1.3.1 wurde bereits eine Klassifikation von Computersicherheitsvorfällen aufgeführt. Diese unterscheidet die Vorfälle zunächst nach zufälligen und vorsätzlichen Ereignissen. Eine derart aufgestellte Unterteilung ist als Klassifikation für Methoden, deren Implementationen Vorfälle erkennen und analysieren sollen, jedoch nicht anwendbar. Das Problem besteht darin, dass durch Methoden nur bedingt erkannt werden kann, ob jemand intentional oder unabsichtlich einen Vorfall erzeugt hat. So können zum Beispiel Intrusion Detection Systeme nicht erkennen, ob ein zu großes ICMP-Echo-Request-Paket (vgl. [MS KB 154174]) mit böswilliger Absicht abgeschickt wurde. Bei Angriffen, die durch mehrere Schritte in einer bestimmten Reihenfolge erfolgen, könnten Intrusion Detection Systeme andererseits mit hoher Wahrscheinlichkeit darauf schließen, dass ein Hacker absichtlich diesen Vorfall erzeugt. Damit würde aber die Qualität der Intentionalitätsaussage von der Komplexität des Angriffs abhängen. Dementsprechend kann die Klassifikation für Vorfälle nicht auf die Methoden, die diese erkennen und analysieren sollen, sinnvoll übertragen werden.

Wenn die Methoden nicht entsprechend der Vorfallsklassifikation in Abschnitt 1.3.1 oder der Analysedifferenzierung in Abschnitt 2.2 unterteilt werden können, dann ist zu überprüfen, ob eine Einteilung entsprechend den Sicherheitsanforderungen gelingt. Folglich ist relevant, ob mit den Methoden Integritäts-, Vertraulichkeits- oder Verfügbarkeitsvorfälle erkannt und analysiert werden können. Viele der heute existierenden Softwaretools, die beispielsweise die aufgerufenen Betriebssystembefehle zum Lesen und Schreiben von Dateien beobachten (vgl. [Cogswell & Russinovich 2003: Filemon]), würden jedoch in mehrere Klassen fallen. Denn wenn als Beispiel aufgrund von Integritätsverletzungen der Konfigurationsdateien ein Web-Server nicht mehr zur Verfügung steht, würde mit der Methode sowohl die Integritäts- als auch die Verfügbarkeits-

verletzung erkannt werden können. Eine Unterteilung von Methoden und deren Implementationen nur nach Vertraulichkeit, Integrität und Verfügbarkeit reicht also nicht aus.

Der aufgestellte Vorfallsvektor $\vec{v}_k = \langle c(t_k), i(t_k), a(t_k), \vec{L} \rangle$ (vgl. Seite 12) unterscheidet Sicherheitsvorfälle nicht nur nach Vertraulichkeit, Integrität und Verfügbarkeit, sondern auch nach der IT-System-Schicht, auf der der Vorfall entstand. Wird dieser Aspekt bei der Methodenklassifikation berücksichtigt, stellt die Integritätsverletzung beim Web-Server kein Gegenbeispiel mehr dar. Somit kann in einem ersten Schritt als Grundlage für eine Klassifikation von Methoden, die eben solche Vorfälle analysieren sollen (vgl. Abb. 11), der Vorfallsvektor genutzt werden.

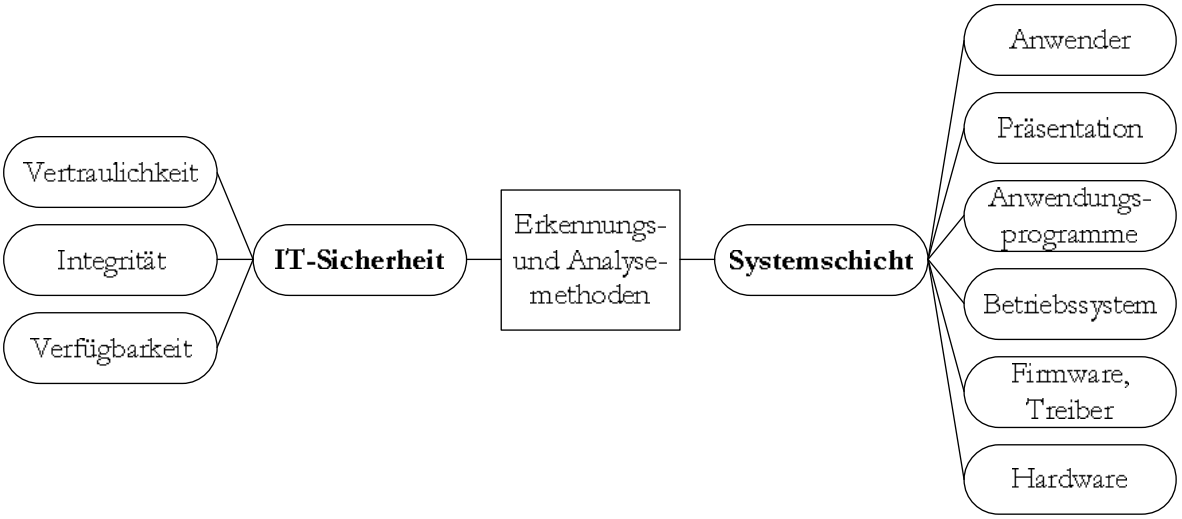


Abb. 11: erste Methodeneinteilung

Basierend auf dieser Einteilung kann eine Matrix aufgestellt werden, mit einer Achse für die Sicherheitseigenschaften Vertraulichkeit, Integrität und Verfügbarkeit und einer weiteren Achse bestehend aus den Schichten der Computerarchitektur (vgl. Abb. 12).

	Vertraulichkeit	**Integrität**	**Verfügbarkeit**
Anwender			
Präsentation			
Anwendung			WatzNew[1]
Betriebssystem	HIDS[2], Filemon[3]	Regmon[4]	
Treiber / Firmware	Ethereal[5]	Tripwire[6]	
Hardware			

Abb. 12: vorläufige Klassifikationsmatrix für Methoden zur Vorfallserkennung und –analyse mit Beispielen

[1] Tool zur Überwachung von Web-Seiten-Änderungen (vgl. [WatzNew 2003])

[2] Hostbasierte Intrusion Detection Systeme (vgl. [Mutlu et al. 2001: 3.2.2])

[3] protokolliert Lese- und Schreibzugriff auf Dateien von Prozessen (vgl. [Cogswell et al. 2003: Filemon])

[4] protokolliert Lese- und Schreibzugriffe auf die Registry (vgl. [Cogswell et al. 2003: Regmon])

[5] zeichnet alle Pakete auf, die eine Netzwerkkarte eines IT-Systems empfängt oder sendet und ermöglicht die Darstellung der Pakete auf verschiedenen Kommunikationsschichten (vgl. [Ethereal 2003])

[6] vergleicht gespeicherte Informationen über Dateien und Verzeichnisse mit den aktuellen (vgl. [Tripwire 1992])

Obwohl somit eine wesentliche Unterscheidung von Methoden zur Vorfallserkennung und -analyse gefunden wurde, gibt es noch weitere relevante Unterschiede in den Methoden. Um diese in der Klassifikation berücksichtigen zu können, werden im Folgenden zunächst die Eigenschaften und Ansätze möglicher Methoden betrachtet.

Zwei Eigenschaften von Methoden zur Vorfallserkennung und -analyse sind abhängig von Ort und Zeit. Letztere wird wesentlich durch die Sicht auf Vorfälle bestimmt.

2.3.1 komparativ statische, quasi-dynamische und dynamische Analysen

Die Vorfallsdefinition (vgl. S. 10) und die Vorfallsrelation $v(t_i)$ (vgl. S. 11) besagen: ein Vorfall wird dadurch bestimmt, dass der Sicherheitszustand $sec(t_i)$ die Sicherheitsanforderungen sec_{req} unterschreitet ($sec(t_i) <_{SEC} sec_{req}$). Diese Definition legt also den Fokus auf die Repräsentation eines Vorfalls in den Sicherheitszuständen.

In den Definitionen von Howard und Longstaff (vgl. Abschnitt 1.3) bezüglich Ereignissen und Angriffen sind Aktionen als Vorfallsaspekt enthalten. Als Bestandteil von Angriffen definieren sie Ereignisse als eine Reihe von zielgerichteten Aktionen. Mit dieser Sicht wird es möglich, Vorfälle als Prozesse aufzufassen, die eine Unterschreitung der Sicherheitsanforderungen bewirken. Demnach überführen Vorfälle ein IT-System von einem Zustand S_{pre}, in dem das IT-System den Sicherheitsanforderungen entspricht, in einen Zustand S_{post}, in dem das IT-System den Anforderungen nicht mehr genügt ($v(t_i) = \textit{wahr}$). Der Vorfall kann entsprechend in einem Petri-Netz als Transition (im Folgenden: *Vorfallstransition*) dargestellt werden (vgl. Abb. 13).

Abb. 13: Vorfallsprozess $S_{pre} \xrightarrow{\text{Vorfall}} S_{post}$

Beispielsweise wäre ein Web-Server im Zustand S_{pre} voll funktionsfähig und würde den Sicherheitsanforderungen entsprechen. Wenn es einem Hacker gelingt, Web-Seiten auf einem Server auszutauschen, dann werden die Sicherheitsanforderungen verletzt und das IT-System befindet sich im Zustand S_{post}. Als Vorfallstransition kann also der Austausch der Web-Dateien ausgemacht werden.

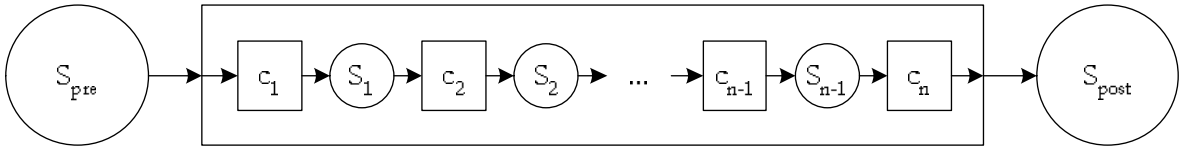


Abb. 14: verfeinerter Vorfallsprozess $S_{pre} \xrightarrow{c_1 \cdot c_2 \cdot \ldots \cdot c_{n-1} \cdot c_n} S_{post}$

Die Vorfallstransition kann, wie in Abb. 14 dargestellt, auch verfeinert werden. In dem vorangegangenen Beispiel muss ein Hacker normalerweise mehrere Schritte vollziehen, mehrere Befehle nutzen, um die Web-Dateien auszutauschen. Also besteht der Vorfall aus einer Reihe von

Transitionen und kann prozessalgebraisch als Verkettung von n vorfallsrelevanten Transitionen c_i ($\forall i \in \{1,\ldots,n\}$) aufgefasst werden:

Vorfallsverfeinerung: $\text{Vorfall} = c_1 \cdot c_2 \cdot \ldots \cdot c_{n-1} \cdot c_n$

Zwischen den Transitionen c_j und c_{j+1} ist das IT-System im Zustand S_j ($\forall j \in \{1,\ldots,n-1\}$).

Eine derartige Prozesssicht auf Vorfälle bestimmt eine wichtige Eigenschaft von Vorfallserkennungs- und -analysemethoden. Abhängig von der jeweiligen Sicht und dem Fokus auf Zustände oder auf Transitionen können Methoden zur Vorfallserkennung und -analyse, wie folgt, unterschieden werden.

Komparativ statische Analysen

Komparativ statische Verfahren vergleichen die Zustände S_{pre} und S_{post}, um Vorfälle zu erkennen und zu analysieren. Sie untersuchen also die Zustandsänderungen aufgrund der Vorfallstransition. Dies erfordert insbesondere, dass es eine Repräsentation des Zustandes S_{pre} gibt, die als Vergleichsreferenz genutzt werden kann. Die Software *Tripwire* (vgl. [Tripwire 1992]) speichert dazu in einer Datenbank die für die Software relevanten Zustandsparameter und vergleicht spätere Zustände mit den Datenbankeinträgen. Auch System-Backups können als Referenz dienen.

Problematisch sind bei den komparativ statischen Analysen die Wahl der Zustände und der Zeitraum zwischen diesen. Bei der Wahl der zu vergleichenden Zustände muss darauf geachtet werden, dass diese auch relevant sind. Wird zum Beispiel bei dem Einsatz von Tripwire die Referenzdatenbank (Zustand S_{pre}) erst nach einem Integritätsvorfall angelegt, dann wird Tripwire die Sicherheitsverletzung nicht entdecken. Wenn als S_{post} ein Zustand gewählt wird, der zeitlich hinter dem Wiedereinspielen eines Backups liegt, dann ist auch dieser ungeeignet, da der Vorfall dann nicht mehr nachvollzogen werden kann.

In der Zeit zwischen S_{pre} und S_{post} werden häufig auch noch Befehle ausgeführt, die mit dem Vorfall selber in keinem direkten Zusammenhang stehen. Beispielsweise ist ein Dateitransfer-Server, der jeden FTP-Befehl protokolliert, durch einen verteilten DoS-Angriff stark ausgelastet. Lediglich wenige reguläre FTP-Anfragen können von ihm bearbeitet werden. Diese Änderungen der Protokolldatei und damit des System-Zustandes sind von solchen zu unterscheiden, die wegen des DoS-Angriffes eintreten. Bei der komparativ statischen Analyse muss entsprechend eine Differenzierung zwischen legitimen und illegitimen Zustandsänderungen erfolgen. Die Legitimität der Änderungen hängt vom jeweiligen Anwendungskontext ab.

Post Mortem Analysen

Auf derselben Idee wie die komparativ statischen Verfahren basieren auch Post Mortem Analysen, bei denen nur der Zustand S_{post} bekannt ist. Anstelle des Vergleiches mit dem Zustand S_{pre} muss die Post Mortem Analyse durch einen Experten oder ein Expertensystem erfolgen. Dieser bzw. dieses muss die Abläufe im System detailliert kennen, so dass die eigentliche Referenz das Verständnis über das IT-System ist. Dementsprechend genügt bei der Post Mortem Analyse der Zustand S_{post}. Der Post Mortem Ansatz wird zum Beispiel von Microsoft bei der *Online Crash Analysis* (OCA) genutzt (vgl. [MS OCA 2003]).

Quasi-dynamische Analyse

Als quasi-dynamische Verfahren werden in dieser Arbeit Methoden bezeichnet, die den komparativ statischen Ansatz nicht nur auf die Systemzustände S_{pre} und S_{post}, sondern auch auf alle Zwischenzustände $S_1 \ldots, S_{n-1}$ anwenden. Sie stellen damit neben den Post Mortem Analysen einen weiteren Spezialfall des komparativ statischen Ansatzes dar. Durch quasi-dynamische Analysen werden die Änderungen, die bei der komparativ statischen Analyse nur insgesamt festgestellt werden können, in eine zeitliche Reihenfolge gebracht. Damit sind die Vorgänge, die zu einem Vorfall geführt haben, transparenter. Denn mit der quasi-dynamischen Analyse kann festgestellt werden, ob $Vorfall_1 = a \cdot b \cdot c \cdot d$ oder $Vorfall_2 = d \cdot c \cdot b \cdot a$ sich ereignet haben, wenn zum Beispiel $S_{pre} \xrightarrow{Vorfall_1} S_{post}$ und $S_{pre} \xrightarrow{Vorfall_2} S_{post}$ möglich wären.

Zudem ermöglicht dieser Analyseansatz die Feststellung von Änderungen, die sich ansonsten gegenseitig aufheben. Während die komparativ statische Analyse von S_{pre} und S_{post} keine oder nur weniger Veränderungen feststellen kann, wäre dies mit quasi-dynamischen Methoden erkenn- und analysierbar. Wenn der $Vorfall = a \cdot b \cdot c \cdot \overline{a}$ mit $\overline{a}$ als Komplement zu a entspricht, dann können a und $\overline{a}$ nur (quasi-)dynamisch festgestellt werden. Dies kann zum Bespiel der Fall sein, wenn ein Hacker als letzte Aktion seines Angriffs die eingesetzte Spionage-Software wieder deinstalliert.

Der quasi-dynamische Ansatz erfordert im Vergleich zu dem allgemeinen komparativ statischen, dass sehr viele Zustände zum Vergleich zur Verfügung stehen. Je nach Definition des IT-System-Zustandes bzw. dessen Ausschnittes, der verglichen werden soll, müssen sehr viele Daten gesichert werden. Derartige Zustandssicherungen sind aufwendig und zeitintensiv. Deshalb werden sie in der Praxis nur selten eingesetzt. Lediglich beim Reverse-Engineering von neuer Malware (vgl. [Janz 2000]) und gelegentlich auch bei der Programmierung von Treibern (vgl. [Knepper 2003]) wird der quasi-dynamische Ansatz angewendet.

Dynamische Analyse

Im Gegensatz zu den bisherigen Verfahren versucht die dynamische Analyse nicht aufgrund der IT-Systemzustände auf den Vorfall zu schließen. Beim dynamischen Ansatz wird davon ausgegangen, dass die Methode einen Bereich fokussiert, der detailliert protokolliert werden kann. Zum Beispiel bieten sich Funktionsaufrufe einer Anwendung mit Parametern oder Betriebssystembefehle an, um diese dynamisch, d. h. beim Aufruf, zu protokollieren und ggf. auch zu analysieren. Derartige Methoden werden häufig Monitore genannt. Sie beobachten die Transitionen c_i der Vorfallverfeinerung. Die im Abschnitt über die quasi-dynamischen Analysen erkannten Vorteile (Unterscheidung von $Vorfall_1$ und $Vorfall_2$, Erkennung von Verschleierungsversuchen) gelten also auch für dynamische Ansätze. Dynamische Vorfallsanalysen ermöglichen also das detaillierte Nachvollziehen der Einzelschritte, die zu einem Vorfall geführt haben, wobei sie sich auf die tatsächlich ausgeführten Befehle konzentrieren.

Erschwert wird die Analyse bei fehlerhaften Implementationen und bei fehlender Zuordnung der protokollierten Aufrufe zu den aufrufenden Prozessen. Wenn die protokollierten Funktionsaufrufe aufgrund eines Fehlers nicht die spezifizierten Auswirkungen haben, dann muss dies bei der Analyse bekannt sein bzw. durch den Einsatz anderer Analysemethoden ermittelt werden können. Bei der Ausnutzung der „URL Decoding“-Schwäche werden zum Beispiel speziell

parametrisierte HTTP-Get-Befehle genutzt (vgl. [ISS 2000645]), um Zugriff auf das IT-System zu erhalten. Wird neben dem Protokoll für Web-Anfragen auch das Starten und Beenden von Prozessen dokumentiert (vgl. [Cogswell et al. 2003: PMon]), können Zusammenhänge zwischen den speziellen HTTP-Get-Befehlen und anderen Ereignissen auf dem System festgestellt werden. Dies ist insbesondere dann möglich, wenn nicht nur erkannt wird, dass ein beobachteter Befehl aufgerufen wurde, sondern auch welcher Prozess dies getan hat. Erst dadurch kann eine gesicherte Zuordnung der Aufrufe durchgeführt und damit ein Sicherheitsvorfall nachvollzogen werden.

Im Vergleich zum komparativ statischen Ansatz müssen dynamische Methoden ständig aktiv sein, da nur zur Zeit des Vorfalls die Daten für den dynamischen Ansatz anfallen. Eine Extraktion dieser Daten nach einem Vorfall ist nicht möglich. Diese Bedingung erfordert die permanente Bereitstellung von Ressourcen. Im Gegenzug besteht beim Einsatz von dynamischen wie auch quasi-dynamischen Verfahren die Möglichkeit, den Vorfallsprozess in Realzeit bzw. zeitnah zu erkennen und eventuell auch verhindern zu können.

Entscheidend für den erfolgreichen Einsatz von dynamischen Methoden ist die korrekte Wahl der zu beobachtenden Befehle. Bei der lokalen Verfügbarkeitsüberprüfung eines Servers reicht es zum Beispiel nicht aus, den Dienst nur auf der Anwendungsschicht zu überwachen. Es muss auch gewährleistet sein, dass alle Schichten, auf die der Dienst aufbaut, verfügbar sind. Würde der Treiber für die Netzwerkkarte ausfallen, dann wäre der Server-Dienst lokal noch verfügbar, jedoch nicht mehr über ein Netzwerk.

Zu den dynamischen Verfahren zählen zum Beispiel die Sysinternals Software *Filemon* (vgl. [Cogswell et al.: Filemon]), die alle Lese- und Schreibzugriffe auf die Festplatten erkennt und protokolliert, und auch netzbasierte Intrusion Detection Systeme (NIDS), die TCP/IP-Pakete dynamisch analysieren (vgl. [Mutlu et al. 2001: 3.4.1]).

Mit der Unterscheidung der Ansätze zwischen komparativ statisch und dynamisch kann die vorläufige Methodeneinteilung (s. Abb. 12) um den Prozessaspekt erweitert werden (s. Abb. 15).

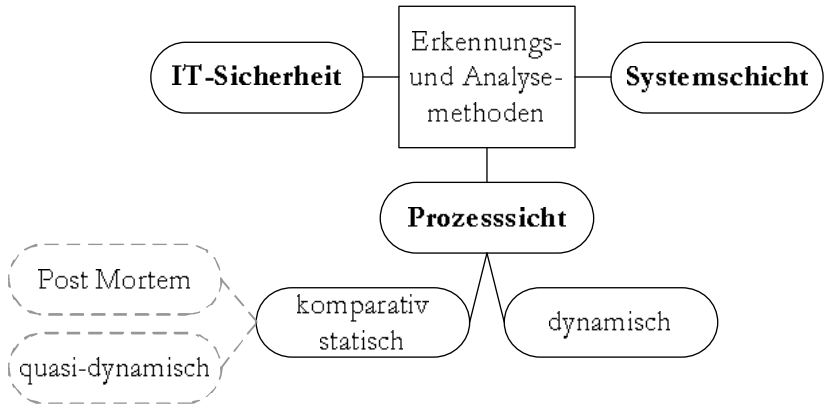


Abb. 15: erweiterte Methodeneinteilung

2.3.2 Lokalität der Analyse-Ressourcen

In den vorangegangenen Abschnitten wurden basierend auf der unterschiedlichen Sichtweise auf den Vorfallsprozess verschiedene Erkennungs- und Analyseansätze diskutiert. Neben dieser Einteilung sind die Methoden auch nach dem Einsatzort differenzierbar.

Die Web-Seiten-Überwachungssoftware *WatzNew* von A.I. Studio (vgl. [WatzNew 2003]) ist sowohl auf dem Web-Server selber als auch an beliebiger Stelle im angeschlossenen Netzwerk zur Verfügbarkeitsüberwachung einsetzbar. Trotz der freien Wahl ist der Einsatzort aber nicht unwichtig: als lokaler Monitor kann WatzNew die Verfügbarkeit bestätigen, obwohl die Verbindung vom Server zum Netzwerk gekappt ist; als netzbasierter Monitor kann WatzNew mit der gleichen Aufgabe ebenfalls die Verfügbarkeit des nicht verbundenen Web-Servers bestätigen, wenn ein zwischengeschalteter Web-Proxy die Web-Seiten aus seinem Cache an den Monitor liefert.

Neben der im Beispiel erwähnten Problematik ist der Einsatzort weiterhin relevant, weshalb im Folgenden die Möglichkeiten aufgezeigt werden.

hostbasierter Einsatz

Unter hostbasierten Methoden werden solche verstanden, deren Implementation losgelöst von Anforderungen an ein Netzwerk eingesetzt werden (vgl. Abb. 16). Solche Beobachter werden lokal installiert und führen die Untersuchungen zur Erkennung und Analyse nur lokal durch. Dabei ist die Einschränkung auf den Host unabhängig von den Prozessaspekten. Es existieren sowohl hostbasierte, dynamische Monitore, wie z.B. HIDS (vgl. [Mutlu et al. 2001: 3.2.2]), als auch hostbasierte, komparativ statische Werkzeuge wie Tripwire (vgl. [Tripwire 1992]).

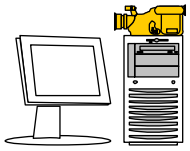


Abb. 16: lokaler Monitor

Dedizierte, nicht vernetzte Systeme, die die Daten für die Analyse lokal etwa mittels Bandsicherungen zur Verfügung gestellt bekommen, gehören nach hier vertretener Ansicht auch in die Gruppe der hostbasierten Erkennungs- und Analysewerkzeuge.

netzbasierter Einsatz

Netzbasierte Lösungen benötigen ein Netzwerk, um Vorfälle erkennen und analysieren zu können (s. Abb. 17). NIDS beobachten zum Beispiel den Datenverkehr auf einem Netzwerksegment. Sie analysieren die Datenpakete in dem überwachten Netzwerk, um Vorfälle zu erkennen (vgl. [Mutlu et al. 2001: 3.2.1]).

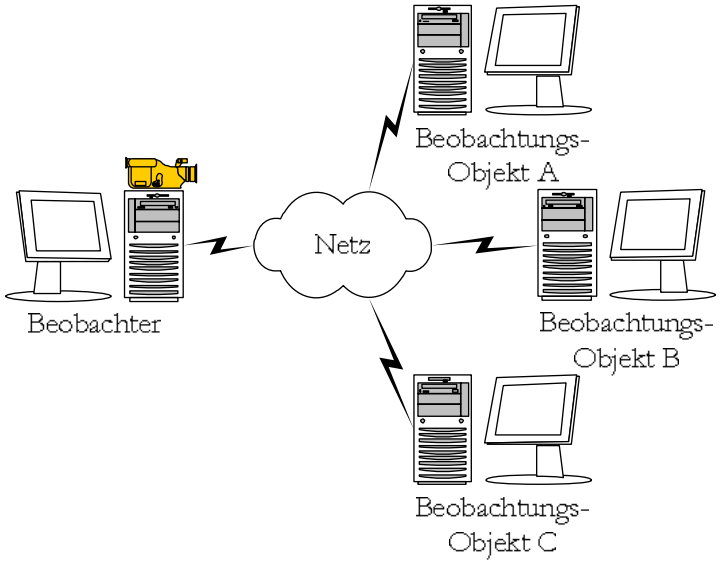


Abb. 17: netzbasierte Beobachtung

Gegenüber den netzbasierten haben die hostbasierten Methoden den Vorteil, dass sie nur durch lokale oder sich lokal auswirkende Ereignisse beeinflusst werden können. Wie das einleitende Beispiel zeigt, hat der Ausfall der Netzwerkverbindung keinen Einfluss auf das hostbasierte Werkzeug. Unter dem Aspekt der Vorfallserkennung und vor allem der Vorfallsanalyse ist der hostbasierte Einsatz problematisch, wenn ein Vorfall sich auf die Erkennungs- und Analysetechnik auswirken kann. So wird ein Vorfall nicht erkannt und kann auch nicht erkannt werden, wenn dazu eine Protokolldatei ausgewertet werden muss und der Vorfall selber verhindert, dass ein Monitor in dieser Datei Einträge vornimmt.

Des Weiteren kann bei der Erlangung der Kontrolle über ein IT-System ein Hacker auch lokale Vorfallserkennungswerkzeuge leichter fingieren, als wenn netzbasiert erkannt und analysiert wird, da der Hacker bereits die Kontrolle über das System hat und nicht erst noch erwerben oder anderweitig täuschen muss. Auch netzbasierte Methoden weisen entsprechende Grenzen auf. Die netzbasierte Integritätsüberwachung kann zum Beispiel gar nicht oder nur eingeschränkt durchgeführt werden, wenn das Beobachtungssystem Opfer eines DoS-Angriffes ist. Werden IT-Systeme nur netzbasiert überwacht, dann braucht ein Hacker auch nur dieses System zu manipulieren, um im beobachteten Netzsegment unentdeckt agieren zu können.

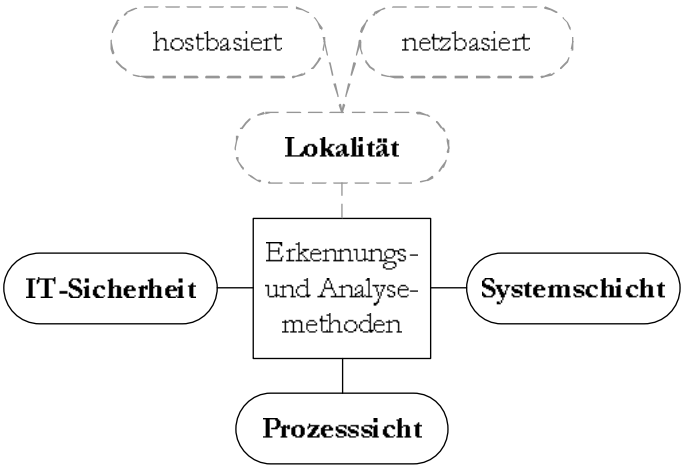


Abb. 18: Aspekte von Methoden zur Vorfallserkennung und –analyse

Die Beispiele und Überlegungen zeigen, dass die Wahl des Einsatzortes die Vorfallserkennungs- und -analysemöglichkeiten unterschiedlich eingrenzt und somit bei der Konzeption von Sicherheitsüberwachungen berücksichtigt werden müssen. Die Lokalität des Werkzeuges ist also ein Aspekt bei den Vorfallserkennungs- und -analysemethoden (s. Abb. 18).

Wie das einführende Beispiel jedoch zeigt, kann die Lokalität keinen weiteren zu berücksichtigenden Beitrag in einer Klassifikation leisten, da Methoden existieren, die bezüglich der Lokalität in beide Gruppen fallen.

2.4 Klassifikation von Erkennungs- und Analysemethoden

Aufgrund der im vorangegangenen Abschnitt beschriebenen Eigenschaften (s. Abb. 19) können Methoden klassifiziert werden. Allerdings müssen die Eigenschaften von einander unabhängig sein, um in einer Klassifikation berücksichtigt werden zu können (vgl. [Howard et al. 1982: 3]).

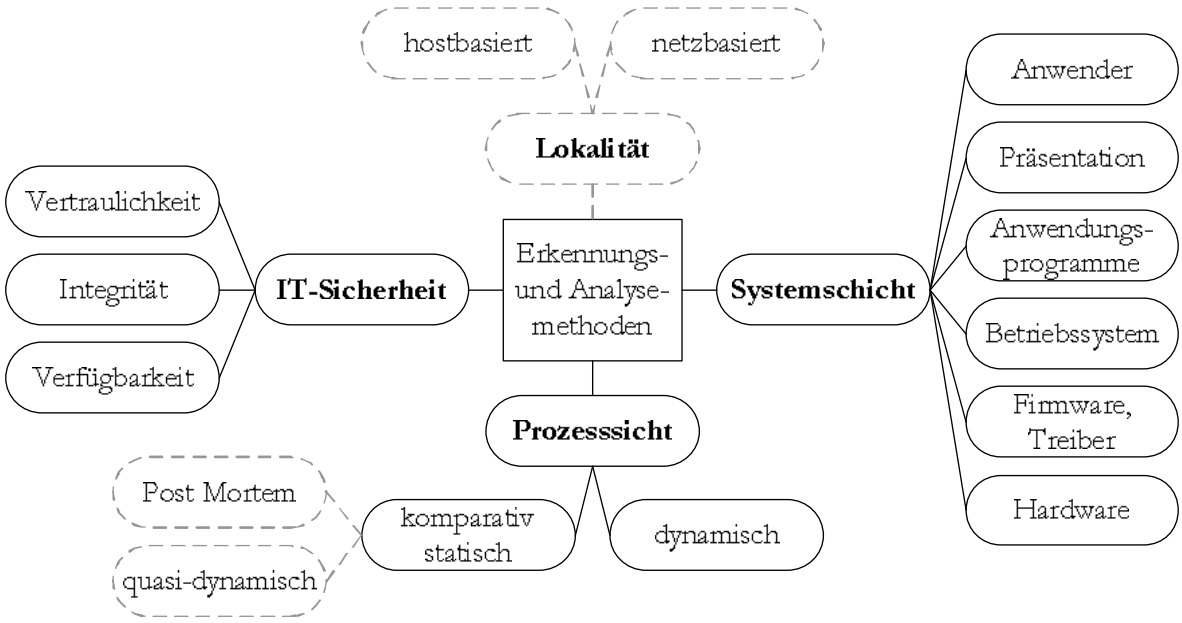


Abb. 19: Eigenschaften von Methoden zur Vorfallserkennung und –analyse

In Abschnitt 2.3.2 wurde bereits diskutiert, dass die Lokalität sich nicht als Aspekt für eine Klassifizierung eignet, da Methoden existieren können und auch existieren, die sowohl host- als auch netzbasiert genutzt werden. Dies ist nicht der Fall bei der Unterscheidung nach komparativ statisch und dynamisch. Bedingung bei der Prozesssicht ist allerdings, dass nicht weiter nach quasi-dynamischen und Post Mortem Analysen differenziert wird, da diese selbst bei Vereinigung eine echte Teilmenge der komparativ statischen Analysen darstellen. Neben dem Gesichtspunkt, wie analysiert wird, ist eine Differenzierung der Methoden nach den erkenn- und analysierbaren Vorfällen wichtig. Diese erfolgt anhand des Vorfallsvektors (vgl. S.12).

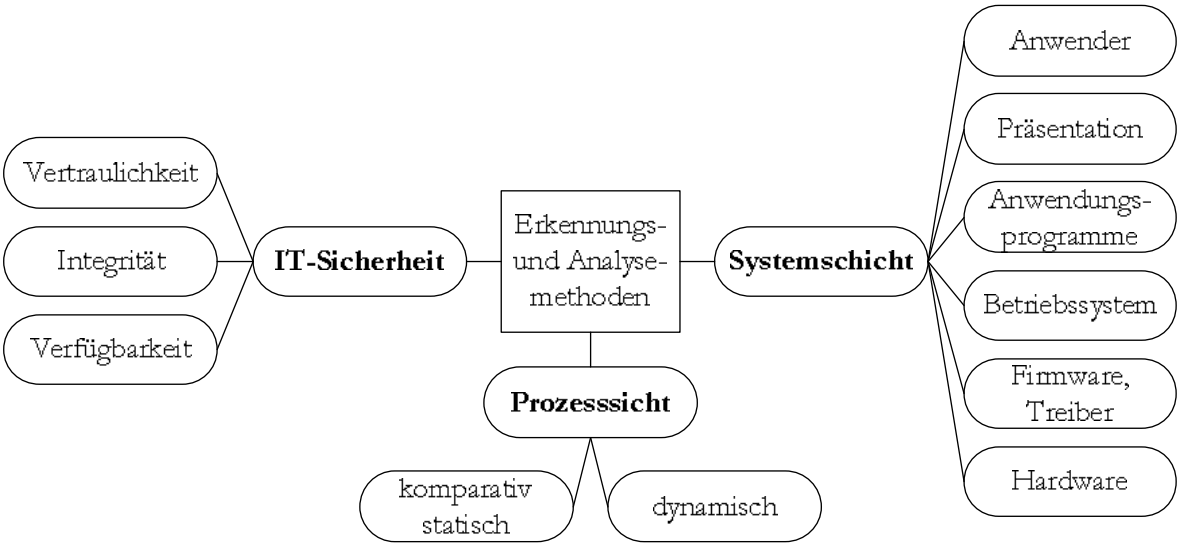


Abb. 20: Klassifikation von Methoden zur Vorfallserkennung und -analyse

Aufgrund der vorangegangenen Diskussion unterscheidet eine Klassifikation für Methoden zur Vorfallserkennung und -analyse also nach den IT-Sicherheitseigenschaften, der Systemschicht und der Prozesssicht. Die entsprechende Klassifikation ist in Abb. 20 dargestellt.

3 Implementation eines komparativ statischen Vergleichswerkzeugs

In diesem Kapitel wird eine Methode des komparativ statischen Systemvergleichs anhand von Festplattenzuständen im Detail erläutert. Dazu entstand im Rahmen der Diplomarbeit die Software *CompareSys* für das IRT.

Für den Entwicklungsprozess von CompareSys wurde das Wasserfallmodell (vgl. u.a. [STE 2000: F. 13.3b]) verwendet. Entsprechend wird mit der Darstellung des Ist-Szenarios gefolgt von dem Wunschszenario begonnen. Die zugrunde liegende Systemgestaltung und der Softwareentwurf werden anschließend umrissen, wobei die wichtigsten implementierten Algorithmen detaillierter vorgestellt werden. Die Darstellungen orientieren sich an der Unified Modeling Language (UML, s. u.a. [TogetherSoft 2002]).

3.1 Derzeitiges Vorgehen des IRT

Das Incident Response Team der Universität Hamburg analysiert in einem Sicherheitslabor bekannte und neue Computervorfälle. Dabei werden die Ereignisse, die zu einem Vorfall geführt haben, reproduziert, protokolliert und ausgewertet. Das folgende Ablaufdiagramm (s. Abb. 21) beschreibt das Vorgehen des IRT genauer:

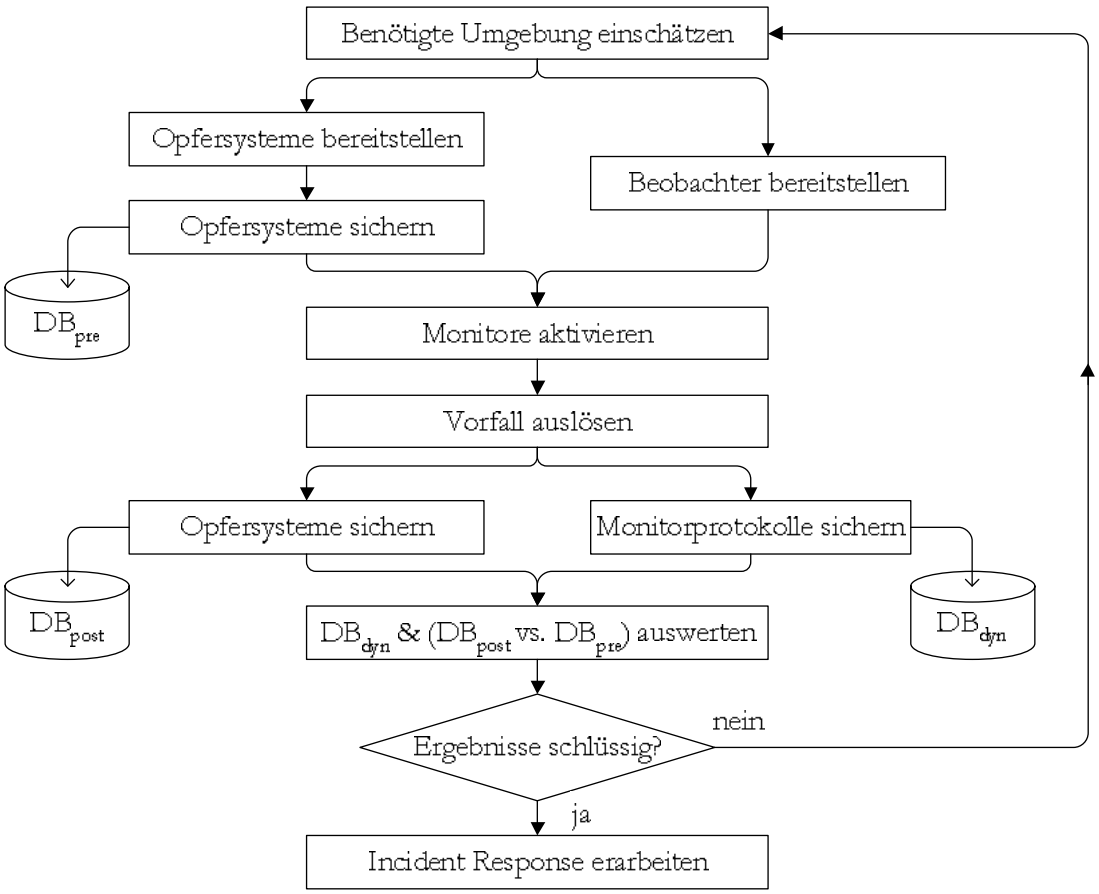


Abb. 21: Workflow im IRT

Jede Untersuchung von Vorfällen durch das IRT erfolgt in einer abgeschotteten Umgebung, in der der zu untersuchende Vorfall reproduziert werden kann. Dazu werden Opfersysteme (wie

Web- und Fileserver etc.) je nach ersten Einschätzung des Vorfalls installiert und konfiguriert. Zusätzlich werden auf den Opfersystemen und auf dedizierten Beobachtern spezielle Monitore eingerichtet. Die Monitore sollen zum Beispiel die laufenden Prozesse (vgl. [Cogswell et. al 2003: PMon]) sowie Dateisystemzugriffe (vgl. [Cogswell et. al 2003: Filemon]) und TCP/IP-Pakete (vgl. [Ethereal 2003]) für eine dynamische Analyse protokollieren.

Ist die Umgebung hergestellt, werden die Daten der Opfersysteme als Zustand S_{pre} gesichert (z.B. Kopie der Festplatte), um komparativ statische Analysen zu ermöglichen. Sind die Vorbereitungen abgeschlossen und die Monitore aktiviert, wird der Vorfall mit den Opfersystemen durchgeführt. Im Anschluss an die Durchführung des Vorfalls werden alle angefallenen Daten gesichert. Das sind zum einen die Aufzeichnungen der Monitore für die dynamische Analyse und zum anderen wiederum die Daten der Opfersysteme (S_{post}), die in der anschließenden komparativ statischen Auswertung mit den Daten der Opfersysteme vor dem Vorfall verglichen werden.

Ergeben die Daten ein schlüssiges Bild über den Vorfall, erarbeitet das IRT aufgrund der Analyseergebnisse Maßnahmen und Konzepte als Response auf den Vorfall. Fehlen offensichtlich Daten, weil zum Beispiel festgestellt wird, dass die zu untersuchende Software versucht, eine HTTP-Verbindung aufzubauen, obwohl kein Web-Server in der Testumgebung bereitgestellt wurde, dann muss die Testumgebung überdacht und der Versuch wiederholt werden.

Bei den Versuchen im Wintersemester 2002 / 2003 wurden im IRT die Würmer W32/Nimda und Linux/Slapper untersucht. Während Nimda nur dynamisch mittels Netzsniffer und diversen Systemmonitoren analysiert wurde, nutzte die Fachgruppe für Linux beim Slapper auch eine komparativ statische Methode (vgl. [IRT 2003b: F.5]). Die genutzte Implementation hieß *diff* und ist in den meisten Linux-Distributionen enthalten.

Die Fachgruppe für Windows hatte auf eine komparativ statische Analyse verzichtet, da vermutet wurde, dass alle Dateizugriffe durch die eingesetzten Systemmonitore festgestellt werden. Der in Anlage B beigefügte Auszug aus der Reportdatei des Dateisystemmonitors (vgl. [Cogswell et al. 2003: Filemon) zeigt jedoch, dass der Monitor zwar das Lesen und Schreiben von Dateien protokolliert hat, aber eine holistische Analyse aufgrund der Aufzeichnungen nur mit sehr großem Aufwand gemacht werden kann: Die Menge der Daten (über 8000 Datensätze für die Dateizugriffe von Nimda in den ersten zwei Sekunden nach Ausführung) erschwert zunächst das Nachvollziehen der Änderungen in den einzelnen Dateien und des gesamten Dateisystems. Des Weiteren werden nur die Zugriffe auf das Dateisystem beobachtet; ob überhaupt und was in den Dateien geändert wurde, protokollierten die genutzten Monitore nicht.

3.2 Anforderungsermittlung

Die beschriebene Situation war für das IRT und insbesondere für die Fachgruppe für Windows nicht zufrieden stellend, da die Masse der Daten sehr viel Analyseaufwand verursachte und den IRT-Mitgliedern kein vollständiges Bild geben konnte. Um dem IRT eine ganzheitliche Analyse von Sicherheitsvorfällen effizient zu ermöglichen, wurde überlegt, eine Unterstützungssoftware für den komparativ statischen Systemvergleich zu erstellen. Neben den vielen existierenden Werkzeugen für die dynamische Analyse soll am Ende der Entwicklung eine Software zur Verfügung stehen, mit der festgestellt werden kann, welche Dateien während eines Vorfalls

erzeugt, geändert oder gelöscht wurden. Des Weiteren sollen Änderungen in Dateien auch im Detail erforscht werden können, indem die Dateiinhalte einander gegenüber gestellt werden.

Bei diesen Analysen sollen auch die besonderen Bedingungen, die im IRT vorliegen, in der Software Berücksichtigung finden. Beispielsweise sind die in der abgesicherten Testumgebung eingesetzten IT-System keine Produktivsysteme und unterliegen deshalb keinen Änderungen durch die „normale“ Nutzung durch Anwender.

Die Aufgaben, die für ein Mitglied des IRT bezüglich eines komparativ statischen Systemvergleichs bestehen, setzen sich aus der Sicherung der Festplattenzustände vor und nach einem Computervorfall, dem Vergleichen dieser Zustände sowie dem Vergleich der Inhalte geänderter Dateien zusammen (s. Abb. 22).

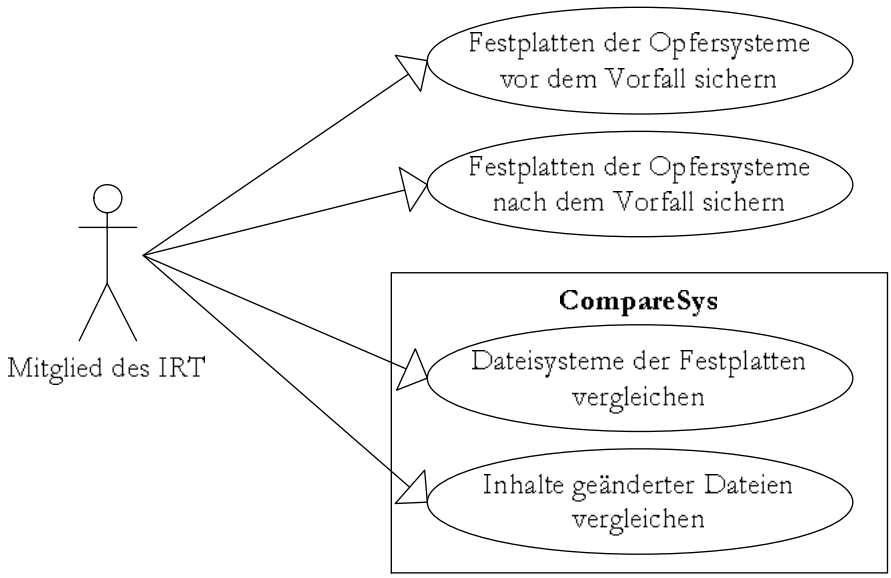


Abb. 22: Use Case für komparativ statische Festplattenanalysen

Die in Abb. 22 eingerahmten Aufgaben sollen durch eine Software realisiert werden, wobei die beiden Aufgaben wie folgt gemäß UML Use Cases definiert sind:

Festplatten-Zustände vergleichen
Kurzbeschreibung Die Daten der vorliegenden Festplattensicherungen von vor und nach dem Vorfall werden verglichen.
Beteiligte Akteure Mitglied des IRT
Vorbedingung „Festplatten vor dem Vorfall sichern“ und „Festplatten nach dem Vorfall sichern“
Nachbedingung „Inhalte geänderter Dateien vergleichen“
Vorgehensweise Die Sicherungen der Festplatten liegen vor. Die Dateien der Festplatten werden dahingehend überprüft, ob sie auf der jeweils anderen Festplatte vorhanden sind (Existenzkontrolle). Sofern die Dateien auch dort existieren, wird überprüft, ob die Dateigröße, das Erstellungsdatum und das Änderungsdatum sowie die Dateiattribute übereinstimmen. Des Weiteren folgt eine Übereinstimmungsprüfung der Dateiinhalte. Das Ergebnis dieser Überprüfungen ist eine Liste der festgestellten Unterschiede.

Inhalte geänderter Dateien vergleichen
Kurzbeschreibung Die inhaltlichen Änderungen von Dateien werden ermittelt. **Beteiligte Akteure** Mitglied des IRT **Vorbedingung** „Festplatten-Zustände vergleichen“ **Vorgehensweise** Die Ergebnisse der vorangehenden Aufgabe werden genutzt, um die geänderte Datei weiter zu analysieren. Dazu wird der Inhalt der ursprünglichen Datei mit dem Inhalt der geänderten Datei verglichen, um zum Beispiel bei ASCII-Dateien feststellen zu können, welche Zeilen und Befehle eingeschoben bzw. entfernt wurden. Das Einbinden von Modulen für entsprechende Analysen anderer Dateitypen soll ermöglicht werden.

3.3 Annahmen und Randbedingungen

Bevor nun bereits existierende Software darauf untersucht wird, ob sie die gestellten Aufgaben bereits löst oder eine neue Software geschrieben werden sollte, werden zunächst wichtige Aspekte beschrieben, die im Weiteren Berücksichtigung finden.

3.3.1 Repräsentanz des Systemzustandes in den Dateien

Der vom IRT gewählte Ansatz berücksichtigt derzeit für die komparativ statische Analyse nur die Festplattenzustände. Die Analyse basiert also nur auf den Dateien und Verzeichnissen, wie sie vor und nach dem Vorfall existieren. Dahinter steckt der Gedanke, dass der gesamte Systemzustand sich in dem Zustand der Festplatten widerspiegelt.

Dies ist allerdings nur bedingt richtig. Wie die jüngste Vergangenheit gezeigt hat, konnte der W32/Sapphire, der nur im Arbeitsspeicher resident war, trotzdem die Verfügbarkeit von Microsoft SQL-Servern herabsetzen (vgl. [Hypponen et al. 2003]). Allerdings würden Überwachungsprogramme, die den lokalen Befall durch einen derartigen Wurm aufzeichnen, Änderungen in den Log-Dateien vornehmen. In einem solchen Fall würde auch eine Analyse der Festplattenzustände Aufschluss geben.

Die Grenzen des Beobachtungsraumes dürfen dementsprechend bei dieser Art der komparativ statischen Analysen nicht aus den Augen verloren werden (s.a. Kapitel 5).

3.3.2 Kenntnisse der Anwender

Neben Kenntnissen darüber, in welchen Bereichen komparativ statische Analysen von Festplattenzuständen auf ihre Grenzen stoßen, sind auch weitere Kenntnisse bei der Untersuchung von Vorfällen notwendig. Aufgrund der Vielfalt der Hardware, Betriebssysteme und Anwendungen gibt es eine große Fülle unterschiedlicher IT-Systeme. Um erkennen zu können, welche Änderungen des Systemzustandes sich wie im Zustand der Dateien und Verzeichnisse widerspiegeln, müssen fundierte Kenntnisse über die Zusammenhänge der betroffenen Komponenten vorliegen. Zum Beispiel wird ein Fachmann für Linux bei der Analyse eines IT-Systems mit

Microsoft Windows vielleicht übersehen, dass die Konfigurationsdaten diverser Programme nicht in eigenen Dateien liegen, sondern in der zentralen Registry abgelegt sind.

Wenn Vorfälle nicht in einem Sicherheitslabor an Testsystemen vorgenommen werden, sondern an operativen Systemen, dann muss bei den Untersuchungen der übliche Gebrauch des zu untersuchenden Systems berücksichtigt werden. Wird das betroffene System beispielsweise dafür genutzt, um neue Gerätetreiber zu testen, dann muss gesondert überprüft werden, ob die Veränderung eines Treibers die Ursache des Vorfalls war oder ob der Treiber während eines Vorfalls von einer unberechtigten oder einer berechtigten Person ausgetauscht wurde.

Zu den notwendigen Kenntnissen gehört auch ein Verständnis, wie Dateien und Dateiinhalte geändert werden können.

3.3.3 Dateiänderungen

In der Beschreibung der Aufgabe „Festplatten-Zustände vergleichen“ wird erwähnt, dass Dateien nicht nur aufgrund ihres Inhaltes verglichen, sondern auch andere Eigenschaften herangezogen werden sollen. Welche Eigenschaften existieren und verändert werden können, geht aus der Spezifikation der verschiedenen Dateisysteme hervor. Im Folgenden werden anhand der Spezifikationen für FAT12, FAT16 und FAT32 (vgl. [MS FAT 2000]) sowie einer weiterführenden Beschreibung (vgl. [Kozierok 2001b]) die Dateieigenschaften erläutert.[7]

Die FAT-Dateisysteme haben ihren Namen aufgrund einer wesentlichen Komponente des Dateisystems, der Dateibelegungstabelle (engl. *File Allocation Table*), erhalten. FAT12 wurde von Microsoft mit der ersten Version von MS DOS ausgeliefert, FAT16 mit MS DOS 2, und seit Microsoft Windows 95 gibt es FAT32. Obwohl die FAT-Dateisysteme vieles gemeinsam haben, besteht der wesentliche Unterschied in der Anzahl der von den Dateisystemen verwaltbaren Clustern. Cluster sind im Prinzip Segmente des Speicherplatzes, die über Nummern adressiert werden können und in denen Daten gespeichert sind.

Der Aufbau von FAT-Dateisystemen (hier: FAT16) unterteilt sich in vier Bereiche:

1) *BIOS Parameter Block*
   Der erste Eintrag auf einem FAT-Datenträger stellt der BIOS Parameter Block (BPB) (auch Boot Sektor genannt) dar, in dem grundlegende Informationen für diese Festplatte gespeichert sind, wie zum Beispiel dem Speicherort des Boot Codes, der Sektorengröße, der Clustergröße usw.

2) *File Allocation Table*
   In den Dateibelegungstabelle (engl. *File Allocation Table*) werden Ketten von Clustern dadurch hinterlegt, dass hinter der Adresse eines jeden Clusters ein Verweis auf den nächsten Cluster bzw. auf das Ende der Cluster-Kette (engl. *chained clusters*) gespeichert ist.

[7] Neben den FAT-Dateisystemen existieren noch weitere, wie z.B. HPFS (vgl.[Kozierok 2001c]) und NTFS (vgl. [Kozierok 2001d]). Im Gegensatz zu diesen kann FAT16 laut [Kozierok 2001a] von allen bekannteren Betriebssystemen genutzt werden. NTFS verfügt neben weiteren auch über die beschriebenen FAT-Dateieigenschaften. Aus diesen Gründen werden in der vorliegenden Arbeit die FAT-Dateisysteme als Beispiel fokussiert.

Werden die Cluster-Inhalte in der Reihenfolge dieser einfach verlinkten Ketten zusammengefügt, ist die auf dem Datenträger verteilt gespeicherte Datei rekonstruiert worden.

3) *Directory Structure*
Verzeichnisse und Dateien werden in der Verzeichnisstruktur (engl. *Directory Structure*) der FAT-Dateisysteme identisch gehandhabt. Für beide existieren Einträge in der Struktur bestehend aus (vgl. Abb. 23): 12 Bytes für den Dateinamen (Byte 0-11), einem Byte für die Dateiattribute (Byte 12) , 5 Bytes für das Erstellungsdatum (Byte 13-17), zwei weitere für das Datum des letzten Zugriffs (Byte 18-19), einer zwei Byte langen Referenz auf das erste Cluster (Byte 20-21), 4 Bytes für das Änderungsdatum (Byte 22-25), zwei weiteren Bytes für das erste Cluster (Byte 26-27) und vier Bytes für die Dateilänge (Byte 28-31).

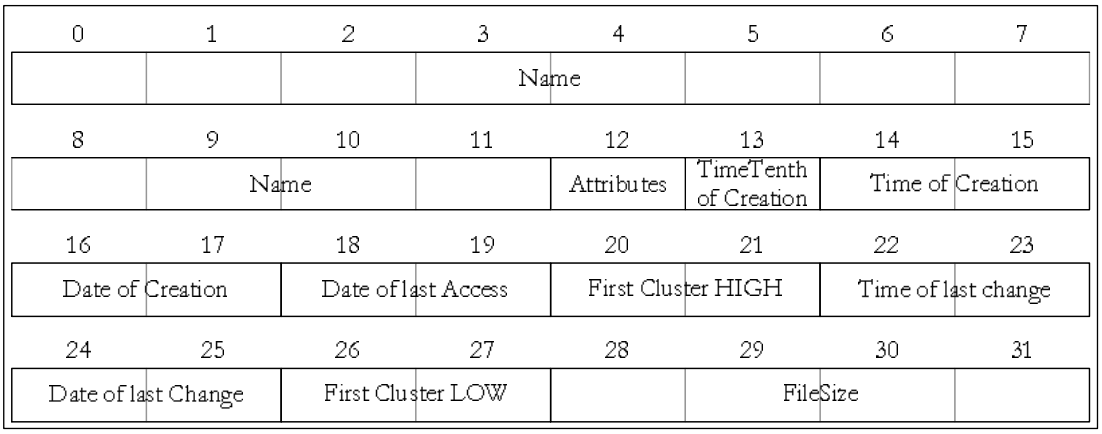


Abb. 23: FAT16-Byte-Struktur nach [MS FAT 2000]

In Byte 12 werden mit den Attributen weitere Eigenschaften abgelegt (s. Abb. 24), wobei allerdings nur sechs der acht Bits (0-5) von den derzeitigen FAT-Dateisystemen genutzt werden.

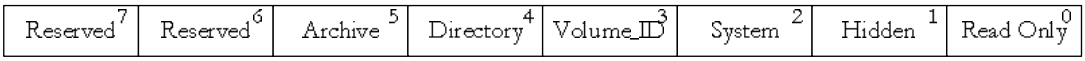


Abb. 24: Detaildarstellung des Attribute-Bytes nach [MS FAT 2000]

Im ersten Bit (Bit 0) wird hinterlegt, ob die Datei oder das Verzeichnis schreibgeschützt (*read only*) ist. Ist dieses Bit gesetzt, sollte die Datei nicht überschrieben oder gelöscht werden dürfen. Bit 1 bestimmt, ob der Verzeichniseintrag angezeigt oder dem Anwender gegenüber versteckt (*hidden*) werden soll. Dateien und Verzeichnisse, bei denen Bit 2 (s*ystem*) gesetzt ist, sollen zum Betriebssystem gehören. Bit 3 (*Volume_ID*) kennzeichnet das Root-Verzeichnis und darf entsprechend nur einmal gesetzt sein. Durch Bit 4 wird bestimmt, ob der zugehörige Eintrag ein Verzeichnis (*Directory*) beschreibt oder eine Datei. Das letzte genutzte Bit (Bit 5) kann laut Microsoft verwendet werden, um Dateien bzw. Verzeichnisse gegenüber Backup-Programmen als seit der letzten Sicherung geändert zu deklarieren.
In FAT32-Dateisystemen sind so genannte lange Namen erlaubt, die im Gegensatz zu FAT12 und FAT16 nicht mehr auf das 8.3-Format (acht Zeichen für den Dateinamen und drei Zeichen für den Dateityp) begrenzt sind. Im Vergleich zu FAT16 sind die Unterschiede im Aufbau der Einträge sehr gering und werden hier deshalb nicht gesondert betrachtet.

4) *Data Area*
   Der Zugriff auf die Daten, deren Inhalte im Datenbereich (engl. *Data Area*) gespeichert sind, erfolgt über die Einträge in der Directory Structure. Dazu wird die Nummer des ersten Clusters für die gewünschte Datei gelesen. Der Inhalt des Clusters wird dann als Anfang der Datei geladen. Danach wird die Kette aus der Dateibelegungstabelle, wie beschrieben, abgearbeitet.

Die in der Spezifikation angegebenen Dateieigenschaften, wie Dateiname, die Dateiattribute, Dateigröße und die diversen Datumsangaben werden bei genauer Betrachtung nur durch die Treiber angegeben und unterliegen damit keinem festgelegten oder unabänderlichen Algorithmus. So wird es dem Treiberprogrammierer in dem Microsoft Whitepaper [MS FAT 2000] freigestellt, ob die Felder für das Erstellungsdatum und für das Datum des letzten Zugriffs genutzt werden. Des Weiteren gibt es keinen Zusammenhang zwischen dem Wert im Feld für die Dateigröße und der tatsächlichen Dateigröße, denn die chained Clusters können durchaus länger oder auch kürzer sein als der Wert, der in der Verzeichnisstruktur in den Bytes 28 bis 31 gespeichert ist.

Hacker könnten versuchen diese Schwächen der FAT-Spezifikation auszunutzen, um z.B. Malware auf dem System dadurch zu „verstecken“, dass zum Beispiel die Dateigröße nicht geändert wird, obwohl ein Virus an einer Datei angehängt wurde. Aus diesem Grund ist eine wichtige Randbedingung für die Software, die im IRT zur Vorfallserkennung und -analyse eingesetzt werden soll, dass es alle Dateieigenschaften neben dem Dateiinhalt beim Vergleich berücksichtigt. Allerdings soll das Datum des letzten Dateizugriffs nicht berücksichtigt werden, da auf die Dateien wegen der Analyse selber sowieso zugegriffen werden muss und sich damit eine Änderung dieses Wertes ergibt.

3.3.4 Veränderungen des Dateiinhalts

Das FAT-Dateisystem wurde spezifiziert, um Programme und Daten in Dateien speichern, laden und verwalten zu können. Da die Dateieigenschaften nur Indikatoren dafür sind, dass Dateien geändert wurden, liegt der Hauptaspekt auf den Dateiinhalten[8].

Änderungen von Dateiinhalten können bei einer zeilenbasierten Betrachtung, wie folgt, aussehen:

- *Hinzufügen einer Zeile an einer Stelle*
- *Löschung einer Zeile*

Die Veränderung einer Zeile kann auch als Löschung der Originalzeile und der Hinzufügung einer neuen Zeile aufgefasst werden und wird deshalb nicht gesondert aufgeführt.

Diese Änderungen der Dateiinhalte und die Veränderung von Dateieigenschaften sollen im IRT computergestützt erkannt und analysiert werden, um damit Rückschlüsse auf die Veränderung des System-Zustandes und damit auf einen Sicherheitsvorfall ziehen zu können.

[8] Bei der Vorfallsanalyse sind die Dateieigenschaften aber durchaus interessant, da sie Aufschluss über das Vorgehen eines Angreifers geben können.

3.4 Betrachtung existierender Werkzeuge

Vor allem im Internet werden tagtäglich immer neue Software-Programme vor- und bereitgestellt. Darunter gibt es bereits eine Anzahl von Programmen, die sich mit den beschriebenen Aufgaben beschäftigen, denn die Probleme stellen sich nicht nur bei der Vorfallsanalyse, sondern auch beim Abgleichen von Dateien verschiedener Versionen. Beispiele sind die Synchronisations-Software wie der Microsoft *Aktenkoffer*, der Dateien zwischen einem Notebook und einem herkömmlichen Computer abgleicht, und dazu feststellen können muss, welche Dateien sich unterscheiden und wo die aktuellere Version sich befindet. Einen sehr ähnlichen Dienst leisten häufig auch FTP-Übertragungsprogramme, wie sie beim Hochladen von Web-Seiten genutzt werden (z.B. Ipswitch *WS_FTP Pro*).

Beide genannten Produkte legen aber nicht offen, aufgrund welcher Dateieigenschaften der Abgleich erfolgt. Diese fehlende Information macht es unmöglich, davon auszugehen, dass die ausgegebenen Differenzlisten vollständig sind.

Bei verteilter Software-Entwicklung tritt zum Beispiel bei Quellcode-Dateien ebenfalls die Aufgabenstellung auf. Hierfür werden Werkzeuge zur Versionsverwaltung wie das *Concurrent Versions System* (CVS) oder Software-Exploration's *DevPortal* verwendet. Laut [CVS 2003: 1.1] versieht CVS jede Datei im CVS-Directory mit einer Versionsnummer. Aufgrund dieser Information, die dem Nutzer verborgen bleibt, werden die Dateien als verschieden eingestuft. CVS nutzt zur Lösung des Problems also nicht die vom Dateisystem bereitgestellten Informationen, sondern verwendet nur davon losgelöste Daten. Das IRT braucht jedoch die Ermittlungsmöglichkeit, welche Eigenschaften von Dateien (wie zum Beispiel Dateigröße, -attribute, -inhalte und Datumsangaben) durch einen Vorfall geändert wurden.

Aus demselben Grund scheiden auch einfache Checksummenvergleiche für die Aufgabe „Festplatten-Zustände vergleichen" aus. Zudem besteht bei Checksummenverfahren das Problem, dass die verwendeten Hash-Funktionen injektiv sind, weil sie die große Menge der möglichen Dateiinhalte auf eine begrenzte kleinere Menge abbilden (vgl. [Pfleeger 2000: 3.7]). Bei der Anwendung einer Hash-Funktion auf zwei verschiedene Inhalte kann also dieselbe Signatur errechnet werden. Dies hängt insbesondere von den eingesetzten Funktionen ab (vgl. [Bosau 2001: 1]). Damit muss bei der Interpretation des Ergebnisses, zwei Dateien seien inhaltlich identisch, dieses Problem berücksichtigt werden. Denn die Angriffstechnik könnte zufällig oder absichtlich dafür sorgen, dass die Änderung einer Datei durch einen Checksummenvergleich nicht bemerkt wird. Die OpenSource Software *Tripwire* erlaubt dem Nutzer, eine Auswahl zu treffen, welche Hash-Funktionen genutzt werden sollen. Durch die gleichzeitige Verwendung verschiedener Funktionen, wird versucht, das eben beschriebene Problem stärker einzugrenzen. Neben den Signaturen verwendet Tripwire je nach Auswahl des Nutzers auch die anderen Dateieigenschaften (vgl. [Tripwire 1992] u. [Bosau 2001: 1]). Diese Informationen werden in einer lokalen Datenbank gespeichert und für einen späteren Vergleich mit den aktuellen Daten herangezogen. Da die Signaturen nicht aber die Dateiinhalte in der Datenbank hinterlegt werden, kann mit Tripwire keine Analyse der genauen Veränderungen der Dateiinhalte durchgeführt werden. Tripwire stellt also lediglich fest, dass der Dateiinhalt geändert wurde. Welche Inhaltsänderungen zu dem Ergebnis führen, kann nicht analysiert werden, da aus den Signaturen die ursprünglichen Dateiinhalte nicht rekonstruiert werden können und somit die Referenz fehlt.

Laut [DevPortal 2000] nutzt das zweite genannte Werkzeug für verteilte Software-Entwicklung *DevPortal* für den Vergleich die Dateiinhalte und vernachlässigt ebenfalls die anderen Dateieigenschaften. Entsprechend arbeitet auch das UNIX-Programm *diff*, das als Ausgabe ein Skript hat, mit dem die eine untersuchte Datei in die andere überführt werden kann. Das Programm diff kann auch auf Verzeichnisse angewendet werden, wobei dann wiederum Paare von Textdateien inhaltlich verglichen werden. Werden keine Unterschiede in den Dateiinhalten festgestellt, erfolgt keine Ausgabe. Ein ähnliches Werkzeug existiert für Microsoft Betriebssysteme und wurde unter anderem mit Microsoft Windows 98 als *WinDiff* ausgeliefert. Beim Verzeichnisvergleich mit WinDiff wird lediglich mitgeteilt, welche Datei neuer ist. Der Einsatz von diff oder WinDiff wäre für das IRT auch nicht zwingend verlässlich, da beispielsweise WinDiff nur das Änderungsdatum als Kriterium verwendet (vgl. [WinDiff 2003]). Es gibt aber Mechanismen, mit denen die Auswirkungen von Dateiänderungen auf das Änderungsdatum zum Beispiel mit der Software Filo (vgl. [Filo 2000]) rückgängig gemacht werden können.

Das für den Einsatz im IRT wohl ausgereifteste Werkzeug zur Erfüllung der Aufgaben ist die Shareware *Beyond Compare* von Scooter Software. Laut [BeyondCompare 2003] schöpft auch dieses Tool die erforderlichen Dateieigenschaften nicht vollends aus, da trotz Nutzung von Änderungsdatum, Dateigröße, Dateiinhalt und Dateiattribute das Erstellungsdatum nicht betrachtet wird. Über das Erstellungsdatum kann zum Beispiel beobachtet werden, ob mit den Befehlen *copy* und *xcopy* der Microsoft Betriebssysteme und Standardtreibern eine Sicherungskopie zurück geschrieben wurde.

Da sich keine Software finden ließ, die den vollen gewünschten Funktionsumfang aufweist, wird eine eigene Implementation angestrebt, die im Folgenden beschrieben wird.

3.5 Systemgestaltung

Um die Software zu strukturieren, erfolgt die Systemgestaltung gemäß dem Werkzeug-Automaten-Material-Ansatz (WAM-Ansatz, vgl. [Züllighoven & Soltos 1998: 3] und Abb. 25).

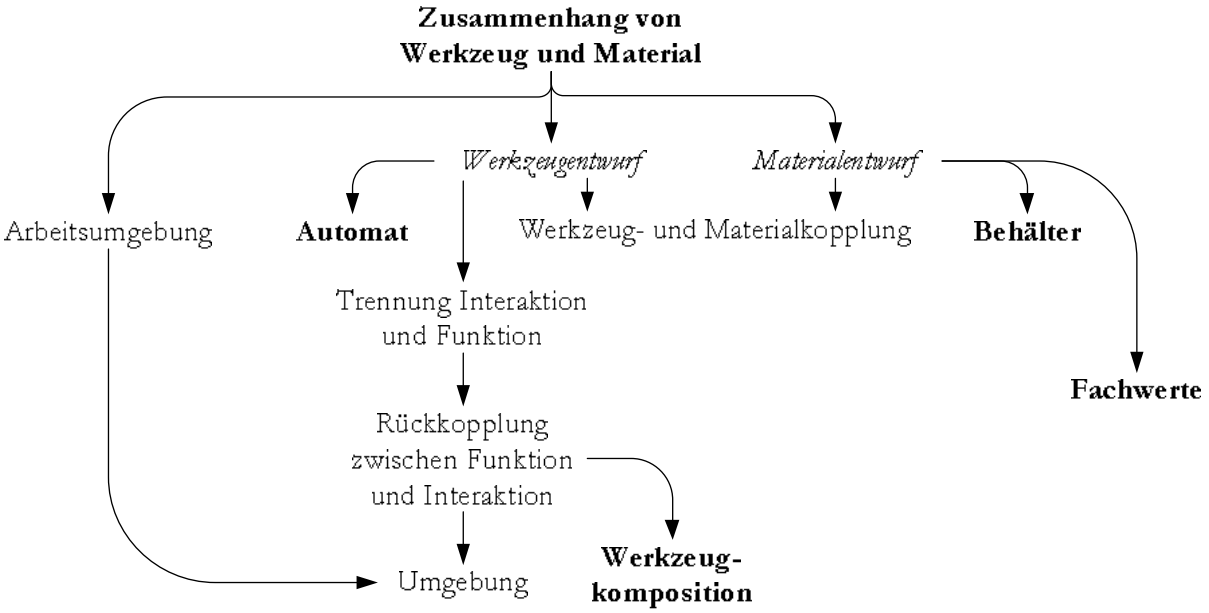


Abb. 25: vereinfachtes WAM-Konzeptionsmuster nach [Züllighoven et al. 1998: 3]

Der WAM-Ansatz sieht für interaktive Software vor, dass real existierende Begebenheiten in der Software-Architektur entsprechend repräsentiert werden. Zum Beispiel ist ein Blatt Papier modelliert in einer Software immer noch ein eigenständiges Objekt und in diesem Fall ein Material. Auf dem Papier kann mit dem Werkzeug Stift geschrieben werden, das auch in der Modellierung ein eigenes Objekt darstellt.

Der Vorteil beim Einsatz von IT-Systemen besteht meistens darin, dass die Möglichkeit einer Automatisierung für die Arbeiten besteht, die mit einem Werkzeug auf einem Material durchgeführt werden. Im Beispiel wird das Schreiben von 1000 inhaltsgleichen Briefen mittels eines Werkzeugs Stift auf dem Material Papier durch einen Automaten wie einem Drucker oder Kopierer übernommen und stellt dadurch eine Arbeitserleichterung für den Anwender dar. Entsprechend sieht der WAM-Ansatz die Implementierung von Automaten zu den Werkzeugen vor.

Das WAM-Leitbild hat damit einen wesentlichen Einfluss auf die Softwaregestaltung. Nicht nur Material und Werkzeug werden getrennt, sondern auch andere Aspekte finden Berücksichtigung. So wird das Material auch in der nicht-virtuellen Realität in *Behältern* aufbewahrt, geordnet und verwaltet (wie z.B. ein Büroordner für beschriebene Blätter). Da Material und Werkzeug auch immer in Beziehung stehen (im Beispiel kann eine Kreidetafel auch als Material aufgefasst werden, auf der ein Stift üblicherweise nicht schreiben kann), bedarf die *Werkzeug- und Materialkopplung* besonderer Beachtung. Dabei wird festgelegt, welche Werkzeuge mit welchen Materialien genutzt werden können.

Für die Werkzeuge, die in einem späteren Schritt in eine automatisierte Anwendung (*Automat*) gebettet werden können, erfolgt außerdem die *Trennung von Funktions- und Interaktionskomponenten.* Damit gibt es software-technisch keinen direkten Zusammenhang zwischen der Schnittstelle zum Menschen und den eigentlichen Funktionen. Ein Fotokopierer zum Beispiel ermöglicht dem Anwender per Tastendruck das Vervielfältigen von Texten. Die Funktionen, wie diese Vervielfältigung von statten geht, bleiben dem Nutzer verschlossen. Im Prinzip ließe sich die Funktionsweise auch austauschen, ohne dass das Ergebnis dem Anwender unterschiedlich erschiene. Gemäß dem Beispiel soll nach dem Trennungsprinzip von Funktion und Interaktion auch in der Software-Technik verfahren werden.

Sind Funktion und Interaktion getrennt, muss die *Rückkopplung zwischen Funktion und Interaktion* gesondert erfolgen. Dies wird anschaulich, wenn zeitintensive Vorgänge betrachtet werden. Beim Kopierer-Beispiel könnte der Bediener wegen der hörbaren mechanischen Vorgänge darauf schließen, dass der Kopierauftrag abgearbeitet wird. Bei IT-Systemen fehlt diese Art der Rückmeldung und bedarf anderer Methoden. Ist auch die Rückkopplung bedacht, können die Werkzeuge erstellt werden.

Arbeitsumgebung und die Umwelt im Allgemeinen stellen den Kontext dar, in dem Material und Werkzeuge anzusiedeln sind. Papier und Stift befinden sich beispielsweise in der Arbeitsumgebung eines Schreibtisches. Befindet sich dieser dann auch noch in einem Großraumbüro und werden Nachrichten von den Angestellten in dem Büro ausgetauscht, dann finden diese Umgebungen nach dem WAM-Ansatz Berücksichtigung in dem Softwareentwurf.

Durch die Nutzung des WAM-Ansatzes fließt das Konzeptionsmuster in die Systemgestaltung der Software CompareSys ein, die im Rahmen dieser Diplomarbeit entsteht. Bei CompareSys geben die besonderen Bedingungen des IRT vor, dass die Arbeitsumgebung ein abgeschotteter Einzelarbeitsplatz ist, der Zugriff auf die Festplattenzustände vor und nach dem Vorfall hat. Als Material werden die Dateien verstanden, die sich vor und nach einem Vorfall auf einem Opfersystem befinden. Sie haben analog zur Spezifikation von FAT16 (s. 3.3.3) folgende Fachwerte:

- *Name* und *Pfad*

  Dateien werden üblicherweise in Baumstrukturen organisiert und können in den Blättern des Verzeichnisbaums über einen eindeutigen Dateinamen adressiert werden.

- *Inhalt*

  Dateien dienen zur Speicherung von Informationen oder von Algorithmen. Die Informationen und Algorithmen stellen ihren Inhalt dar.

- *Dateigröße*

  Aus dem Dateiinhalt kann die Dateigröße abgeleitet werden, denn Informationen und Algorithmen werden in Bits gespeichert und die Anzahl der Bits ergibt die Dateigröße.

- *Erstellungsdatum*

  Sobald Dateien das erste Mal geschrieben werden, erhalten sie ein Erstellungsdatum, das sie bis zu ihrer Löschung behalten.

- *Änderungsdatum*

  Das Änderungsdatum ähnelt dem Erstellungsdatum, jedoch wird es bei jeder Dateiveränderung aktualisiert.

- *Dateiattribute*

  Unter den Dateiattributen werden spezielle Eigenschaften von Dateien zusammengefasst. Diese geben beispielsweise Informationen darüber, ob die Datei aufgelistet werden soll, ob sie zum Betriebssystem gehört oder ob sie nicht gelöscht werden darf.

Die Dateien werden in einem speziellen Behälter für den Festplattenzustand aufbewahrt.

Als weiteres Material für diese komparativ statische Analyse können Differenzdateien aufgefasst werden. Diese Klasse von Dateien zeichnet sich dadurch aus, dass sie als Repräsentationen der Festplattendateien vor dem Vorfall andere Eigenschaften hatten als nach dem Vorfall. Differenzdateien entsprechen im Prinzip dem zuvor beschriebenen Material Datei. Da sie jedoch zwei Dateien – nämlich die vor dem Vorfall und die nach dem Vorfall – repräsentieren, bestehen die Differenzdateieigenschaften auch aus allen Eigenschaften der beiden Dateien.

Zuletzt gibt es noch den Materialtyp, der die Unterschiede der Dateiinhalte darstellt (Inhaltsdifferenzen). Objekte dieser Materialklasse bestehen aus den Skripten, die die ursprüngliche Datei in die veränderte Datei überführen können, und dem ursprünglichen Dateinamen.

Als Werkzeuge werden für CompareSys aufgefasst:

- *Vergleichswerkzeug für Festplattenzustände*

  Dieses Werkzeug überprüft Dateien aus zwei Behältern darauf, ob eine Datei des einen Behälters im anderen Behälter an gleicher Stelle vorhanden ist und ob die Eigenschaften identisch sind. Die zu untersuchenden Dateien werden ihrem Ursprungsbehälter entnommen. Werden Unterschiede zwischen zwei Dateien festgestellt, kommen diese als Differenzdatei in einen gesonderten Behälter, ansonsten werden sie verworfen.

- *Vergleichswerkzeug für Dateiinhalte*

  Beim Vergleich der Dateiinhalte wird dieses Werkzeug auf die Differenzdateien angewendet. Das Werkzeug ermittelt dabei, an welchen Stellen im Dateiinhalt Ergänzungen, Änderungen oder Löschungen vorgenommen wurden. Es erarbeitet aus den Differenzdateien die Inhaltsdifferenzen.

Der Automat in CompareSys soll die Anwendung der Werkzeuge nicht nur für einzelne Dateien bzw. Differenzdateien ermöglichen, sondern auch für deren Behälter.

3.6 Softwareentwurf

Der Softwareentwurf erfolgt gemäß dem WAM-Ansatz. Somit werden Werkzeug und Material genauso wie Funktions- und Interaktionskomponenten, wie beschrieben, getrennt und auch getrennt implementiert.

Zusammengeführt werden die Komponenten durch das Modul CmpSys (s. Abb. 26), in dem diverse Rückkopplungsfunktionen und auch die Main-Prozedur zum Programmstart enthalten sind. Mit Main werden die Funktionskomponenten der Werkzeuge für den Festplatten- und den Dateiinhaltsvergleich und auch deren Interaktionskomponenten initialisiert.

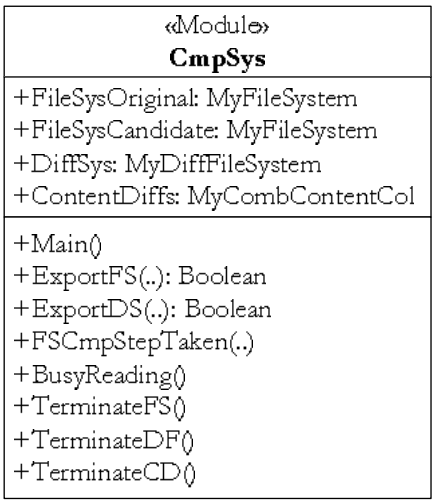


Abb. 26: CmpSys-Modul

Das Material (s. Abb. 27) besteht aus den Klassen, die den Festplattenzustand als Sammlung (MyFileSystem) von Dateien (MyFile) in der Software repräsentieren und aus einer Sammlung (MyDiffFileSystem), in der als Elemente Differenzdateien (MyDiffFile) enthalten sind.

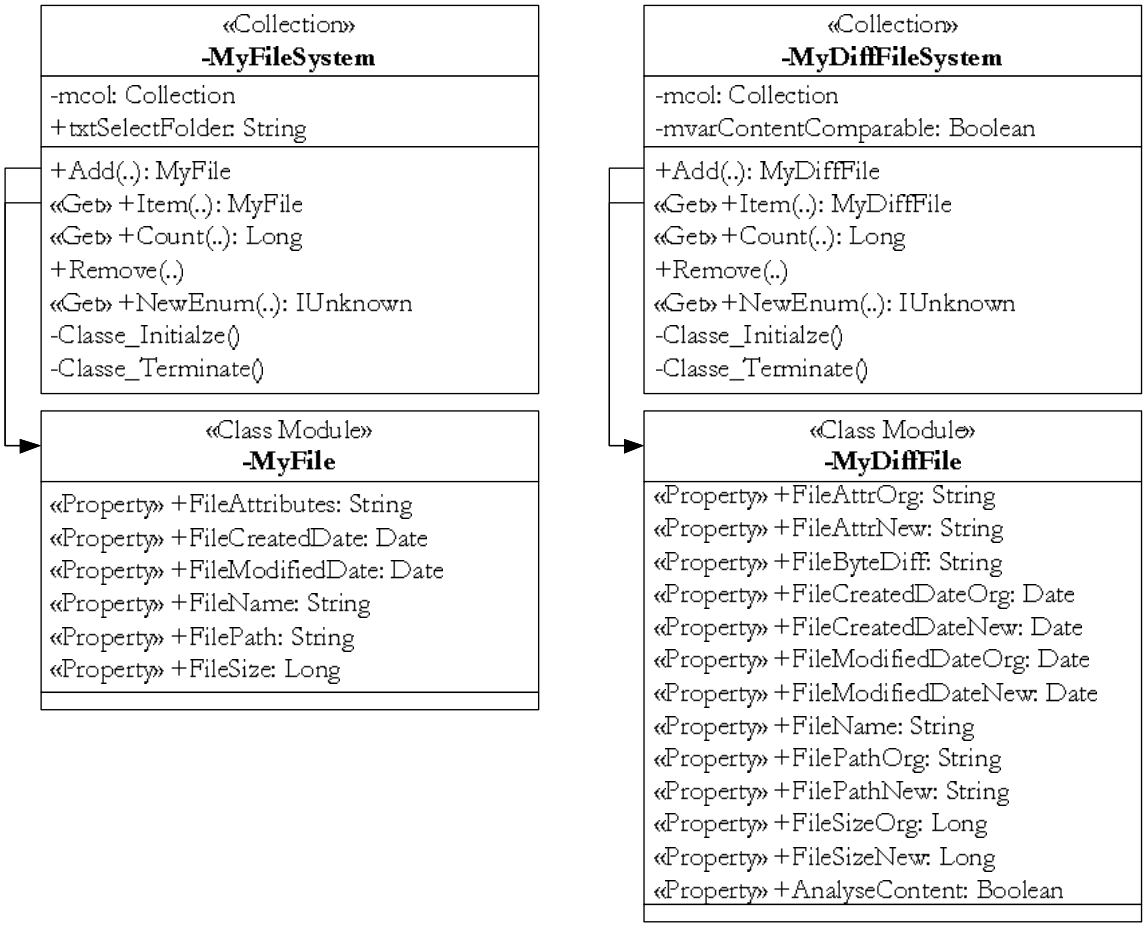


Abb. 27: Material-Klassen für die Festplattenanalyse

Des Weiteren werden die analysierten Unterschiede der Dateiinhalte (Inhaltsdifferenzen) in Objekten der Klassen MyCombContentFile abgebildet. Sie werden in dem Behälter MyCombContentCol aufbewahrt, der wie alle anderen Behälter Zugriffsmethoden auf die Elemente des jeweiligen Behälters zur Verfügung stellt (s. Abb. 28).

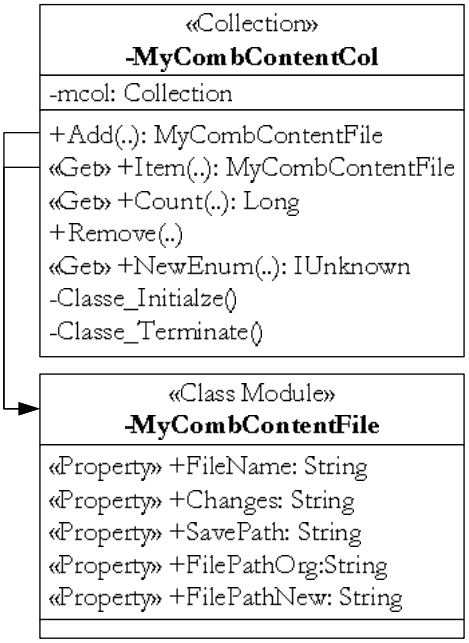


Abb. 28: Materialklassen für die Ergebnisse der Dateiinhaltsanalyse

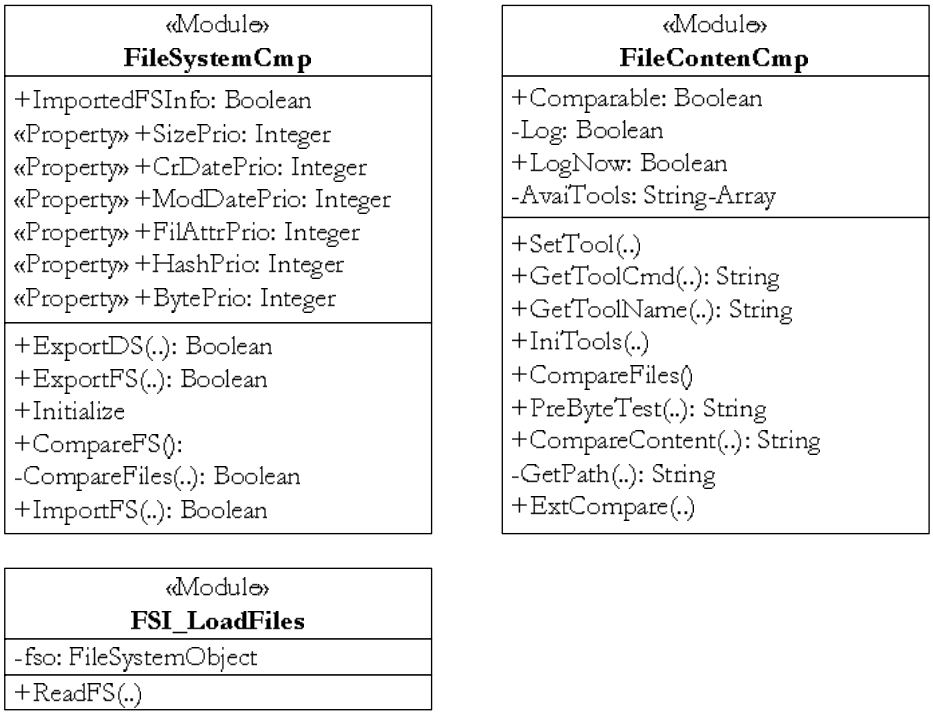


Abb. 29: Haupt-Funktionskomponenten

Die Werkzeuge, die auf dem genannten Material arbeiten können, werden unterteilt in Funktions- und Interaktionskomponenten. Die Funktionskomponenten (s. Abb. 29) sind Module, die initial über das Hauptmodul CmpSys gesteuert werden. Die Module FileSystemCmp und FileContentCmp kapseln dabei jeweils die Funktionen für den Vergleich der Festplattenzustände beziehungsweise für den Vergleich von Dateiinhalten. Damit erfolgt auch im Software-Entwurf eine Trennung der beschriebenen Aufgaben und die Bereitstellung separater Werkzeuge für diese Aufgaben (s. 3.5).

Neben den Hauptaufgaben des Vergleichs führen beide Funktionskomponenten noch Einstellungsparameter. So verfügt FileContentCmp über eine Aufzählung verwendbarer externer Werkzeuge (AvaiTools). Beim Vergleich der Festplattenzustände durch FileSystemCmp kann zur Effizienzoptimierung die Reihenfolge bestimmt werden, in welcher die Dateieigenschaften verglichen werden sollen. Dies wird durch eine Prioritätszuordnung zu den Dateieigenschaften ermöglicht (z.B. der Wert in SizePrio bestimmt die Priorität des Dateigrößenvergleichs).

Den Zugriff auf die realen Dateisysteme stellt das Modul FSI_LoadFiles her, das im Wesentlichen nur für den Aufbau der MyFileSystem-Behälter für das Originalsystem und für das veränderte System benötigt wird.

Zwei weitere Module (s. Abb. 30) stellen Import- und Export-Funktionen bereit. Das Modul ImExportReg bietet insbesondere den Modulen FileSystemCmp und FileContentCmp den Zugriff auf die zentrale Betriebssystem-Konfigurationsdatei, der so genannten Registry. In der Registry werden für CompareSys die Vergleichsreihenfolge der Dateieigenschaften und die externen Werkzeuge für den Vergleich von Dateiinhalten hinterlegt.

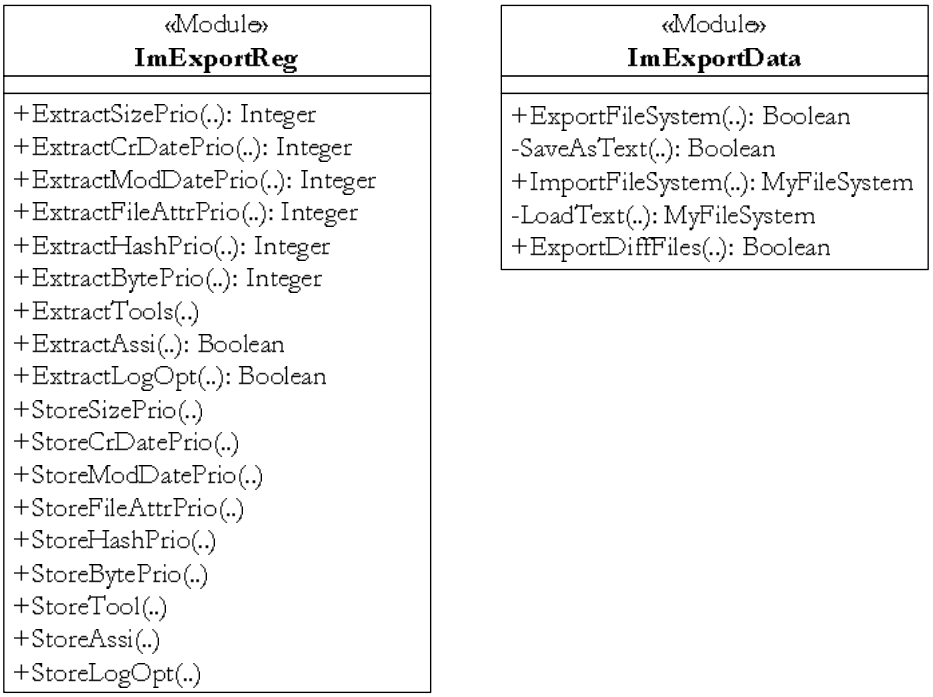


Abb. 30: Funktionssubkomponenten

Das Modul ImExportData ermöglicht das Speichern und Laden der Materialbehälter, damit ein Austausch zu gegebenenfalls anderen Programmen durchgeführt werden kann.

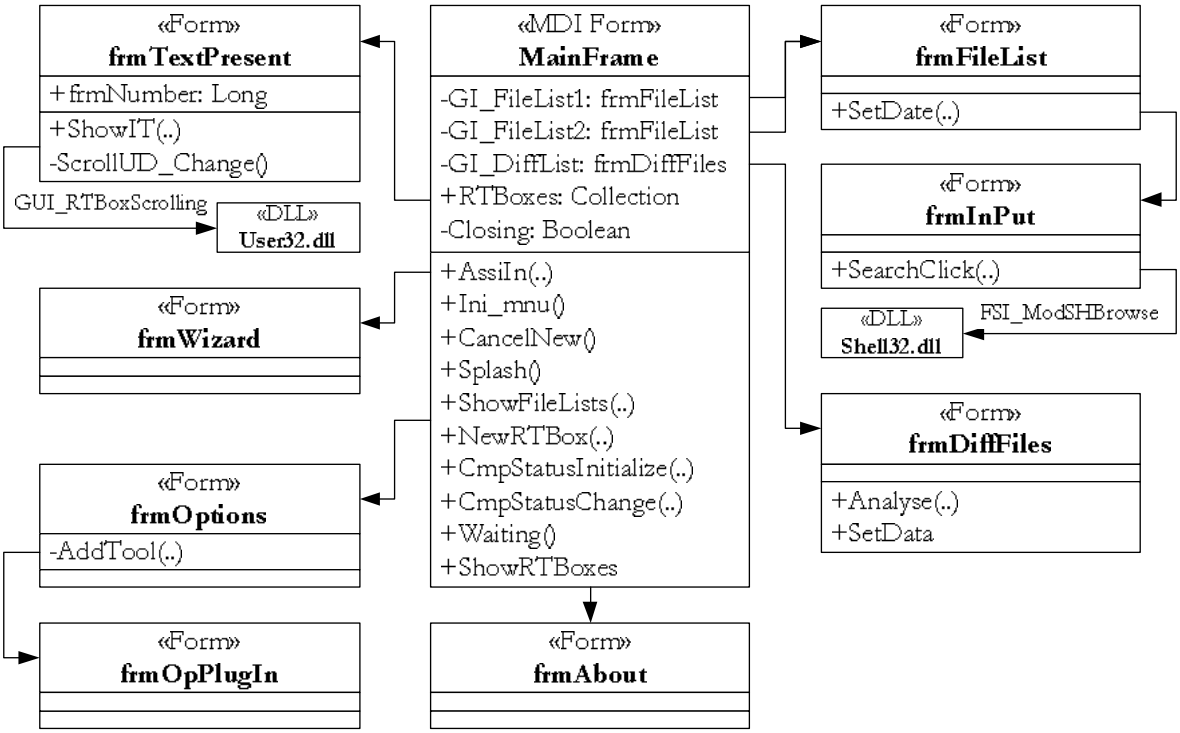


Abb. 31: Interaktionskomponenten

Die Schnittstellen zwischen den Funktionskomponenten und dem Anwender werden in Interaktionskomponenten (s. Abb. 31) realisiert. Sie bieten dem Anwender die Möglichkeit, das Material zu sichten, damit zu arbeiten und Rückmeldungen zu erhalten.

Die Hauptaufgabe für die Interaktionskomponenten übernimmt das Formular MainFrame. Aus diesem werden alle anderen Fenster verwaltet und gesteuert. Bevor dem Anwender Main-

Frame erscheint, wird eine Instanz der Fensterklasse frmSplash angezeigt, um den Anwender über den Programmstart zu informieren. Währenddessen durchlaufen die Funktionskomponenten diverse Initialisierungsprozesse. Die Komponenten frmWait und frmCmpStatus übernehmen analog zu dem zuvor geschilderten Sachverhalt rein informative Aufgaben (s. Abb. 32).

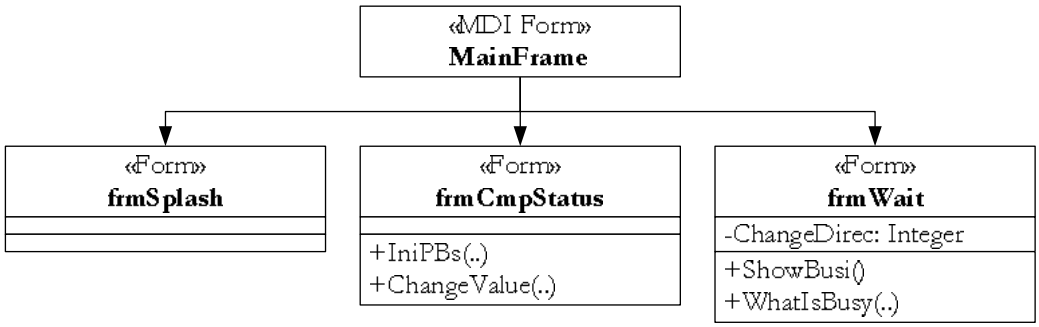


Abb. 32: GUI-Komponenten mit nur informativen Aufgaben

Mit MainFrame werden neben einem frmSplash-Objekt vier weitere Objekte instanziiert: GI_FileList1 und GI_FileList2 entstammen der Formularklasse frmFileList und präsentieren dem Anwender die Dateien des Originalsystems beziehungsweise des veränderten Systems. Vor der Präsentation der Daten müssen diese durch eine Funktionskomponente eingelesen werden. Dazu muss der Anwender der Software mitteilen, wo sich die zu untersuchenden Festplatten bzw. deren Zustandsspiegelungen befinden. Diese Eingabe erfolgt über das Formular frmInPut. Um die Eingabe möglichst komfortabel zu gestalten, nutzt frmInPut das Modul FSI_ModSH-Browse als Schnittstelle zur shell32.dll, die ein anwenderfreundliches Durchsuchen der Festplatten ermöglicht. Als weiteres Objekt wird GI_DiffList von der Klasse frmDiffList mit MainFrame instanziiert. In diesem Formular werden dem Anwender alle Differenzdateien angezeigt. Als letztes Objekt wird der Behälter RTBoxes erzeugt. Dieser Behälter beinhaltet alle Formulare des Typs frmTextPresent, die dem Anwender die Unterschiede in den Dateiinhalten verdeutlichen sollen.

Für die Konfiguration der Vergleiche ermöglicht das Formular frmOptions die Eingabe von Parametern. Hier werden die Prioritäten für den Dateivergleich eingestellt und die externen Vergleichswerkzeuge mittels frmOptPlugIn integriert.

CompareSys besteht also zusammenfassend aus den im Komponentendiagramm (s. Abb. 33) aufgeführten Klassen und Modulen.

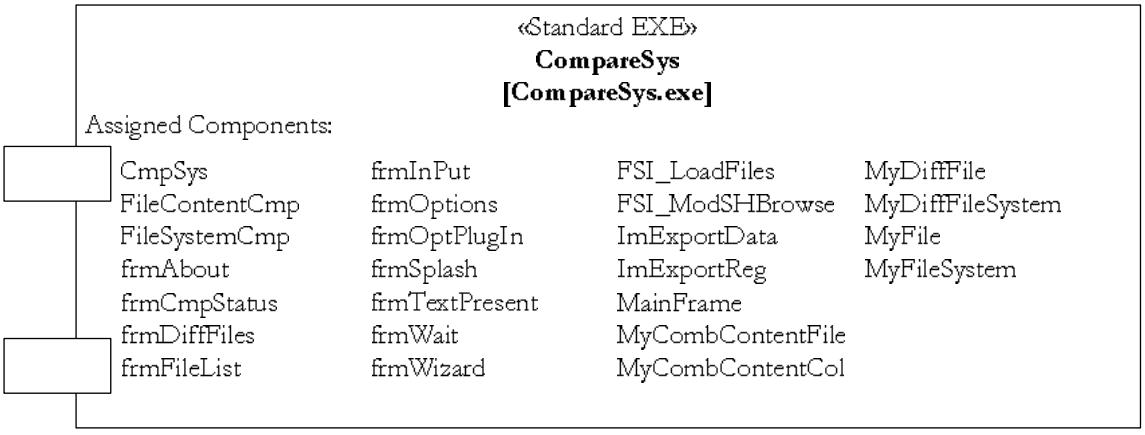


Abb. 33: UML-Komponentendiagramm für CompareSys

3.7 Implementation

In diesem Anschnitt wird die software-technische Realisierung des Festplattenvergleichs und der Analyse von Dateiinhaltsänderungen beschrieben.

3.7.1 Vergleich der Festplattenzustände

Der Vergleich der Festplatten von vor und nach dem Sicherheitsvorfall erfolgt durch das Einlesen der Dateien in Listen. Dabei wird für jede vorgefundene Datei ein Eintrag mit ihren Eigenschaften in der jeweiligen Liste vorgenommen. Das Listenelement kann über den eindeutigen, relativen Pfad adressiert werden. Nachdem eine Liste für den ursprünglichen Festplattenzustand (vor dem Vorfall, *OriginalSys*) und eine Liste für den aktuellen Zustand (nach dem Vorfall, *CurrentSys*) erzeugt sind, erfolgt der Vergleich (vgl. Abb. 34).

```
For Each CurFile In CurrentSys
  If Exists(OriginalSys.Item(CurFile.Path)) Then
    different = False
    If OrgFile.Properties ≠ CurFile.Properties Then different = True
    DiffContent = (OpenFile(OrgFile) ≠ OpenFile(CurFile))
    If different Or DiffContent Then Differences.Add OrgFile, CurFile
    OriginalSys.Remove (CurFile.FileName)
  Else
    Differences.Add 0, CurFile
  End If
  CurrentSys.Remove (CurFile.FileName)
Next Element

For Each OrgFile In OriginalSys
  Differences.Add OrgFile, 0
  OriginalSys.Remove (OrgFile.FileName)
Next Element
all created, changed or deleted files are listet in differences
```

Abb. 34: Algorithmus zur Ermittelung veränderter Dateien

Zunächst wird die Existenz der Dateien beziehungsweise deren Eintrag in der jeweiligen Liste ermittelt. CompareSys beginnt mit der Liste für den aktuellen Zustand (CurrentSys) und sucht für jedes Listenelement nach dem Gegenstück in der Liste für den ursprünglichen Festplattenzustand (OriginalSys). Existiert auch in OriginalSys ein Element mit demselben relativen Pfad, dann werden die gespeicherten Eigenschaften in einem direkten Vergleich auf absolute Übereinstimmung geprüft. Sind die Elemente nicht identisch, wird ein Element in einer dritten Liste (*differences*) für die unterschiedlichen Dateien (Differenzdateien) angelegt. Sie beinhalten die kombinierten Eigenschaften der beiden ursprünglichen Elemente. Unabhängig von den Unterschieden der verglichenen Elemente, werden beide aus den jeweiligen Listen OriginalSys und CurrentSys entfernt.

Wird in OriginalSys kein Eintrag mit demselben relativen Pfad gefunden, dann wurde die Datei also während des Vorfalls erzeugt. In einem solchen Fall wird ebenfalls ein Element in differences angelegt, wobei dieses nur Informationen über die Datei im aktuellen Festplattenzustand enthält. Das untersuchte Element aus der Liste des aktuellen Festplattenzustandes wird dann ebenfalls gelöscht.

Nachdem alle Elemente aus CurrentSys untersucht und anschließend entfernt wurden, ist die Liste leer. Da bei Vorfinden eines Elementes mit übereinstimmendem, relativem Pfad in OriginalSys auch dieses entfernt wurde, sind in OriginalSys nur noch Elemente enthalten, die während des Vorfalls gelöscht wurden. Solche Dateien werden analog zu den Dateien, die während des Vorfalls erzeugt wurden, in differences aufgenommen.

Der Algorithmus endet mit der Ausgabe der Liste differences, in der alle während des Vorfalls erzeugten, geänderten und gelöschten Dateien enthalten sind. Für geänderte Dateien folgt in der Implementation die Untersuchung, wie sich die Dateiinhalte geändert haben.

3.7.2 Analyse der Dateiinhaltsänderungen

Bei der Analyse der Änderungen der Dateiinhalte soll nach der größten gemeinsamen Sequenz (*LCS* – longest common sequence) gesucht werden. Eugene W. Myers stellte bereits 1986 in [Myers 1986: 2] fest, dass LCS-Probleme mit der Problemklasse der kürzesten Editierskripte (*SES* – Shortest Edit Script) übereinstimmen. Es ist also irrelevant, ob die größte Gemeinsamkeit für zwei Dateien oder ein minimaler Weg zur Überführung einer Datei in eine andere gefunden werden soll. Des Weiteren löst Myers das LCS- bzw. SES-Problem mittels der Graphentheorie. Beim Vergleich zweier Wörter A und B erzeugt er zunächst eine Matrix, indem er auf einer Achse das Wort A und auf der anderen Achse das Wort B einträgt. Die Felder der Matrix werden dann als Punkte verstanden (s. Abb. 35 mit A = TOTO und B = LOTTO).

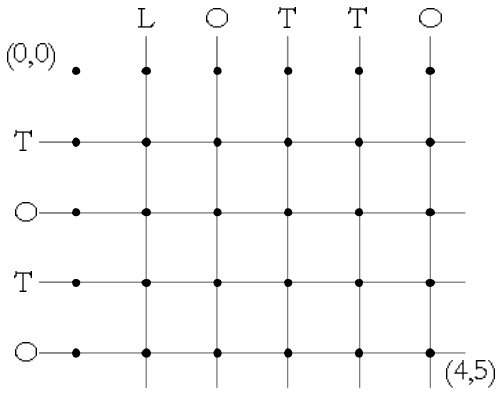


Abb. 35: Graphenaufbau für „TOTO“ und „LOTTO“ nach [Myers 1986: 2]

Nachdem der Grundaufbau des Graphen hergestellt ist, werden die Übergänge von einem Koordinatenpunkt zu einem Benachbarten nach folgendem Verfahren eingezeichnet: Sofern das senkrecht geschriebene Wort (hier: A) in das waagerechte (hier: B) überführt werden soll, bedeuten gerichtete waagerechte Verbindungen (in Abb. 36 blau gekennzeichnet), dass der Buchstabe dessen Spalte nach dem Schritt erreicht wird, geschrieben wurde. Gerichtete senkrechte Verbindungen bedeuten entsprechend, dass der Buchstabe aus Wort A gelöscht wurde. Zu den waage- und senkrechten Verbindungen trägt Myers noch gerichtete diagonale ein, wenn identische Buchstaben vorliegen.

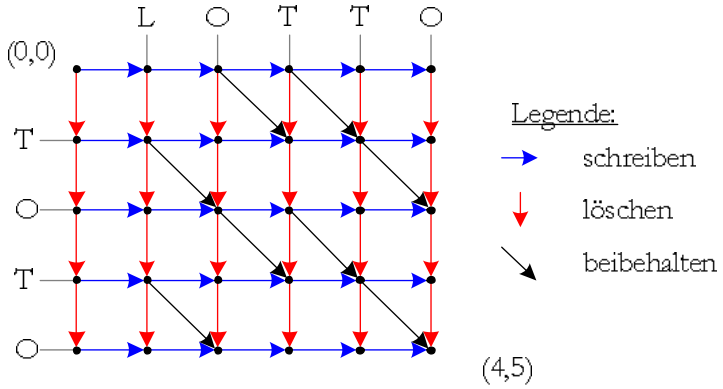


Abb. 36: Überführungen des Wortes „TOTO“ nach „LOTTO“ nach [Myers 1986: 2]

Sämtliche so entstandenen Pfade von dem Punkt (0,0) zu dem Punkt (Länge des Wortes A, Länge des Wortes B) stellen die Möglichkeiten dar, wie das Wort A in das Wort B überführt werden kann.

Für die Lösung der LCS- bzw. des SES-Probleme reicht es nach [Myers 1986: 3] nun aus, den kürzesten Pfad vom Punkt (0,0) zum Punkt (Länge des Wortes A, Länge des Wortes B) in diesem Graphen zu finden. Anstatt Dijkrastas bekannten „Shortest Path“-Algorithmus zu verwenden (vgl. [P3 1999: S.73 ff.]), entwickelte Myers einen Speicher und Zeit optimierten Algorithmus (s. Abb. 37), der diagonale Schritte bevorzugt, indem die Länge von diagonalen Schritten null entspricht und alle anderen die Länge eins haben.

Des Weiteren nutzt Myers ein Feld V zur Speicherung der bereits besuchten Punkte, wobei lediglich die A-Koordinaten gespeichert werden müssen, da die B-Werte durch Subtraktion des Feldindexes bestimmt werden können.

```
Constant MAX = Length(A) + Length(B)
Var V: Array [- MAX .. MAX] of Integer
V[1] ← 0
For D ← 0 to MAX Do
  For k ← -D to D in steps of 2 Do
    If k = -D or k ≠ D and V[k- 1] < V[k+ 1] Then
      x ← V[k+ 1]
    Else
      x ← V[k- 1]+1
      y ← x - k
      While x < N and y < M and a x + 1 = b y + 1 Do (x,y) ← (x+1,y+1)
        V[k] ← x
        If x ≥ N and y ≥ M Then
          Length of an SES is D
          Stop
```

Abb. 37: Algorithmus zur Ermittelung der Länge des kürzesten Editierskriptes aus [Myers 1986: 3]

Der Algorithmus bricht mit der Ausgabe der Pfadlänge D ab, wenn das Ziel – also der Punkt (Länge des Wortes A, Länge des Wortes B) – erreicht wurde (s. Beispiel in Abb. 38).

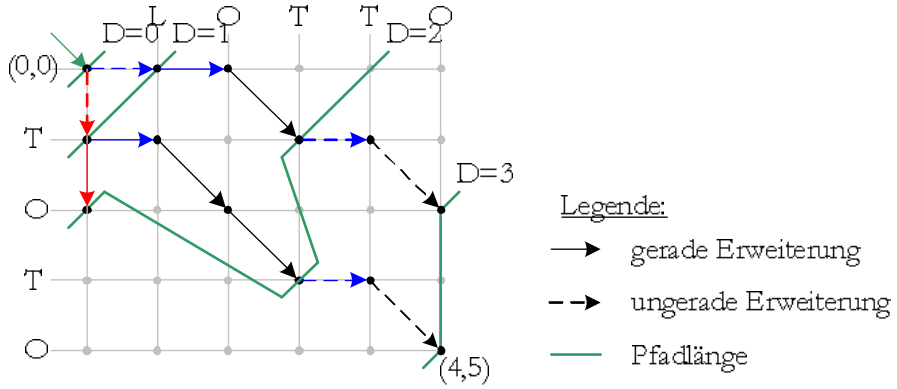


Abb. 38: Pfadlängen nach [Myers 1986: 3]

Aufgrund der speicher- und zeitoptimierten Gestaltung des Algorithmus erfolgt keine Ausgabe des Pfades. Dieser kann nur ermittelt werden, wenn während der Suche des Pfades in einer Liste dokumentiert wird, welche Punkte mit welcher Gesamtpfadlänge besucht wurden. Mit der Liste kann ausgehend vom ursprünglichen Zielpunkt rekursiv ermittelt werden, wie der Pfad aussieht.

Der Algorithmus von Eugen W. Myers hat bei der Ermittlung der Pfadlänge eine Zeit- und Platzkomplexität von O((L(A)+L(B))·D), wobei L(A) bzw. L(B) die Länge des zu untersuchenden Wortes A bzw. B und D die Anzahl der Veränderungen ist (vgl. [Myers 1986: 3]).

Gegenüber anderen Algorithmen, wie zum Beispiel dem Algorithmus vom Dan Hirschberg (vgl. [Eppstein 1995]), hat Myers ein zeit-effizienteren Algorithmus entwickelt, solange die Anzahl der Änderungen niedrig ist. Hirschbergs Algorithmus, der im Surveiller des Malware Crawlers (vgl. [Freitag 2000] und [Soller 2002]) eingesetzt wird, hat für die Berechnung der Pfadlänge eine lineare Platz- und eine Zeitkomplexität von O(L(A) ·L(B)). Solange die Anzahl der Änderungen unter der Effizienzgrenze bleibt, ist der Algorithmus von Myers unter Zeitaspekten zu bevorzugen.

Effizienzgrenze:

$$D \leq \frac{L(A) \cdot L(B)}{L(A) + L(B)}$$

Je größer die Dateilängen und je geringer die Anzahl der Änderungen sind, desto vorteilhafter ist der Algorithmus von Myers gegenüber dem von Hirschberg. Übersteigt die Anzahl von Änderungen den Quotienten der Effizienzgrenze, hat der Myers Algorithmus Nachteile. Aufgrund der Erfahrungen des IRT befindet sich meistens die Anzahl der Dateiinhaltsänderungen bei Computersicherheitsvorfällen unterhalb dieser Effizienzgrenze, was die Nutzung des Algorithmus von Myers sinnvoll macht.

Mit den beschriebenen und zugrunde liegenden Algorithmen verfügt CompareSys über die Funktionalität, die vom IRT gefordert wurde. Anhand von Versuchsreihen, die im folgenden Kapitel beschrieben werden, sollen diese Behauptung belegt und die Grenzen der Software erkannt werden.

4 Versuchsreihen mit CompareSys

Das in dem vorangegangenen Kapitel beschriebene Werkzeug *CompareSys* wird im Incident Response Team des Fachbereichs Informatik der Universität Hamburg eingesetzt. Die folgenden Versuche wurden in dem Projekt in enger Zusammenarbeit mit Nils Michaelsen durchgeführt. Michaelsen führte im Rahmen seiner Diplomarbeit „Penetrationstests – Möglichkeiten und Grenzen“ (s. [Michaelsen 2003]) Angriffe auf einen Web-Server durch, deren Auswirkungen auf das System durch CompareSys Version 1.2.0 ermittelt und analysiert wurden.

4.1 Allgemeiner Versuchsaufbau

Der Versuchsaufbau orientierte sich an einem Szenario: Die Firma Alarm Mayer betrieb auf einem eigenen Server eine Web-Site (s. Abb. 39), auf der die Firma sich und ihre Produkte präsentierte. Bereitgestellt wurden diese Informationen durch den Microsoft *Internet Information Services* 5 (IIS) in Verbindung mit *PHP* 4.3.2RC3 und *MySQL* 3.23.56. Die Software lief auf dem Microsoft Betriebssystem *Windows 2000 Professional*, das auf einer FAT32-Partition installiert war. Bedenklich im Sinne der IT-Sicherheit war das Versäumnis, sämtliche verfügbaren Bugfixes, Patches, Updates und Service Packs einzuspielen, und auch eine Firewall, ein Intrusion Detection System und AntiMalware waren nicht installiert. Immerhin wurde der Web-Server regelmäßig mit dem Image-Tool *Ghost* von Norton gesichert, so dass im Bedarfsfall nicht nur die Web-Site, sondern die gesamte Software des Web-Servers inklusive des Betriebssystems zurückgespielt werden konnte.

Abb. 39: Homepage der Firma Alarm Mayer

Im Laufe des Betriebes wurden Veränderungen an dem System und Sicherheitsverletzungen durch den Betreiber erkannt. Diese Änderungen konnten nicht durch legitime Handlungen der Administratoren erklärt werden. Deswegen wurde CompareSys verwendet, um sämtliche Abweichungen zu entdecken und zu analysieren.

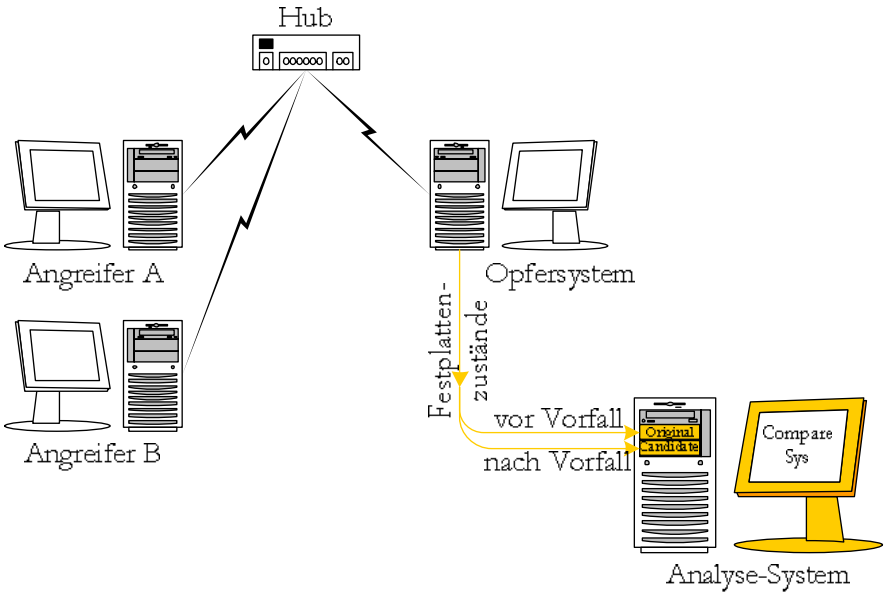


Abb. 40: Versuchsaufbau

Um das Internet zu simulieren, befanden sich im Versuchsaufbau (s. Abb. 40) neben dem Web-Server als Opfersystem zwei weitere Computer, die über einen Hub mit dem Web-Server verbunden waren. Diese zwei Rechner stellten aus Sicht des Web-Servers die Angreifer dar und dienten Michaelsen für die Penetrationstests (vgl. [Michaelsen 2003: 6]). Neben diesen vernetzten Systemen wurde als Stand-Alone noch ein Analyse-System mit CompareSys betrieben. Das System verfügte über Wechselplatten-Slots, mit denen mittels verschiedener Festplatten unterschiedliche Zustände ins CompareSys-System gebracht werden konnten.

4.2 Änderungen durch ordentlichen Gebrauch

Der IT-System- und insbesondere der Festplattenzustand verändern sich durch den Gebrauch. Beim Starten des Betriebssystems, Laden von Anwendungsprogrammen, Bearbeiten von Daten werden Dateien geöffnet, geschrieben und gelöscht. Deshalb war das Ziel des ersten Versuches, ein besseres Verständnis für das Betriebssystem zu bekommen und die Änderungen aufgrund eines Neustarts und diverser Web-Seiten-Aufrufe festzustellen. Eine solche Versuchsreihe sollte immer der eigentlichen Vorfallsuntersuchung vorausgehen, um ein Grundverständnis für Änderungen am System- bzw. Festplattenzustand zu erlangen, wie sie im jeweiligen Anwendungskontext üblich sind. Dabei ist es essentiell, dass die in diesem Versuch genutzten IT-System-Daten „sauber“ sind, das heißt, nicht durch Hackerangriffe beeinflusst, durch Malware infiziert und sonst wie verändert worden sind. Die Erkenntnisse aus dieser Versuchsreihe sollen als Basiswissen über Änderungen des IT-Systems dienen. Würden hier zum Beispiel schon Änderungen durch Malware als „ordentlicher Gebrauch“ deklariert, dann führt diese Annahme zu falschen Ergebnissen bei Folgeuntersuchungen.

Für die Versuchsreihe wurde das nach der Fertigstellung der Installation und Konfiguration angelegte Image auf zwei Festplatten zurückgespielt, wobei davon ausgegangen wurde, dass das Image von einer sauberen Installation stammt. Der Web-Server bootete mit einer der Festplatten. Danach besuchten simulierte Anwender von diversen Rechnern die Web-Seiten der Firma Alarm Mayer. Im Anschluss wurde diese Festplatte mittels CompareSys mit der zweiten unveränderten

Festplatte verglichen. Aufgrund dieses Versuches konnten insgesamt 32 Dateiänderungen und zwei neue Dateien ausgemacht werden, die im Folgenden anhand der Anlage C erläutert werden.

Das Verzeichnis WINNT/CSC wird vom Betriebssystem als so genannter *Client Side Cache* (CSC) verwendet (vgl. [Krugler 2003]). Dort abgelegte Dateien (Nr. 1, 2 und 16 in der Anlage C) können von einem Anwender laut [MS CSC 2002] genutzt werden, um diese ohne Netzverbindung weiter nutzen und bearbeiten zu können. Die vorgefundenen Dateien *00000001*, *00000002* und *csc1.tmp* werden von dem Dienst erstellt und genutzt. Aufbau und Nutzung der Dateien und des Dienstes sind jedoch nicht genau dokumentiert, so dass eine fundierte Analyse erschwert wird, zumal die Informationen in den Dateien auch nicht in Klartext geschrieben werden.

Besser dokumentiert ist der Nutzen der Dateien *ipsecpa.log* (Nr. 3 in der Anlage C) und *oakley.log* (Nr.5) sowie deren Sicherungen *ipsecpa.log.last* (Nr. 4) und *oakley.log.sav* (Nr.6). Laut [Lancaster 2002] werden in den Dateien Informationen über IPSec-Aktivitäten (*ipsecpa.log*) und diesbezügliche Schlüsselaustauschinformationen (*oakley.log*) gespeichert. Um derartiges durch die Local Security Authority Service Shell (LSASS) direkt protokollieren zu können, werden die Dateien bei Systemstart geladen und beim Herunterfahren des Betriebssystems wieder gespeichert. Dadurch ergeben sich neue Werte für das Änderungsdatum, auch wenn wie im vorliegenden Fall nichts zu protokollieren war und die Dateien leer blieben. Ebenfalls leer war in diesem Versuch die Datei *PASSWD.LOG* (Nr. 7). Auch hier wurde das Änderungsdatum erneuert, denn die Datei wird laut [Lancaster 2002] von der LSASS genutzt, um Informationen über lokale Benutzerkonten abzulegen, insbesondere wenn diese automatisierten Änderungen unterzogen werden, was in diesem Versuch nicht der Fall war.

Die Dateien *AppEvent.Evt* und *SysEvent.Evt* (Nr. 8 und 9 in der Anlage C) gehören zu der Ereignisanzeige des eingesetzten Microsoft Windows 2000 Professional. Die Daten der Datei *AppEvent.Evt* werden von Microsoft als Anwendungsprotokoll bezeichnet. „Das Anwendungsprotokoll enthält Ereignisse, die von Anwendungen oder Programmen aufgezeichnet wurden. Von einem Datenbankprogramm könnte etwa ein Dateifehler im Anwendungsprotokoll aufgezeichnet werden. Die Entwickler des jeweiligen Programms entscheiden, welche Ereignisse überwacht werden“ (aus [MS MMC 2001]). Entsprechend werden die Daten der Datei *SysEvent.Evt* als Systemprotokoll bezeichnet. „Das Systemprotokoll enthält Ereignisse, die von den Windows 2000-Systemkomponenten aufgezeichnet wurden. So wird beispielsweise das Fehlschlagen des Ladens eines Gerätetreibers oder einer anderen Systemkomponente beim Betriebssystemstart im Systemprotokoll aufgezeichnet. Die von den Systemkomponenten aufgezeichneten Ereignistypen sind durch Windows 2000 Professional vordefiniert“ (aus [MS MMC 2001]). Da sich der Inhalt der beiden Dateien nicht geändert hat, wird deutlich, dass das Betriebssystem sie öffnet, um Ereignisse direkt eintragen zu können. Wie das Änderungsdatum für beide Dateien zeigt, werden diese erst gespeichert, wenn das Betriebssystem wieder beendet wird.

Die Existenz und Änderungen der Protokoll-Dateien *PollSt.txt* (Nr. 11 in der Anlage C) und *Pollog.txt* (Nr. 12) hängen mit dem Einsatz der Grafiktreiber *Catalyst* von ATI Technologoies zusammen. Laut [ATI IB 4218] nutzen die Treiber die Dateien, um in der *Polllog.txt* die Instanzierung der Treiber und in *PollSt.txt* die Nutzung der Schnittstellen zum Betriebssystem zu proto-

kollieren. Die Aufzeichnungen werden in *Pollog.txt* ersetzend geschrieben (s. Abb. 41) – deshalb verändert sich die Dateigröße nicht – und in *PollSt.txt* angehängt (s. Abb. 42).

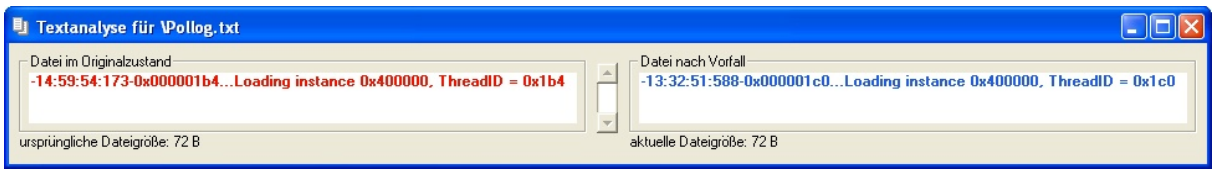


Abb. 41: CompareSys-Darstellung[9] für die Inhaltsänderungen in der Datei Pollog.txt

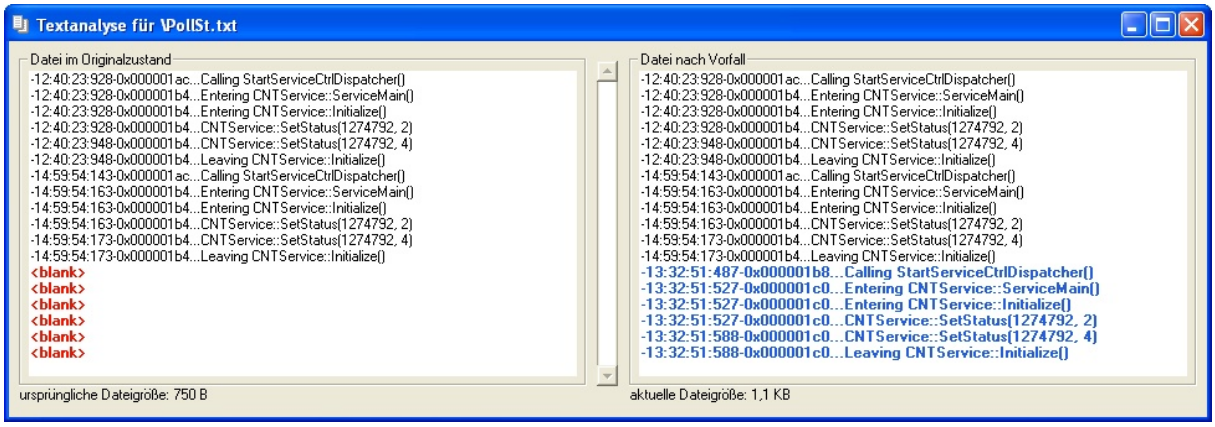


Abb. 42: CompareSys-Darstellung[10] für die Inhaltsänderungen in der Datei PollSt.txt

Zu der Registry, die in Microsoft Betriebssystemen seit Windows 3.1 (und verstärkt seit Windows 95 und NT 4.0) als Konfigurationstabelle genutzt wird, gehören die Dateien *NTUSER.DAT* (Nr. 13), *ntuser.dat.LOG* (Nr. 14), *SAM* (Nr. 20), *SAM.LOG* (Nr. 21), *SECURITY* (Nr. 22), *SECURITY.LOG* (Nr. 23), *SOFTWARE* (Nr. 24), *software.LOG* (Nr. 25), *SYSTEM* (Nr. 26) und *SYSTEM.ALT* (Nr. 27).

Die Dateien ohne Dateiendungen (Nr. 20, 22, 24 und 26) sowie die Datei *NTUSER.DAT* (Nr. 13) bilden laut [MSDN 2002] die Registry. Für die Datei *SYSTEM* gibt es eine zusätzliche Sicherung. Diese wird in der Datei *SYSTEM.ALT* (Nr. 27) gespeichert. Änderungen an diesem Teil der Registry werden zunächst in *SYSTEM* durchgeführt und bei Erfolg in *SYSTEM.ALT* nachgeholt. Zudem werden Änderungen der Registry in den zugehörigen *.LOG*-Dateien (Nr. 14, 21, 23, 25 und 27) protokolliert. Die Art der Einträge in den *.LOG*-Dateien erlaubt in einem eingeschränkten Rahmen das Rückgängigmachen der Änderungen, sofern dies notwendig wird (z.B. wenn nicht alle anstehenden Änderungen wegen eines Stromausfalls durchgeführt werden konnten).

Die Registry ist in einer Baumstruktur organisiert. Im Vergleich zu Festplatten heißen die Verzeichnisse in der Registry direkt unter der Wurzel *Registry Hives* und deren Unterverzeichnisse *Registry Keys* (oder auch Schlüssel). In der Festplatten-Analogie bleibend, werden die Festplattendateien *Values* (Werte), die Inhalte *Data* (Daten) genannt. Unterhalb der Wurzel Arbeitsplatz gibt es laut [MSDN 2002] vier Hives:

[9] Rote Zeilen wurden gelöscht, blaue Zeilen ergänzt. Sofern beides an gleicher Stelle auftritt, handelt es sich um eine Zeilenänderung.

[10] Der Eintrag <blank> dient als Platzhalter, wenn Zeilen gelöscht oder erzeugt nicht aber ersetzt wurden.

- HKEY_CURRENT_CONFIG
  Zu diesem Teil der Registry gehören alle SYSTEM-Dateien. Wie der Name des Schlüssels andeutet, werden in diesem Registry Hive die aktuelle Konfiguration und insbesondere die zu nutzenden Treiber hinterlegt.

- HKEY_CURRENT_USER
  In dem „Current User"-Hive sind die Konfigurationseinstellungen des aktuell angemeldeten Anwenders hinterlegt. Dieser Registry Hive wird in der Datei NTUSER.DAT gespeichert.

- HKEY_USERS
  In diesem Hive befinden sich Konfigurationsinformationen über andere als dem angemeldeten Benutzerkonto. In dem Schlüssel HKEY_USERS\Default befinden sich die Standart-Daten, wenn ein neues Benutzerkonto angelegt werden soll. Die Datei DEFAULT wird dann also als Vorlage für die kennungsspezifische NTUSER.DAT.

- HKEY_LOCAL_MACHINE
  Dieser Hive unterteilt sich laut [MSDN 2002] in vier weitere:
  - SAM
    Der Schlüssel HKEY_LOCAL_MACHINE\SAM beinhaltet Informationen über Benutzerkonten und -gruppen. Sie werden in der Datei SAM gespeichert.
  - Security
    In diesem Unterschlüssel werden sowohl lokale als auch benutzerspezifische Sicherheitseinstellungen gespeichert. Gespeichert werden diese Informationen in der Datei SECURTIY.
  - Software
    Der Software-Schlüssel kann von Programmierern genutzt werden, um Softwarekonfigurationsdaten zu hinterlegen. Die Informationen werden in der Datei SOFTWARE gespeichert.
  - System
    Da die Konfigurationsinformationen sehr wichtig für das Betriebssystem und für Anwendungen sind, werden im diesem Schlüssel diese Informationen abgelegt und auch die älteren Konfigurationen für Wiederherstellungszwecken gesichert. Die SYSTEM-Dateien gehören auch zu diesem Teil der Registry.

Alle Dateien, die im Zusammenhang mit der Registry stehen, werden beim Systemstart gelesen und beim Herunterfahren des Systems zurück geschrieben. Aufgrund dieses Vorgehens ergeben sich Auswirkungen auf das Änderungsdatum der Dateien. Zudem gibt es einige Ereignisse, die in der Registry protokolliert werden. So speichern zum Beispiel manche Shareware-Programme in der Registry, wann sie installiert worden sind oder zum wievielten Male sie aufgerufen wurden. Wegen der Baumstruktur der Registry und weil die Informationen nicht im Klartext gespeichert werden, sind die Dateien bei einer ASCII-Betrachtung (wie in CompareSys) schwer zu verstehen. Selbst die *.LOG*-Dateien wirken kryptisch. Für die Detailanalyse dieser Dateien empfiehlt sich der Einsatz von spezielleren Werkzeugen als CompareSys-Erweiterung.

Die Freeware *Dumphive* von Markus Stephany (vgl. [Stephany 2001]) ist zwar kein solches Werkzeug, ermöglicht aber die Konvertierung der Registry Dateien NTUSER.DAT, SAM, SECURITY, SOFTWARE und SYSTEM in ASCII-Dateien. Unter der Annahme, derart konvertierte ASCII-Dateien sind korrekte und vollständige Abbildungen der Registry Hives, können diese dann einer ASCII-Analyse unterzogen werden:

- NTUSER.DAT
  Neben einigen Änderungen der Einstellungen für den Umgang mit Laufwerken wurde der CleanShutdown-Wert[11] verändert. Dieser Wert zeigt an, ob das Betriebssystem sauber heruntergefahren wurde. Er wird nach einem Systemstart automatisch auf 0 gesetzt, falls das Betriebssystem abstürzt oder nicht sauber heruntergefahren wird. Beim problemlosen Beenden des Betriebssystems wird der Wert auf 1 gesetzt, so dass dies beim nächsten Systemstart erkannt werden kann. Der vorgefundene Wert zeigt somit an, dass das Betriebssystem vor der Erstellung des Images (ursprünglicher Zustand) nicht sauber beendet worden war. Zudem wurde der ViewView2-Wert[12] für die Standard-Ordneransicht (vgl. [Tasch 2003]) und auch der DefaultIcon-Wert[13] geändert, der für das aktuelle Desktop-Symbol für den Papierkorb zuständig ist. Änderungen der UserAssist-Daten[14] sind laut [Tasch 2003] auf die Nutzung von Programmen und die Erstellung von Favoriten zurückzuführen. Des Weiteren fanden sich Änderungen in den PostSetup-Keys[15], die mangels Dokumentation nicht weiter erläutert werden können.

- SAM
  Die Änderungen beschränken sich auf Anpassungen von Hex-Daten für die F-Werte aller Benutzerkennungen[16] und speziell für den Administrator[17] und Besucher der Web-Site IUSR_WWW3[18]. Sie beschränken sich auf den Bereich der Passwortspeicherung.

- SECURITY
  Die konvertierte Datei zeigte keine Dateiinhaltsänderungen mehr auf. Dies verwundert, da der Bytevergleich der unkonvertierten SECURITY Dateien Unterschiede feststellte. Dieses Resultat lässt sich nur dadurch erklären, dass die Konvertierung der Datei nicht korrekt und vollständig erfolgte.

- SOFTWARE
  In diesem Registry Teil konnten neun Eintragsänderungen festgestellt werden. Darunter sind vier Zeiteinträge (TimeStamps[19]) des Synchronisationsmanagers (SyncMgr), sowie die

[11] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer
[12] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ Streams\ Desktop
[13] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\CLSID\ {645FF040-5081-101B-9F08-00AA002F954E}
[14] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Windows\ CurrentVersion\Explorer\UserAssist
[15] in den Schlüsseln HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ Discardable\PostSetup*\Enum
[16] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account
[17] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000001F4
[18] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000003E8
[19] in den Schlüsseln SOFTWARE\Microsoft\Windows\CurrentVersion\Syncmgr\AutoSync*

Festhaltung, wann das Administrator-Profil[20] zuletzt geladen wurde. Des Weiteren wurde eine UUID-Sequenznummer[21] (Universally Unique Identifier) für das Remote Procedure Call (RPC) Protokoll geändert. Die UUID wird verwendet, um bei entfernten Prozeduraufrufen den Aufrufer zu identifizieren (vgl. [MS RPC 2003]). Auch der Seed-Wert[22] des Random Number Generators (RNG), der Zufallszahlen für Microsofts Windows Kryptographie-Komponente erzeugt, wurde erneuert.

- SYSTEM
  Insgesamt 47 Ergänzungen und Änderungen des Registry-Teils SYSTEM konnten nach der Konvertierung ausgemacht werden. Neben der Speicherung von Anmelde- (LogonTime[23]) und Herunterfahr-Zeiten (ShutDownTime[24]) wurden entsprechend der Datei *setupapi.log* (Nr. 18) Konfigurationsänderungen[25] festgestellt. Da Windows die alten Informationen nicht sofort löscht, sondern in einer Registry-internen Sicherung beibehält, ergibt sich die hohe Anzahl von Änderungen.

Die Datei *setupapi.log* dient laut [Smith 2001: 1] dazu, die Installation von Geräten und Treibern zu protokollieren. Werden neue Geräte oder Treiber installiert, wird das Protokoll entsprechend erweitert. Die festgestellten neuen Einträge in der Protokolldatei weisen daraufhin, dass sich die Konfiguration des IT-Systems geändert hat: Windows 2000 Professional installierte beim Systemstart einen Treiber für den Monitor und erkannten einen anderen „Datenträger“ und „Standarddatenträger“. Als Ursache für Datei- und Konfigurationsänderungen bzgl. der Datenträger konnte ausgemacht werden, dass das Betriebssystem auf einer anderen Festplatte als der im weiteren Versuchsverlauf Genutzten installiert worden war. Die Änderung des Monitortreibers kann nicht nachvollzogen werden, da stets derselbe Monitor mit der bis auf die Festplatten unveränderten IT-System-Konfiguration genutzt wurde.

Neben der Registry zählt zu den Konfigurationsdateien laut [ComTechDoc 2001] die Datei *ntuser.ini* (Nr. 10). Sie wird für jedes Benutzerkonto auf Windows-Systemen angelegt und kann Informationen über das Roaming Profil und ähnliches enthalten. Entsprechend der Registry Dateien wird sie bei Anmeldung eines Nutzers geladen und beim Abmelden des Nutzers wieder gespeichert, auch wenn wie in diesem Fall der Dateiinhalt unverändert blieb. Das Speichern führt zu einer Anpassung des Änderungsdatums und das Setzen des Archivierungsbits der Attribute.

Laut [MySQL 2003: 5.9.1] werden in der MySQL-Protokolldatei *mysql.err* (Nr. 15 in der Anlage C) sowohl das Starten als auch das Beenden des MySQL-Servers, sowie sämtliche Fehlermeldungen dokumentiert, die während der Laufzeit des Servers auftreten. Die Einträge werden an die schon bestehenden Meldungen angehängt. In dieser Versuchsreihe wurde dokumentiert, dass der MySQL-Dienst gestartet und später wieder beendet wurde und wie der Einsatz von MySQL effizienter gestaltet werden könnte (s. Abb. 43).

---

[20] im Schlüssel HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\S-1-5-21-1645522239-436374069-1343024091-500

[21] im Schlüssel HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\RPC

[22] im Schlüssel HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\RNG

[23] im Schlüssel HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Print\Providers

[24] im Schlüssel HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Windows

[25] in diversen Schlüsseln

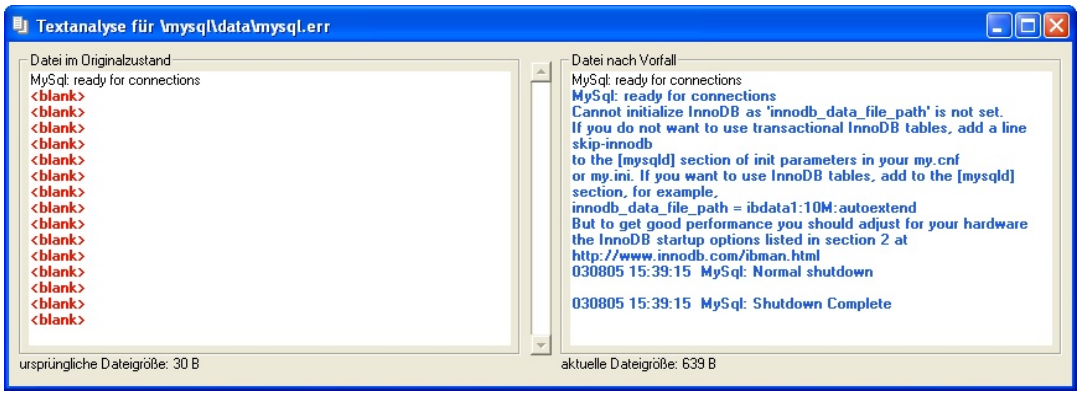


Abb. 43: CompareSys-Darstellung für die Inhaltsänderungen der Datei mysql.err

Für das Protokollieren des Startens und Beendens des Taskplaners wird laut Microsoft (vgl. [MS KB 303204]) die Datei *SchedLgU.Txt* (Nr. 17 in der Anlage C) genutzt. Der Taskplaner erlaubt das Planen von regelmäßig wiederkehrenden Aufgaben. Zum entsprechend eingestellten Zeitpunkt führt der Taskplaner automatisch und ohne weitere Anwenderinteraktionen die Aufgaben durch. Dazu wird der Taskplaner-Dienst genutzt. Der Start und das Beenden dieses Dienstes werden in *SchedLgU.Txt* protokolliert (s. Abb. 44). Änderungen an dieser Datei, sind also durch das Hoch- und Herunterfahren des Systems begründet, denn der Dienst soll gemäß den Einstellungen immer mit dem Betriebssystem zur Verfügung stehen.

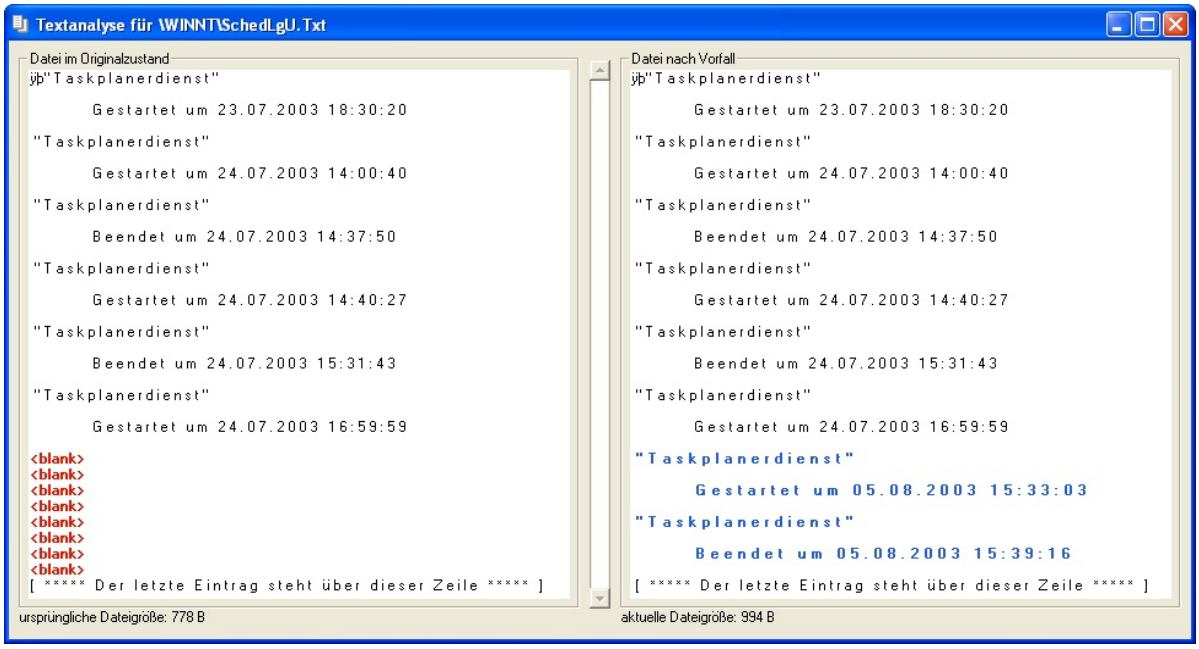


Abb. 44: CompareSys-Darstellung für die Inhaltsänderungen der Datei SchedLgU.Txt

Die Datei *SA.DAT* (Nr. 32) scheint aufgrund des Speicherungsortes (c:\WINNT\Tasks) ebenfalls im Zusammenhang mit dem Microsoft Taskplaner zu stehen. Die genaue Aufgabe konnte nicht geklärt werden. Die Datei wird weder im Klartext geschrieben, noch existieren öffentlich zugängliche Dokumentationen bezüglich dieser Datei. Die Microsoft Deutschland GmbH war auf Anfrage auch zu keiner Aussage über die Funktion und Nutzung dieser Datei bereit.

Um im Windows Explorer kleine Grafiken, so genannte Icons, schnell zur Verfügung zu stellen, hat laut [ShellIconCache 2003] Microsoft einen Shell Icon Cache implementiert. In die-

sem Zwischenspeicher werden die Icons gespeichert, die per Index adressiert werden können. Hinter diesem Cache verbirgt sich die Datei *ShellIconCache* (Nr. 19 in der Anlage C), die beim Systemstart geladen und beim Herunterfahren wieder geschrieben wird. Änderungen an dieser Datei beziehen sich also nur auf den Cache. Exploits für diese Datei sind nicht bekannt, weshalb die hier festgestellten Änderungen unter Umständen vernachlässigt werden können. Problematisch wird es, wenn die Datei bei der Analyse ausgeschlossen wird, aber ein Angreifer eine Schwachstelle in diesem Konzept entdeckt und ausnutzt.

Interessanter für die Ausnutzung durch Hacker ist da schon die Datei *MetaBase.bin* (Nr. 28). Darin werden laut [MS IIS 2001] die meisten Einstellungen für den IIS gespeichert. Ohne diese Einstellungen bzw. ohne diese Datei funktioniert der IIS nicht, womit sie ein Angriffsziel auf die Verfügbarkeit des IIS darstellt. Trotz dieser Relevanz der Datei konnten die vorgenommenen Änderungen des Dateiinhaltes nicht im Detail nachvollzogen werden, da der Dateiinhalt zum einen nicht gut genug dokumentiert ist und zum anderen dieser nicht im Klartext gespeichert wird, worauf auch schon die Dateiendung hinweist. Ebenfalls zu dem IIS gehören normalerweise die Dateien im Verzeichnis WINNT\system32\LogFiles\W3SVC1\. Die dort abgelegten *.log*-Dateien tragen als Namen eine Kombination aus „*ex*“ und dem aktuellen Systemdatum. Der Dateiname deutet schon darauf hin, dass für jeden Tag eine eigene Log-Datei (wie *ex030805.log*, Nr. 34 in Anlage C) angelegt wird. In diesen Dateien werden je nach Konfiguration des IIS sämtliche empfangene HTTP-Befehle dokumentiert. Für diese Versuchsreihe wurden Übermittlungsaufrufe von Dateien der Web-Site per HTTP-GET durch Systeme mit verschiedenen IP-Adressen zu unterschiedlichen Zeiten protokolliert (s. Abb. 45).

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-08-05 13:34:02
#Fields: time c-ip cs-method cs-uri-stem sc-status
13:34:02 192.168.0.22 GET /index.html 304
13:34:02 192.168.0.22 GET /title.html 304
13:34:02 192.168.0.22 GET /alarm.html 304
13:34:02 192.168.0.22 GET /navigation.html 304
13:34:02 192.168.0.22 GET /kontakt.html 304
13:34:02 192.168.0.22 GET /images/alarm.gif 304
13:34:02 192.168.0.22 GET /images/title.gif 304
13:34:02 192.168.0.22 GET /images/button2.gif 304
13:34:02 192.168.0.22 GET /images/button1.gif 304
13:34:02 192.168.0.22 GET /images/button5.gif 304
13:34:02 192.168.0.22 GET /images/button3.gif 304
13:34:02 192.168.0.22 GET /images/button4.gif 304
13:34:05 192.168.0.22 GET /agb.html 304
13:34:07 192.168.0.22 GET /kontakt.html 304
13:34:10 192.168.0.22 GET /agb.html 304
…
13:35:35 192.168.0.23 GET /index.html 304
13:35:35 192.168.0.23 GET /alarm.html 304
13:35:35 192.168.0.23 GET /title.html 304
…
```

Abb. 45: Auszug aus der Datei ex030805.log

Der Removable Storage Manager (RSM) dient in Windows 2000 Systemen dazu, um entfernbare Speicher in das Betriebssystem einzubinden. Dazu nutzt der RSM eine Datenbank, die standardmäßig im Verzeichnis WINNT\system32\NtmsData\NTMSDATA abgelegt wird (vgl. [MS RSM 2002]). Die Datenbank wird dabei in der Datei *NTMSDATA* (Nr. 29 in Anlage C) gespeichert und in der Datei *NTMSDATA.bak* (Nr. 30) zusätzlich gesichert. Die Datei NTMSIDX

(Nr. 31) stellt dabei eine zusätzliche Index-Datei dar, über die Einträge in der Datenbank über einen anderen als dem in der Datenbank verwendeten Index gefunden werden können. Wieso die festgestellten Änderungen vorgenommen wurden, konnte aufgrund unzureichender Dokumentation nicht definitiv festgestellt werden.

Im Windows-Installationsverzeichnis WINNT\inf wurde die neue Datei *7-zip.PNF* (Nr. 33 in Anlage C) vorgefunden. Hierbei handelt es sich um eine so genannte „vorkompilierte Setup-Datei", die laut [Grigsby 1999: 3.4] zukünftige Instanzen unterstützt, wenn die Treiberinstallationsdatei ausgeführt wird. Die vorgefundenen 7-zip.PNF gehört zu der OpenSource Freeware *7-Zip*, mit der Dateien unter Verwendung von Komprimierungsalgorithmen archiviert werden können. Dass diese Datei nicht schon auf dem ursprünglichen Festplattenzustand zu finden ist, deutet darauf hin, dass die Installation noch nicht vollständig abgeschlossen war, als die erste Sicherung des Systemzustandes erstellt wurde.

Die 32 festgestellten Änderungen und die zwei neuen Dateien beziehen sich also auf eine ganze Reihe von Betriebssystem- und Anwendungsprogrammen. Viele der erwähnten Dateien lassen sich entweder in die Klasse der Protokolle oder der Konfigurationsdateien einsortieren. Änderungen an den Konfigurationsdateien geben bei der Untersuchung von Computersicherheitsvorfällen zum Beispiel Hinweise darauf, welche Ziele ein Angreifer verfolgt. Aber auch Protokolldateien sind bei Vorfallsanalysen sehr wichtig, da sie Informationen über die Abläufe geben (vgl. Log-File Analyse im Abschnitt 2.2). Die Protokolldateien können also dabei helfen, ein besseres und vollständigeres Verständnis über den Vorfallsablauf zu erlangen.

4.3 Defacement und Platzierung einer Angriffssoftware

Der zweite Penetrationstest (vlg. [Michaelsen 2003: 6]) bestand darin, dass die so genannte „URL Decoding"-Schwäche des IIS (vgl. [ISS 2000645]) ausgenutzt wurde. Diese Schwäche erlaubt laut der Microsoft Knowledgebase „einem böswilligen Benutzer, […] Betriebssystembefehle auf einem davon betroffenen Web-Server auszuführen" (aus [MS KB 295534]).

Bei diesem Versuch wurde die Schwäche ausgenutzt, um die Start-Seite der Web-Site auszutauschen (so genanntes *Defacement*, vgl. Abb. 39 mit Abb. 46). Des Weiteren wurde die Open-Source Software *Back Orifice 2000* (BO2k, vgl. [Back Orifice 2000]) installiert. Die Software erlaubt das Steuern des Systems, auf dem sie installiert wurde. BO2k wird häufig von Hackern verwendet, um illegalen bzw. illegitimen Zugriff auf IT-Systeme zu erlangen. Deshalb wird BO2k von vielen AntiMalware-Herstellern in den Enzyklopädien als Trojaner bezeichnet (vgl. z.B. [Podrezov 2003], [McAfee 1999] und [Elnitiarta & Han 1999]), auch wenn BO2k der hier verwendeten Definition von Malware (s. 1.3.2) nicht entspricht.

Abb. 46: Homepage der Firma Mayer nach dem Defacement

Bei der Analyse des derart angegriffenen Systems konnten 35 Dateiänderungen und vier neue Dateien mit CompareSys identifiziert werden (s. Anlage D), wobei keine Datei des ursprünglichen Festplattenzustandes dauerhaft gelöscht worden ist. Es wurden also nur neue Dateien und Dateiänderungen entdeckt. Der verwendete komparativ statische Ansatz schließt allerdings aus, dass erkannt wird, ob Dateien zwischen den beiden betrachteten Zuständen erzeugt und wieder gelöscht worden sind.

Unter den 35 Änderungen befinden sich zwölf Dateien (Nummer 1 bis 12 in Anlage D), bei denen sich nur das Änderungsdatum, nicht aber der Inhalt oder eine sonstige Eigenschaft geändert hat. Dies ist ein deutlicher Hinweis darauf, dass die Dateien geladen und wieder geschrieben worden sind. Da sich aber der Inhalt der Dateien nicht geändert hat, bestehen zwei Interpretationsmöglichkeiten:

1) Die Dateien wurden vom Betriebssystem und Applikationen geöffnet, um Änderungen direkt speichern zu können. Beim Herunterfahren des Systems werden diese Dateien dann spätestens zurück geschrieben, auch unter dem besonderen Umstand, dass sich die Dateiinhalte nicht geändert haben. Dies ist zum Beispiel bei den Dateien des LSASS (Nr. 6 bis 10) und bei den Ereignis-Protokolldateien (Nr. 11 und 12) der Fall, die beim Systemstart immer gleich mit Schreibzugriff geladen werden.

2) Von den Inhalten der Dateien wurden Sicherheitskopien während des Vorfalls angelegt oder ein Hacker besitzt die Kompetenz, Änderungen an diesen Dateien, die durch seinen Angriff verursacht werden, zu identifizieren. Nach dem eigentlichen Angriff hat der Hacker zur Verschleierung der Spuren die veränderten Dateiinhalte durch die ursprünglichen Informationen wieder ersetzt bzw. die Spuren seines Angriffs gelöscht. Mit einem derartigen Vorgehen könnte eine unveränderte Protokoll-Datei vorgetäuscht werden.

Da der Angriff über ein Netzwerk durchgeführt wurde, könnten Netzsniffer und lokale, dynamische Monitore bei der Aufklärung dieser Änderungen helfen. Liegen wie in diesem Fall keine Monitorprotokolle vor, dann muss aufgrund der Ergebnisse aus der Versuchsreihe über den ordentlichen Gebrauch (s. 4.2) geschlossen werden, welcher der beiden Fälle für die Dateien jeweils vorliegt. Die Ergebnisse zeigen in diesem Fall, dass die Änderungen der Dateien (Nr. 6 bis 12) mit hoher Wahrscheinlichkeit nicht auf den Vorfall zurückzuführen sind. Die Analyse der

Dateien aus dem Verzeichnis für das Administrator-Benutzerkonto (Nr. 1 bis 3) weist nach, dass zwischen den Zeitpunkten, von denen die Festplattenzustände stammen, auf dem System eine Anmeldung mit dem Benutzerkonto erfolgte. Bei dieser Anmeldung wurden die *index.dat*-Dateien geöffnet und beim Abmelden wieder geschrieben.

Ein zusätzlicher Vergleich des späteren Zustandes der Untersuchung aus Abschnitt 4.2 mit dem Zustand nach der Ausnutzung der „URL Decoding"-Schwäche des IIS ergab: Die Dateien des CSC (Nr. 4, 5 und 20 in Anlage D) wurden übereinstimmend mit den Ergebnissen der ersten Versuchsreihe geändert. Dies gilt ebenfalls für die Protokolldateien der ATI-Treiber (Nr. 14 und 15), des Taskplaners (Nr. 21 und 35), des MySQL-Servers (Nr. 19), des Geräte- und Treiberinstallationsprogrammes (Nr. 22) sowie für die Datenbank- und -indexdateien des RMS (Nr. 32 bis 34) als auch für die Konfigurationsdateien *ntuser.ini* (Nr. 13) und *MetaBase.bin* (Nr. 31). Des Weiteren wurde wieder die ergänzte, vorkompilierte Setup-Datei *7-zip.PNF* (Nr. 37) vorgefunden.

Zu den so nicht erklärbaren Dateiveränderungen gehört die Homepage *index.html* (Nr. 18) der Firma Alarm Mayer. Die Analyse der Dateieigenschaften lässt zwar vermuten, dass der Inhalt, nicht aber die Datei ausgetauscht wurde, da das Erstellungsdatum noch übereinstimmt; dies ist aber ein Trugschluss, wie die Beschreibungen in [Michaelsen 2003: 6] darstellen: denn die Datei wurde durch eine andere ersetzt. Dies erfolgte durch die Nutzung der Übertragungssoftware *tftp*, die von Microsoft mit Windows 2000 Professional ausgeliefert wird. Aufgrund dieser Erkenntnisse wird es offensichtlich, dass schon die Standardtreiber und -programme von Windows 2000 Professional nicht der FAT-Spezifikation entsprechen, da das Erstellungsdatum hätte geändert werden müssen. Diese Erkenntnis hat insbesondere Konsequenzen für folgende Vorfallsuntersuchungen und Ergebnisinterpretationen. Der festgestellte, geänderte Inhalt der Datei (s. Abb. 47) stellt das Defacement dar.

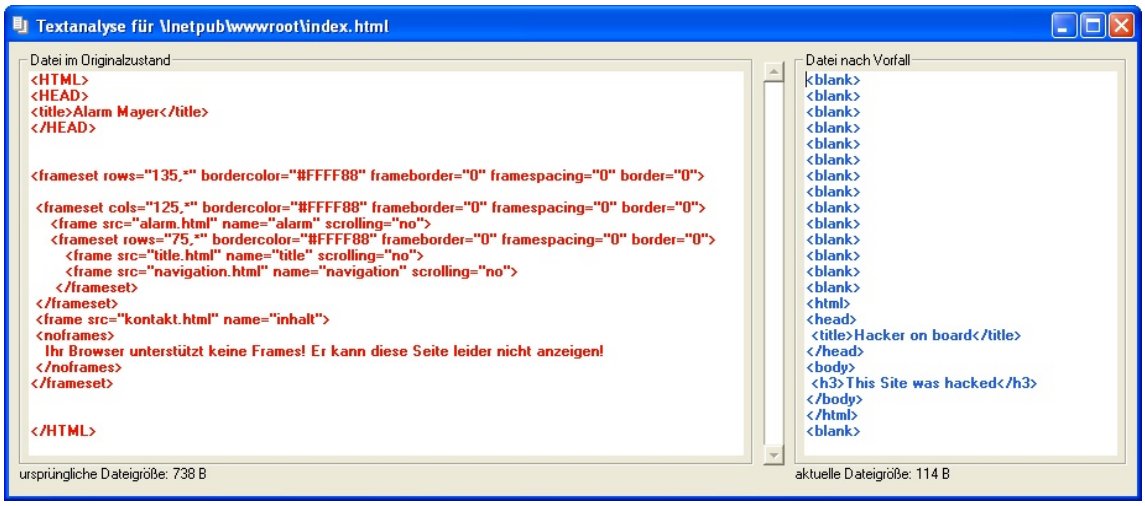


Abb. 47: CompareSys-Darstellung für die Inhaltsänderungen der Datei index.html

Neben den bisher erwähnten Dateien, befinden sich vier neue Dateien auf dem System (Nr. 36 bis 39 in der Anlage D). Dazu gehört aus denselben Gründen wieder die Datei *7-zip.PNF*. Zu den anderen drei neuen Dateien gehören die im Root-Verzeichnis befindliche *bo2k.exe* und die im System32-Verzeichnis des Betriebssystems gespeicherte *UMGR32.exe*. Beide gehören typischerweise zu Back Orifice 2000. Entsprechend der BO2k-Beschreibung von Alexey Podrezov (vgl. [Podrezov 2003]) müsste je nach Konfiguration des BO2k-Servers ein Registry Wert mit Daten in HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

hinterlegt werden. Der Registry Key wird erweitert, damit der BO2k-Server automatisch mit dem Systemstarten zur Verfügung steht. Die vorgefundenen Werte des Registry Keys wiesen allerdings diesen Eintrag nicht auf. Neben dem Defacement der Homepage der Firma Alarm Mayer wurde also auch die „Fernwartungssoftware“ *Back Orifice 2000* installiert, deren Dienst von dem Hacker aufgrund unvollständiger Installation allerdings nach jedem Server-Reboot wieder gestartet werden muss.

Ein Blick in die neue Zugriffsprotokoll-Datei *ex030729.log* des IIS (Nr. 38 in der Anlage D, s. Abb. 48) verrät den Weg, der genutzt wurde, um die Dateien auf dem Server zu platzieren. Diverse Einträge der Art „GET /Scripts/..%5c../winnt/system32/cmd.exe“ deuten darauf hin, dass der Angreifer die „URL Decoding“-Schwäche (vgl. [ISS 2000645]) ausgenutzt hat, um die Kontrolle über das System zu erlangen.

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2003-07-29 09:00:49
#Fields: time c-ip cs-method cs-uri-stem sc-status
…
09:05:01 192.168.0.18 GET /scripts/..%5c../winnt/system32/cmd.exe 502
09:05:47 192.168.0.18 GET /scripts/..%5c../winnt/system32/cmd.exe 502
09:06:07 192.168.0.18 GET /scripts/..%5c../winnt/system32/cmd.exe 502
…
```

Abb. 48: Auszug aus der Datei ex030729.log

Die bisherigen Erkenntnisse erfordern eigentlich die detaillierte Analyse der Registry und der dazugehörigen Dateien (Nr. 16, 17 und 23 bis 30). Eine derartige Untersuchung ist aufgrund des Dateiaufbaus und der in CompareSys zur Verfügung stehenden Mittel nur mit einem Zusatzmodul möglich, das aber nicht Bestandteil der Betrachtungen dieser Diplomarbeit ist.

Um dennoch einen Eindruck über die Änderungen der Registry zu bekommen, wurden die Registry Dateien wieder in ASCII-Dateien konvertiert:

- NTUSER.DAT
  In diesem Teil der Registry wurde die Veränderung der Windows Explorer[26] und der Shell Explorer[27] Toolbars sowie der Veränderung von Fenstergrößen und -positionen[28] dokumentiert (vgl. [MS KB 171002]). Diese Änderungen sind nicht Teil des Vorfalls, sondern können auf Handlungen eines Administrators zurückgeführt werden. Zusätzlich wurden zwei neue Laufwerke (E: und F:) festgestellt[29] (vgl. [MS KB 235994]), die aufgrund einer im zweiten Wechselrahmen verbliebenen Festplatte erklärt werden können. Außerdem konnte eine Änderung der Proxy-Konfiguration[30] für den Internet Explorer ausgemacht werden, die aber nicht durch den Angriff erklärbar ist.

[26] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Toolbar\Explorer

[27] im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Toolbar\ShellBrowser

[28] in den Schlüsseln HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\StreamMRU und HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Streams

[29] in den Schlüsseln HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints\ E* bzw. …\F*

[30] SavedLegacySettings im Schlüssel HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections

- SAM, SOFTWARE und SYSTEM
  In den Registry Dateien SAM und SOFTWARE wurden nur Registry Keys verändert, die auch in der Versuchsreihe über den normalen Gebrauch modifiziert worden waren. Bei SYSTEM wurde zusätzlich der Wert LsaPid[31] modifiziert, der im Zusammenhang mit der Local Security Authority steht, ansonsten aber nicht näher von Microsoft und anderen Quellen dokumentiert wird.

- SECURITY
  Wieder wurde festgestellt, dass der Bytevergleich der SECURITY-Dateien Unterschiede erkannte, diese nach der Konvertierung aber nicht mehr zu ermitteln waren.

Aufgrund der festgestellten Änderungen liegen je nach aufgestellten Sicherheitsanforderungen Vorfälle im Bereich der Integrität der Festplatten-Daten für den Web-Server und des Betriebssystems vor. Da die Web-Site betroffen ist, liegt je nach Anforderungen auch ein Verfügbarkeitsvorfall vor. Durch die Entdeckung von *Back Orifice 2000* besteht auch die Möglichkeit von Unterschreitungen der Vertraulichkeitsanforderungen auf Betriebssystem-Ebene, da diese Software Benutzerkennungen und Passwörter auslesen kann.

4.4 Schaffung einer Benutzerkennung und Defacement

Für den dritten Versuch nutzte Michaelsen im Rahmen der Penetrationstests (vgl. [Michaelsen 2003: 6]) eine Schwäche aus, die laut [MS03-007] aufgrund eines unkontrollierten Puffers in der Programmbibliothek *ntdll.dll* existiert. Laut Microsoft erlaubt diese Schwäche einem Hacker, beliebige Software auf dem Opfersystem auszuführen. Michaelsen nutze die Schwäche, um zunächst eine neue Benutzerkennung zu erstellen und sie in die Administratoren-Gruppe aufzunehmen. Nach intensiver Sichtung der Verzeichnisse führte er dann wieder ein Defacement der Web-Site durch (vgl. Abb. 49 mit Abb. 39), installierte die *Jesper NT Tools* von Jesper Lauritsen (vgl. [Lauritsen 1999]) und kopierte und löschte Log-Dateien. Insgesamt führten diese Aktionen zu 52 Dateiänderungen, fünf Dateilöschungen und 93 neuen Dateien, die CompareSys feststellte (vgl. Anlage E).

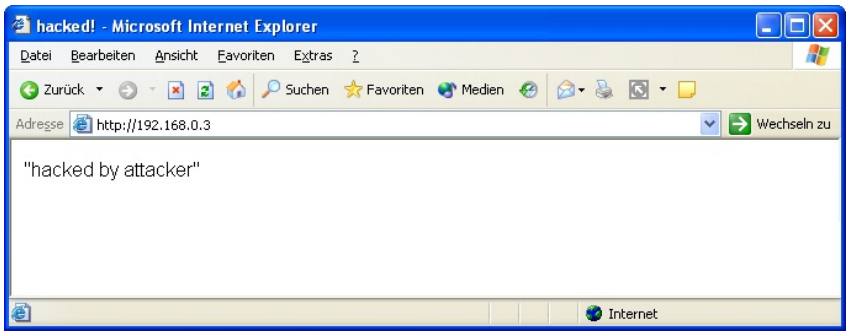


Abb. 49: zweites Defacement der Web-Site

Unter diesen Änderungen finden sich wieder die bereits erwähnten und hier unproblematischen Dateien des CSC (Nr. 8 in der Anlage E), der *ShellIconCache* (Nr. 34), die Protokolldateien der ATI-Treiber (Nr. 25 und 26), des Taskplaners (Nr. 31 und wohl auch 19), des MySQL-Servers (Nr. 23), des Geräte- und Treiberinstallationsprogrammes (Nr. 33) sowie die Datenbank-

[31] gespeichert im LsaPid des Schlüssels HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa

und -indexdateien des RSM (Nr. 48, 49 und 16) als auch die Konfigurationsdateien *ntuser.ini* (Nr. 113) und *MetaBase.bin* (Nr. 15). Unverändert leer sind wieder die LSASS-Protokoll-Dateien (Nr. 9 bis 13). Auch die Log-Datei des Web-Servers (Nr. 47) zeigt nur übliche HTTP-Befehle, wie sie in Abschnitt 4.2 festgestellt wurden. Alte Log-Dateien wurde einstellungskonform gelöscht (Nr. 53 bis 57).

In der Liste der Änderungen werden einige Dateien geführt, die im Zusammenhang mit dem Windows Media Player stehen. Die Dateien *Windows Media Player.lnk* (Nr. 22 in der Anlage E), *mplayer2.inf* (Nr. 28) und *mplayer2.PNF* (Nr. 29) wurden aufgrund der Installation des Service Packs 3 erzeugt. Wie auch bei der Datei 7-Zip.pnf in den vorangegangenen Versuchen scheint die Installation hier ebenfalls noch nicht abgeschlossen gewesen zu sein, als das Image erzeugt wurde.

Neben den bereits erwähnten und wieder festgestellten Änderungen können die ersten vier Einträge in der Anlage E (Nr. 1 bis 4) dadurch erklärt werden, dass das Betriebssystem mit der Benutzerkennung des Administrators gestartet wurde (s. Abschnitt 4.3). Die namensgleichen Dateien in den Verzeichnissen des Benutzerkontos „Default User“ (Nr. 5 bis 7) können allerdings nicht mit einer „Default User“-Anmeldung begründet werden. Dieses Benutzerkonto wird von Windows 2000 Professional nur verwendet, um Grundeinstellungen für neue Benutzerkonten bereitzustellen. Das Laden und unveränderte Speichern dieser Dateien deutet daraufhin, dass ein neues Benutzerkonto erstellt wurde und zu diesem Zwecke die Informationen geladen und unverändert zurück geschrieben wurden. Ein weiteres Indiz dafür ist, dass das Betriebssystem genauso mit der Datei *DEFAULT* im Verzeichnis der Registry Dateien (Nr. 36) verfahren ist. Diese Datei dient dazu, Standard-Einstellungen für den benutzerkontenbezogenen Registry Hive HKEY_CURRENT_USER (gespeichert in *NTUSER.DAT,* Nr. 109) bereitzustellen, wenn neue Kennungen eingerichtet werden. Bestätigt wird die Vermutung, dass ein neues Benutzerkonto eingerichtet wurde, durch die Existenz diverser Dateien (Nr. 58 bis 142). Alle diese Dateien liegen im Verzeichnis des neuen Benutzers „hacker“ (und dessen Unterverzeichnissen). Für diese Kennung wurde der Microsoft Web-Browser *Internet Explorer* (Nr. 30, 58, 59, 63, 68 bis 96 und 102), diverse Komponenten des Betriebssystems wie die Schnellstartleiste (Nr. 60 bis 62), der *Windows Explorer* (Nr. 113 bis 116 und 131 bis 142) und das Startmenü (Nr. 117 bis 130) vorkonfiguriert. Auch die *ntuser.ini* (Nr. 111) und die *ntuser.dat.log* (Nr. 110) wurden erstellt. Des Weiteren werden noch Verzeichniseinstellungen in mehreren *desktop.ini* Dateien (Nr. 66, 67, 99 bis 101, 103 bis 107 und 112) und Beispieldateien (Nr. 64, 65, 97 und 98) erzeugt.

```
USERENV(374.224) 15:19:07:112 RegisterGPNotification: CreateEvent failed with 5
USERENV(374.224) 15:19:07:122 RegisterGPNotification: CreateEvent failed with 5
USERENV(374.224) 15:19:07:122 RegisterGPNotification: CreateEvent failed with 5
```

Abb. 50: neue Zeilen in der Datei userenv.log

Zusätzlich zu der Kennungseinrichtung ist es dem Angreifer auch gelungen, im Verzeichnis WINNT\system32\Microsoft\Crypto\ntt\ die Jesper NT Tools (Nr. 144 bis 150) zu installieren, die diverse Funktionen zum Eingriff in das Betriebssystem beinhalten, wie zum Beispiel das Lesen, Sichern und Löschen der Eventlog-Dateien. Die Bearbeitung der Dateien *AppEvent.Evt* (Nr. 35) und *SysEvent.Evt* (Nr. 44) kann auch festgestellt werden. Hier wurden offensichtlich Einträge gelöscht und damit Spuren verwischt. Die Fehlermeldungen (s. Abb. 50) in der Datei

userenv.log (Nr. 27) könnten in diesem Zusammenhang aufgetreten sein, da das Betriebssystem dort vermerkt hat, dass ein Ereignis nicht protokolliert werden konnte.

Die Datei {*02D4B3F1-FD88-11D1-960D-00805FC79235*}.*crmlog* (Nr. 14 in der Anlage E) ist neu in der Reihe der unverändert gespeicherten Protokolldateien. Sie wurde mit demselben Änderungszeitpunkt wie die Dateien *$WinMgmt.CFG* (Nr. 17), *CIM.REP* (Nr. 18) und *CIM.REC* (Nr. 52) versehen, wenngleich auch nur die letzte inhaltlich verändert wurde. Beide CIM-Dateien gehören zu dem Microsoft *Common Information Model* (CIM), das als Teil der CIM-Erweiterung zu der *Windows Management Instrumentation* (WMI) erhalten geblieben ist. Laut [MS WMI 1999] dient WMI der Administration von Windows 2000 Professional und ist eine Implementation der Initiative *Web-Based Enterprise Management* (WBEM) der Desktop Management Force (DMTF). WMI bietet zum Beispiel durch die *Microsoft Management Console* (MMC, vgl. [MS MMC 1998]) Zugriff unter anderem auf die Ereignisanzeigen, die Ordnerfreigaben, die lokalen Benutzerkonten sowie auf die Datenträger- und auch auf die Diensteverwaltung des IT-Systems. Dazu greift WMI offensichtlich auf den *Compensation Resource Manager* (CRM) der *Component Services* (COM+) zurück (vgl. [MS COM+ CRM 2003]). Da der CRM-Dienst für jede Anwendung, von der er genutzt wird, eine eigene .crmlog-Datei erstellt, trägt der Dateiname in den geschweiften Klammern die Identifikationsnummer der Anwendung. Die vorgefundene .crmlog-Datei wurde also zu Protokollzwecken geöffnet und unverändert wieder gespeichert. Im Gegensatz dazu stehen in den WMI-Dateien *wbemcore.log* (Nr. 50) und *WinMgmt.log* (Nr. 51) Hinweise, dass der Kern des WMI heruntergefahren wurde und folglich zuvor gestartet wurde. Da niemand die MMC an dem Opfersystem direkt genutzt hat, ergaben die Untersuchungen einen Fernzugriff auf die MMC. Aufgrund der zentralen Verwaltungsaufgaben, die über die MMC in Verbindung mit WMI genutzt werden können, muss vermutet werden, dass der Hacker sich einen guten Überblick über das IT-System verschaffte, aber auch die Möglichkeit hatte, in die Konfiguration einzugreifen.

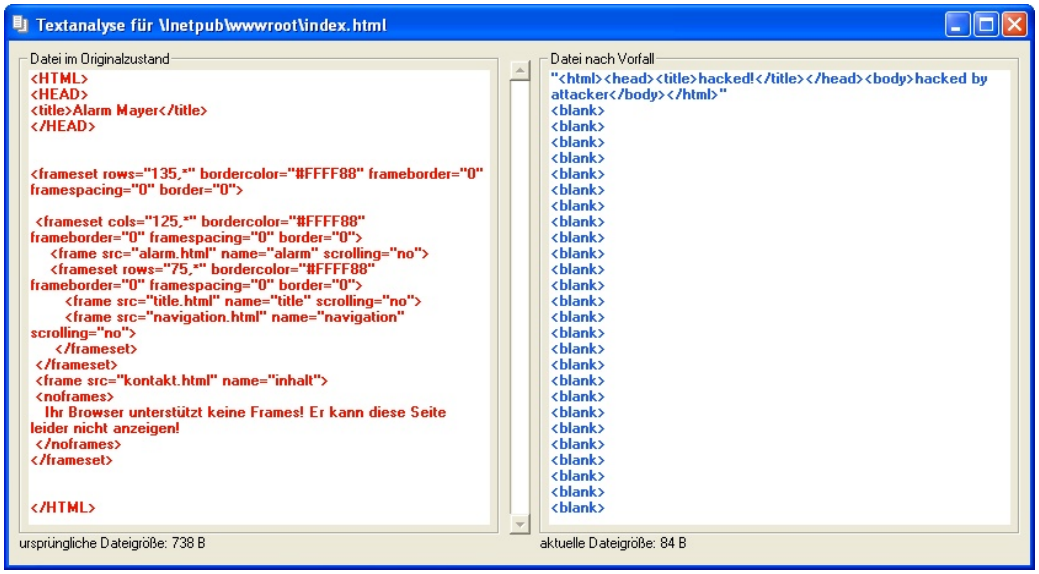


Abb. 51: CompareSys-Darstellung für die Inhaltsänderungen der Datei index.html

Es lassen sich auch Veränderungen der Homepage *index.html* (Nr. 24 in der Anlage E, s. Abb. 51) feststellen, die zu einem Defacement (vgl. mit Abb. 49) geführt haben. Aufgrund des geänderten Erstellungsdatums kann angenommen werden, dass die ursprüngliche Datei durch eine neue Datei ersetzt wurde. Unter Berücksichtigung der neuen Datei *index.orig* (Nr. 143) wird das Vorgehen des Hackers deutlich: der Angreifer änderte die Dateiendung der ursprünglichen

Datei *index.html* in *.orig* um, denn sowohl Inhalt als auch die anderen Dateieigenschaften sind identisch mit der ursprünglichen *index.html* (Änderungen der Dateiendung haben keine Auswirkungen auf das Änderungsdatum der Datei). Danach wurde eine neue Datei mit anderem Inhalt unter dem Namen *index.html* erstellt.

Dem Hacker ist es zusammenfassend also gelungen, eine Kennung einzurichten, Protokolldateien zu verfälschen und ein Defacement durchzuführen. Anzunehmen ist auch, dass er sich mittels MMC einen guten Überblick über das IT-System verschaffen konnte.

Unklar sind noch die Einträge in der Datei *scepol.log* (Nr. 32 in der Anlage E). Laut [MS KB 277675] wird die Datei dafür genutzt, die Vorgänge der Local Security Authority zu protokollieren. Jedoch wird nicht erklärt, was die vorgefundenen Einträge („Initialize NotificationQSync", „Entered NotificationQSync for unflushing queue", „Leaving NotificationQSync for unflushing queue", „Unflush Notification Queue") bedeuten. Sie könnten aber in Zusammenhang mit dem Leeren der Anwendungs- und Systemprotokolle stehen.

Wie bei der letzten Vorfallsanalyse aus Abschnitt 4.3 erfordern die bisherigen Erkenntnisse eine detaillierte Analyse der Registry Hives und der zugehörigen Dateien (Nr. 20, 21, 36 bis 43 sowie 45, 46, 109 und 110). Eine derartige Untersuchung lässt sich derzeit nur mit den durch dumphive konvertierten und dabei eventuell veränderten Registry Hives durchführen:

- SAM
  In dem Registry Hive SAM kann festgestellt werden, dass neben den bereits bekannten Änderungen eine Benutzerkennung 000003EA[32] (Alias Hacker[33]) angelegt und in die Gruppen 00000220[34] (Administratoren) und 00000221[35] (Benutzer) aufgenommen wurde.

- SECURITY
  Wieder wurde festgestellt, dass der Bytevergleich der Registry Dateien Unterschiede erkannte, diese nach der Konvertierungen aber nicht zu ermitteln waren.

- SOFTWARE
  Neben den im Abschnitt 4.2 beschriebenen Registry-Änderungen wurde das neue Benutzerkonto den Benutzer- und Administrator-Gruppen zugeordnet[36]. Des Weiteren wurden wie für alle anderen auch für die neue Kennung Einträge für den SyncManager[37] vorgenommen. Zudem wurde die NTLM-Authentifizierung (New Technology LAN Manager, vgl. [MS01-001]) des Telnet-Servers durch Veränderung des NTLM-Wertes[38] deaktiviert.

---

[32] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\000003EA

[33] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\Users\Names\hacker

[34] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Builtin\Aliases\00000220

[35] im Schlüssel HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Builtin\Aliases\00000221

[36] Anpassungen in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\GroupPolicy\S-1-5-21-1645522239-436374069-1343024091-1002\GroupMembership und HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ ProfileList\S-1-5-21-1645522239-436374069-1343024091-1002

[37] in den Schlüsseln HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Syncmgr\AutoSync\WWW3_hacker*

[38] im Schlüssel HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TelnetServer\1.0

- SYSTEM
  Außer den zu erwartenden und schon bekannten Änderungen der LogOn- und Shut-Down-Zeiten wurde während des Vorfalls die Konfigurationsdaten eines Telnet-Server in diesem Teil der Registry hinterlegt[39].

Die Registry-Analyse hat weitere Aufschlüsse ergeben. So kann festgehalten werden, dass die neue Benutzerkennung in die Gruppe der Administratoren aufgenommen und der Vorfall zumindest teilweise auch mit Telnet durchgeführt wurde, wobei keine NTML-Authentifizierung erfolgte.

Die Versuche haben gezeigt, dass Sicherheitsvorfälle, wie angenommen, Spuren auf den Festplatten der IT-Systeme hinterlassen und diese mit CompareSys festgestellt werden können. Die Spuren sind aber nur bedingt dafür geeignet, einen Sicherheitsvorfall vollständig zu erklären. Einschränkungen und deren Ursachen bei der Vorfallsanalyse mit CompareSys werden im folgenden Kapitel diskutiert.

[39] in HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00*\Enum\Root\ LEGACY_TLNTSVR*

5 Grenzen des Anwendungsbereichs von CompareSys

Wie erwartet ist die Software CompareSys als Nachweis eines Konzeptes (engl. *proof of concept*) kein Universalwerkzeug. In den folgenden Abschnitten werden aufgrund allgemeiner Überlegungen und der im vorangegangenen Kapitel beschriebenen Versuchsreihe die Grenzen des Anwendungsbereiches von CompareSys beschrieben.

Die Einordnung von CompareSys in die in Abschnitt 2.4 aufgestellte Klassifikation und unter weiterer Berücksichtigung der Lokalität (vgl. 2.3.2) verdeutlicht zunächst den Einsatzbereich: CompareSys kann Integritätsverletzungen auf der Treiberebene bzgl. spezieller Hardware (Festplatten) durch komparativ statische Analysen hostbasiert feststellen. Durch diese Einreihung und die genaue Betrachtung der Algorithmen lassen sich die im Folgenden beschriebenen Grenzen des Anwendungsbereiches erkennen.

5.1 Fokus auf Integritätsprüfung

CompareSys überprüft Festplattenzustände, indem die Verzeichnisse und Dateien direkt gemäß ihren Eigenschaften verglichen werden. Auch wenn alle Eigenschaften überprüft werden, die die FAT-Dateisysteme kennen, können dadurch nur Integritätsverletzungen dieser IT-System-Schicht entdeckt werden.

Manche Datei- und Betriebssysteme (z.B. NTFS 5 und Windows 2000) ermöglichen die Vergabe von Zugriffsrechten für einzelne Dateien. Die Zugriffsrechte können dann auch als Dateieigenschaft verstanden werden. Durch die Untersuchung dieser Eigenschaft könnten potentielle Vertraulichkeitsverletzungen erkannt werden. Sofern keine zusätzlichen Monitore zum Einsatz kommen, kann eine solche Analyse nur die Möglichkeit, nicht aber die konkrete Ausnutzung feststellen. Aus diesem Grund betrachtet CompareSys Zugriffsrechte und damit auch Vertraulichkeitsverletzungen nicht.

Auch Verletzungen der Verfügbarkeitsanforderungen an Dateien kann durch diesen komparativ statischen Festplattenvergleich nicht immer entdeckt werden. Die Verfügbarkeit wird üblicherweise in Prozent von Zeitspannen angegeben. Als Beispiel kann ein Anbieter von Web-Space, so genannte Web-Hoster, seinen Kunden eine Verfügbarkeit des Web-Servers von 95 % eines Jahres zusichern. Durch den verwendeten komparativ statischen Ansatz werden jedoch nur Zustände von zwei konkreten Zeitpunkten verglichen. Eine Analyse, ob eine Verfügbarkeitsverletzung vorliegt, kann dann nur für diese Zeitpunkte festgehalten werden, wenn eine Integritätsverletzung eine Unterschreitung der Verfügbarkeitsanforderungen verursacht. Wenn ein Web-Server-Prozess nicht gestartet werden kann, weil die notwendigen Dateien nicht mehr vorhanden oder integer sind, dann sind der Web-Server und auch die Web-Site nicht verfügbar. Dies führt je nach gestellten Verfügbarkeitsanforderungen zu einem Vorfall. Derartige Vorfälle, die durch Integritätsverletzungen auf der Festplatte verursacht werden, können auch durch CompareSys festgestellt werden.

5.2 Fehlende Berücksichtigung anderer Schichten

Integritätsvorfälle werden mit CompareSys durch Vergleich von Festplattenzuständen ermittelt. Sofern in dieser Diplomarbeit von Festplattenzuständen gesprochen wird, sind immer die Dateien gemeint, wie sie dem Betriebssystem zur Verfügung stehen. Die Betrachtung der Daten anderer Schichten, insbesondere der Low-Level-Informationen wurden außer Acht gelassen. Dies birgt aber das Risiko, dass nicht alle Daten analysiert werden, die sich auf einer Festplatte befinden und von einem Hacker genutzt werden könnten.

Beispielsweise gibt es eine Spezifikationsschwäche im Alternate Data Stream (ADS). Aufgrund der ADS Spezifikation für das New Technology File System (NTFS) können laut [Carvey 2001] Informationen in Dateien versteckt werden, ohne dass diese durch einen herkömmlichen Dateiaufruf geladen werden können. Selbst die Dateieigenschaften geben keinen Hinweis auf eine versteckte Datei. Diese Schwäche der Spezifikation wurde in [MS KB 101353] von Microsoft anerkannt und die Ausnutzung in [Carvey 2001] beschrieben. Eine derartige Möglichkeit kann von Hackern genutzt werden, um beispielsweise Spionage- oder „Fernwartungs"-Software wie NetBus (vgl. [Podrezov & Hypponen 2001]) zu verstecken. Das Ausführen der versteckten Dateien wird durch das Verbergen nicht verhindert. Um derart unsichtbare Dateien aufzufinden, sind besondere Tools wie Frank Heynes Freeware *LADS – List Alternate Data Streams* (vgl. [Heyne 2003]) nötig.

Aufgrund der in CompareSys verwendeten Zugriffsmethoden auf das Dateisystem können solche versteckten Informationen hingegen nicht erkannt werden. Um auch die Ausnutzung anderer vielleicht noch nicht bekannter Schwächen ausschließen zu können, müssten die Festplatten auf dem Low-Level untersucht werden. Eine solche Untersuchung wird wegen des hohen zeitlichen Aufwandes aber meistens nur nach ersten Hinweisen oder bei besonders hohen Anforderungen an die IT-Sicherheit durchgeführt.

Auch die höher liegenden Schichten der IT-Systeme werden von CompareSys nur indirekt berücksichtigt, sofern die Daten von den höheren Schichten auf die Festplatte geschrieben werden. Da Programme der höheren IT-System-Schichten auf die von CompareSys betrachtete Schicht direkt oder indirekt aufsetzen, können auch Integritätsverletzungen auf höheren Schichten erkannt werden, allerdings nur wenn diese auf der Festplatte abgebildet werden.

5.3 Grenzen des komparativ statischen Ansatzes

In Abschnitt 2.3.1 wurde das jeweilige Verständnis des Vorfallsprozesses von komparativ statischen und dynamischen Methoden erläutert. Bei komparativ statischen Ansätzen wird der gesamte Vorfallsvorgang als eine Transition verstanden. Dass die Vorfallstransition verfeinert werden kann und diverse Zwischenzustände existieren, bleibt bei dieser komparativ statischen Methode außer Acht. Zudem ergibt sich generell bei den komparativ statischen Ansätzen das Problem, wie der Zustand des IT-Systems definiert wird, beziehungsweise ob der Zustand des gesamten IT-Systems oder nur Zustände von Teilkomponenten betrachtet werden.

5.3.1 Beschränkung des Systemzustandes auf die Festplattenzustände

In Abschnitt 3.3.1 wurde bereits erwähnt, dass CompareSys lediglich die Repräsentanz der IT-System-Zustände auf den Festplatten untersucht. Der eigentliche Zustand eines IT-Systems wird aber durch die Zustände aller Systemschichten bestimmt und bezogen auf die Hardware-Komponenten auch nicht nur aus den Daten auf den Festplatten, sondern auch durch Betrachtung aller Speicher (inkl. kontextsensitiver Funktionseinheiten) festgestellt. Insbesondere der Arbeitsspeicher, der dem W32/Sapphire für seine Ausbreitungsfunktion ausreichte (vgl. [Hypponen et al. 2003]), aber auch die Speicher von Netzwerkkarten und anderen Hardware-Komponenten sogar bis hin zu den Speichern von angeschlossenen Druckern müssen für eine holistische Untersuchung von Sicherheitsvorfällen berücksichtigt werden, wie auf dem 19. Chaos Communication Congress gezeigt wurde (vgl. [CCC 2002]).

CompareSys liefert also nur einen Teilaspekt zu einer holistischen Untersuchung. Sofern durch gezielte Protokollierungseinstellungen alle Protokolldaten sich im betrachteten Zustandsausschnitt widerspiegeln, kann andererseits argumentiert werden, dass sehr viele der Sicherheitsverletzungen durch diesen Analyseansatz schnell und zielgerichtet identifiziert werden können. Bei der Bewertung der Analyseergebnisse und der darauf basierenden Schlussfolgerungen muss die beschriebene Einschränkung des Beobachtungsraumes aber berücksichtigt werden.

5.3.2 Ausnutzung von CompareSys zur Spurenverschleierung

Wie bei allen Analysen können auch die Ergebnisse von CompareSys genutzt werden, um die Spuren eines Sicherheitsvorfalls zu verwischen. Die eigentliche Intention hinter CompareSys besteht darin, dass die Analyseergebnisse genutzt werden, um Sicherheitsvorfälle zu verstehen und derart geschädigte Systeme bereinigen sowie bestenfalls wieder einsatzfähig machen zu können. Wenn das Ziel bei einem Sicherheitsvorfall nicht eine andauernde Schädigung durch Veränderung des Systems, sondern in temporären Veränderungen besteht, können die Ergebnisse auch genutzt werden, um die Spuren zu verwischen.

Der gewählte komparativ statische Ansatz hat in dieser Beziehung eine grundsätzliche Grenze. Mit diesem Ansatz können nur Änderungen erklärt werden, deren Auswirkungen an den zu vergleichenden Zuständen gemessen werden können. Heben Änderungen die Auswirkungen gegenseitig auf, dann kann dennoch eine Sicherheitsverletzung stattgefunden haben, obwohl beim Vergleich der Zustände dies nicht erkannt wird.

Gelingt es einem Angreifer, bei einem Web-Server seine eigenen Informationen auf der fremden Web-Site zu platzieren (wie in Abschnitt 4.3), so kann diese Information (z.B. ein Schlüssel zur Freischaltung von Software) daraufhin vielfach von „interessierten“ Besuchern der Web-Site herunter geladen werden. Nachdem die Information im Umlauf ist, kehrt der Angreifer zu dem Web-Server zurück und löscht die Information wieder von der Site. Die Analyse des Systems aufgrund der Festplattenzustände vor dem Angriff und nach Abschluss der Löschung wird nur wenige Hinweise darauf geben, dass hier eine Integritätsverletzung vorgelegen hat. Lediglich ein gut konfigurierter Web-Server wird die HTTP-Anfragen protokolliert haben, aufgrund derer erkannt werden müsste, dass Informationen, die eigentlich nicht zum Download bereit standen, herunter geladen wurden. Dieses Beispiel verdeutlicht, dass die Wahl der zu vergleichenden Zustände essentiell für eine erfolgreiche Vorfallsanalyse ist.

5.3.3 Nachvollziehbarkeit der wirklichen Abläufe

Wie in dem vorangegangenen Beispiel gibt es ein ähnliches Problem bei der Analyse der Dateiinhalte. Eugene Myers Lösung des LCS/SES-Problems gibt nur den kürzesten Pfad durch den Graphen aus (s. 3.7.2). Ob die Datei wirklich mit dem wenigsten Aufwand (also dem kürzesten Pfad) geändert wurde, kann wie bei allen komparativ statischen Analysen nicht festgestellt werden, da diese Informationen sich nicht aus den Zuständen ablesen lassen. Selbst unter Vernachlässigung des Aspektes, ob bei der Ersetzung einer Zeile zuerst die Ursprüngliche gelöscht und dann die Aktuelle geschrieben wurde oder andersherum, existieren in dem Graphen noch andere Pfade, die die Veränderungen erklären würden (exemplarisch in Abb. 52 dargestellt).

Zudem ist der Graph auch nicht vollständig. Zwischen den beiden betrachteten Zuständen können theoretisch noch viele Einträge in der Datei gemacht und wieder gelöscht worden sein. Sollen diese im Graphen berücksichtigt werden, dann muss der Graph unendlich viele Knoten enthalten, da unendlich häufig Zeichen geschrieben und wieder gelöscht worden sein könnten. Selbst wenn die Analyse wie bei CompareSys auf den Vergleich von Zeilen beschränkt wird, stellt dies keine Reduktion der Knoten des vollständigen Graphen dar, denn unendlich viele Variationen von Zeilen variabler Länge sind möglich und auch diese könnten beliebig oft geschrieben und wieder gelöscht worden sein.

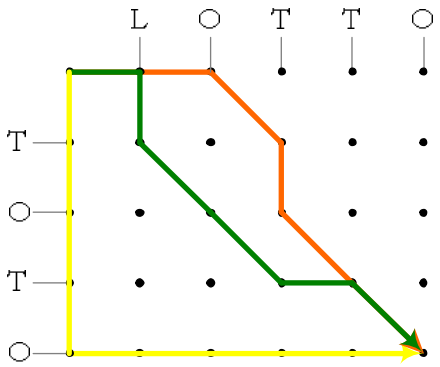


Abb. 52: Überführungsgraph mit alternativen Pfaden am Beispiel aus 3.7.2

Auch mit diesen Einschränkungen ist die Analyse mit dem Myers Algorithmus sinnvoll. Basierend auf den Ergebnissen können die weiteren Auswirkungen des Sicherheitsvorfalls auf das IT-System festgestellt werden. Wurden zum Beispiel Konfigurationsdateien verändert, um weitere Sicherheitsvorfälle zu ermöglichen oder den Angriff fortzusetzen, dann reicht die Anwendung des LCS/SES-Algorithmus aus, um dieses festzustellen. Mit den Analyseresultaten können auch so genannte Removal Tools entwickelt werden, die Veränderungen durch Sicherheitsvorfälle rückgängig machen.

5.4 Einschränkungen durch die Einsatzlokalität von CompareSys

CompareSys ist entwickelt worden, um lokal auf dedizierten IT-Systemen eingesetzt zu werden. Zwar ist der Zugriff auf die Festplattenzustände theoretisch auch über Netzwerke möglich, wodurch die Festplattenzustände physikalisch im zu untersuchenden IT-System verbleiben könnten, jedoch zeigt die Erfahrung, dass zumindest bei Microsoft Betriebssystemen eine Reihe

von Dateien vor dem Lesezugriff über das Netzwerk geschützt sind, insbesondere wenn es sich dabei um geöffnete Dateien wie denen der Registry handelt.

Aus dem dargelegten Grund sollten sich optimalerweise die Festplattenzustände physikalisch direkt im Analyse-System befinden (s. a. Beschreibung des Versuchsaufbaus in Abschnitt 4.1). Dafür ist es notwendig, das zu untersuchende IT-System herunterzufahren. In einer separaten Versuchsreihe wurde jedoch festgestellt, dass alleine durch das Starten und Herunterfahren eines IT-Systems Änderungen auf der Festplatte vorgenommen werden. Bei diesem Vorgang werden Datei-Eigenschaften verändert, da geöffnete Betriebssystem-Dateien geschlossen und mit neuem Änderungsdatum gespeichert werden. Aufgrund dieser Tatsache sind für eine derartige Analyse fundierte Kenntnisse über die Vorgänge eines Betriebssystems und der darauf installierten Software unerlässlich (vgl. Analyse der Softwareumgebung in 2.2).

5.5 Grenzen aufgrund der gewählten Algorithmen

Neben Grenzen, die sich schon aufgrund der Klassifikation dieser Methode zur Vorfallserkennung und -analyse herleiten lassen, schränken auch die in der Software angewandten Algorithmen den Anwendungsbereich weiter ein.

5.5.1 Einschränkungen bei der Ermittlung veränderter Dateien

Der im Abschnitt 3.7.1 beschriebene Algorithmus zur Ermittlung veränderter Dateien nutzt die Kombination aus Dateipfad und -name (z.B. WINNT\system32\shell32.dll) zur Identifikation von Dateien. Ein derart bezeichnetes Objekt, das in dem späteren Festplattenzustand gefunden wurde, wird nur anhand derselben Kombination im ursprünglichen Festplattenzustand gesucht. Wird im ursprünglichen Zustand kein entsprechendes Objekt gefunden, geht der Algorithmus davon aus, dass die Datei ergänzt bzw. im umgekehrten Falle gelöscht wurde.

Denkbar wäre, dass eine Datei zum Beispiel während eines Vorfalls umbenannt oder in ein anderes Verzeichnis verschoben worden ist. Der angewandte Algorithmus würde diese Art der Zustandsveränderung nicht direkt erkennen, sondern lediglich feststellen, dass es eine Datei im ursprünglichen Zustand gab, die im späteren Zustand nicht mehr existiert und eine andere Datei zuvor nicht vorkam, aber im späteren Zustand vorliegt. Die Aufgabe, zu erkennen, was an dieser Stelle wirklich geschah und in welcher Beziehung diese Dateien stehen, wird somit an den Anwender weitergereicht.

5.5.2 Schwäche des Myers Algorithmus in diesem Anwendungskontext

Der für die Feststellung der Dateiinhaltsänderungen verwendete Algorithmus von Eugene Myers wurde für das SES-Problem entwickelt. Ziel war es also das kürzeste Skript zu Überführung eines Dateiinhaltes in einen anderen Dateiinhalt zu finden. Dieses Ziel steht zwar nicht im Widerspruch zu dem Ziel des komparativ statischen Vergleichswerkzeugs, birgt aber eine Grenze, die im Folgenden aufgezeigt wird.

Wird CompareSys auf eine Datei angewendet, deren Inhalte lediglich umsortiert wurden, dann identifiziert der Algorithmus dies nicht. Zwar lässt sich dieser Problembereich vermutlich

auf Datenbank-Dateien, deren Inhalte im Zuge der Nutzung anders sortiert werden, und auf Quellcode-Dateien u.ä., deren Inhalte durch Anwendung von Optimierungswerkzeugen umsortiert werden, reduzieren, dennoch erkennt der Algorithmus nicht, dass die Zeilen nur getauscht wurden. Statt einer Ausgabe, die eine Umsortierung verdeutlicht, wird angezeigt, dass diverse Zeilen gelöscht (in Abb. 53 links rot dargestellt) und entsprechend viele geschrieben wurden (in Abb. 53 rechts blau dargestellt). Zwar ist diese Lösung für das vorliegende SES-Problem nicht falsch, aber für die Analyse von Sicherheitsvorfällen auch nicht optimiert.

Abb. 53: SES-Lösung bei Umsortierung von Dateiinhalten

Trotz der Einschränkungen bei Umsortierungen von Dateiinhalten kann der Algorithmus auf alle Dateien angewendet werden. Der Algorithmus stützt sich zudem darauf, dass die Dateiinhalte im ASCII-Format vorliegen, da zur zeilenbasierten Analyse nach dem Zeilenvorschub-Zeichen (ASCII-Zeichen 10) gesucht wird. Zwar können alle anderen Formate auch mit dem Algorithmus in CompareSys untersucht werden, doch wäre es möglich durch die Erstellung von dateiformatspezifischen Modulen für den Anwender besser lesbare Ergebnisse zu erreichen. Aus diesem Grund wird in CompareSys eine Schnittstelle zur Verfügung gestellt, über die diese Module eingebunden werden können.

5.6 Konsequenzen der Anwendungsgrenzen von CompareSys

In diesem Kapitel wurde gezeigt, dass CompareSys für einen eingegrenzten Anwendungsbereich gedacht und einsetzbar ist. Diese Grenzen zu kennen und ihren Einfluss auf die Analyseergebnisse zu verstehen, ist bei der Untersuchung von Computersicherheitsvorfällen sehr wichtig. Wie an einigen Beispielen gezeigt wurde, können sonst Sicherheitsverletzungen übersehen oder die Analyseresultate falsch interpretiert werden.

Holistische Analysen von Sicherheitsvorfällen können entsprechend der vorangegangenen Diskussion nur unter Betrachtung aller Aspekte der Klassifikation aus 2.4 erfolgen. CompareSys liefert dazu einen Beitrag. Des Weiteren kann, wie in Kapitel 4 dargestellt, die Software dazu genutzt werden, einige Sicherheitsvorfälle zu analysieren, wenn lediglich die Festplatten per Backup gesichert, andere Schutzmaßnahmen, wie die Installation von IDS oder anderen Monitoren, versäumt wurden.

6 Zusammenfassung & Ausblick

Im Kapitel 0 wurde aufgrund der Historie und der hohen wirtschaftlichen Schäden gezeigt, dass die Erkennung und Analyse von Sicherheitsvorfällen wichtig ist. Diese Gründe haben sich trotz der „Kampagnen [in den Unternehmen] zur Steigerung des Security-Bewusstseins“ (vgl. [Fox 2003]) in den letzten sechs Monaten, in denen diese Diplomarbeit entstand, nicht geändert. Obwohl die Anzahl der an das CERT/CC neu gemeldeten Schwachstellen im ersten Halbjahr 2003 in etwa der Hälfte der des letzten Jahres entspricht, sind in den ersten beiden Quartalen 2003 schon fast genauso viele Vorfälle an das CERT/CC berichtet worden wie in den vier Quartalen 2002 (vgl. [CERT/CC 2003]). Der Trend der Vorfallsverdoppelung scheint sich also fortzusetzen – unter anderem auch wegen der Würmer W32/MsBlaster (vgl. [Erdelyi 2003]) und W32/Sobig.F@MM (vgl. [Carrera 2003]).

In Kapitel 1 wurden die Grundlagen für diese Arbeit dargestellt. Dabei wurde festgestellt, dass Computersicherheitsvorfälle sich auf verschiedenen, interagierenden IT-System-Schichten ereignen und als Verletzungen der Anforderungen an Vertraulichkeit, Integrität und Verfügbarkeit definiert werden können. Die unterschiedlichen Vorfallsarten wurden entsprechend der Klassifikation von Moses (vgl. [Jackson et al. 1992: 23.3.2.2]) vorgestellt und Angrifftechniken schematisch skizziert. Ebenso schematisch war die Darstellung der Kausalkette für die Vorfallsbearbeitung durch CSIRTs.

Im Kapitel 2 wurden dann die Aufgaben Erkennung und Analyse hervorgehoben und unterstützende Methoden für diese CSIRT-Aufgaben eingeteilt. Neben der Differenzierung der Methoden (welche Sicherheitsaspekte auf welcher IT-System-Schicht können sie erkennen und analysieren) wurden Unterschiede bei der Auswertung des Vorfallsprozesses festgestellt: die dynamischen Methoden stützen sich auf die Transitionen und damit auf die Befehle, die ausgeführt werden; die komparativ statischen Methoden vergleichen Systemzustände.

Aus der letzteren Gruppe der Methoden wurde in Kapitel 3 eine ausgewählt und implementiert. Die so entstandene Software CompareSys vergleicht dabei nicht den gesamten Systemzustand, sondern ermittelt Veränderungen auf Festplatten und in den Dateien. Experimente mit der Software wurden in Kapitel 4 dokumentiert und ausgewertet. In Kapitel 5 konnten aufgrund der Ergebnisse und der Einordnung der Software in die Methodenklassifikation aus Kapitel 2 die Grenzen der Software ermittelt werden: CompareSys untersucht komparativ statisch Integritätsverletzung auf der Treiberebene. Aufgrund der Detailtiefe der gelieferten Ergebnisse konnten aber auch Rückschlüsse auf Sicherheitsverletzungen auf anderen IT-System-Schichten gezogen werden. Trotz dieser Möglichkeiten liefert CompareSys aus vier Gründen kein holistisches Vorfallsbild:

1) CompareSys verfügt nicht über die Möglichkeit verschlüsselte oder per ASCII semantisch nicht lesbare Dateiinhalte ausführlich zu analysieren und darzustellen.

2) CompareSys beschränkt sich auf den Vergleich von Festplattenzuständen. Diese repräsentieren aber nicht das gesamte IT-System, wie mit einem W32/Sapphire-Vorfall (vgl. [Hypponen et al. 2003]) leicht nachgewiesen werden kann.

3) CompareSys ermöglicht nur Rückschlüsse auf Vertraulichkeits- und Verfügbarkeitsverletzungen und kann nicht oder nur in beschränktem Maße diese selber feststellen.

4) Komparativ statische Systemanalysen hängen von der Wahl der Zustände ab und können vorfallsrelevante, rückgängig gemachte Zustandsänderungen nicht feststellen.

Neben dem Einsatz von weiteren Methoden zur Vorfallserkennung und -analyse können der erste und der vierte Grund durch Maßnahmen beseitigt werden.

Um mit CompareSys mehr als die ASCII-lesbaren Dateien sinnvoll analysieren zu können, muss die Software um entsprechende Module erweitert werden. Die Module bringen dadurch spezifisches Know-how in die Software ein. So wären Analysen der Konfigurationsdateien z.B. des IIS möglich, wenn ein Modul diese interpretieren kann. Weitere entsprechende Module etwa für Microsoft Word Dateien wären genauso sinnvoll, wie auch alle anderen in Kapitel 4 als derzeit mit CompareSys nicht analysierbar beschriebenen Dateien. Eine Schnittstelle ist für eine modulare Software-Erweiterung bereits vorgesehen. Der erste Grund für eine nicht-holistische Analyse mit CompareSys kann also durch Erweiterungen der Software behoben werden.

Um den vierten Grund für den Nicht-Holismus von CompareSys aufzuheben, muss das Verfahren quasi-dynamisch angewendet werden. Dies erfordert keine Anpassung der Software selber, sondern eine Anpassung des Vorfallsopfers bzw. des Versuchsablaufes. Die Zwischenzustände des vom Vorfall betroffenen Systems müssen für eine spätere Analyse gesichert werden. Da die Sicherung des Zustandes verhältnismäßig viel Zeit beansprucht, muss die Möglichkeit bestehen, den Vorfall zu unterbrechen und später fortzusetzen, ohne dass der Vorfall oder der Systemzustand dadurch beeinflusst wird. Dies ist in der Praxis üblicherweise nur in Testlabors wie dem IRT möglich.

Trotz der zwei genannten Maßnahmen kann mit CompareSys aber selbst kein holistisches Vorfallsbild und -verständnis erzeugt werden. Es liefert dennoch einen Beitrag zur ganzheitlichen Vorfallsuntersuchung. Im Vergleich zu anderen Werkzeugen, ermöglicht es nicht nur die Feststellung von Integritätsverletzungen, sondern auch eine genau Untersuchung der Veränderungen. Damit entspricht CompareSys den Anforderungen des IRT. Da keine Eingriffe in ein IT-System notwendig sind, um eine derartige Systemanalyse durchführen zu können, bietet sich dieser Ansatz vor allem auch dann an, wenn das betroffene IT-System nicht durch andere Monitore beobachtet wurde und nur eine Systemsicherung in Form eines Backups existiert.

"Good analysis is critical to the provision of a competent incident response service."

(aus [West-Brown et al. 1998: 3.4.2])

Quellenverzeichnis

Sofern es möglich war, wurden Bezüge zu den im Folgenden aufgeführten Quellen in Kombination mit einer Kapitelnummer hergestellt. Falls die Kapitel nicht nummeriert waren oder eine derartige Unterteilung nicht vorgesehen ist, greife ich stattdessen auf Seitenzahlen (S.), Foliennummern (F.) oder ähnliches zurück.

[ATI IB 4218]
ATI Technologies (Hrsg.): Info-Base# 4218: *Pollog.txt and PollSt.txt appear in the root directory after installing display drivers*. Internet http://mirror.ati.com/support/infobase/4218.html, Download 24.08.2003.

[aVTC 2003]
antiViren Test Center (Hrsg.): *Viren und Malware: Eine Einführung*. Universität Hamburg, 2003.

[BeyondCompare 2003]
Scooter Software Inc. (Hrsg.): *Beyond Compare Features*. Internet http://www.scootersoftware.com/features.html, Download: 29.06.2003.

[Back Orifice 2000]
SourceForge.net (Hrsg.): *BO2k – OpenSource Remote Administration Tool*. Internet http://www.backorifice2000.com, Download: 01.08.2003.

[BBAW 2003]
Berlin-Brandenburgische Akademie der Wissenschaften (Hrsg.): *Das Digitale Wörterbuch der deutschen Sprache des 20. Jahrhunderts: Erkennen*. Internet: http://www.dwds.de/cgi-bin/portalL.pl?search=Erkennen, Download: 14.09.2003.

[Bosau 2001: 1]
Bosau, Klaus: *Safety First!: Tripwire – eine Ortsbestimmung, Teil 1*. Internet http://www.linux-magazin.de/Artikel/ausgabe/ 2001/01/tripwire/tripwire.html, Download: 19.08.2003.

[Brömme 2001]
Brömme, Arslan: *Common Criteria 2.1*. Beitrag in [GBI 2001:5.5], Universität Hamburg, 2001.

[Brunnstein 1999]
Brunnstein, Klaus: *From AntiVirus to AntiMalware software and beyond: Another approach to the protection of customers from dysfunctional system behaviour*. Beitrag zur 22. National Information Systems Security Conference, 1999.

[Brunnstein 2003]
Brunnstein, Klaus: *Aktuelle Probleme der IT/Netzsicherheit*. Beitrag im Projektseminar, Universität Hamburg, 2003.

[BSI 2003]
Bundesamt für Sicherheit in der Informationstechnik (BSI): *Grundschutzhandbuch*. CD-ROM, Bonn: BSI, 2003.

[Carvey 2001]
Carvey, Harlan: NTFS *Alternate Data Streams*. Artikel in *Windows Security*. Ausgabe von 02.2001. Internet http://www.chi-publishing.com/portal/backissues/pdfs/ISB_2001/ISB0601/ISB0601HC.pdf, Download: 01.08.2003.

[CA-1999-04]
CERT/CC (Hrsg.): *CERT® Advisory CA-1999-04 Melissa Macro Virus*. Internet http://www.cert.org/advisories/CA-1999-04.html, Download: 30.08.2003

[Carrera 2003]
Carrera, Ero: *F-Secure Virus Descriptions: Sobig.F*. Internet http://www.europe.f-secure.com/v-descs/sobig_f.shtml, Download: 11.09.2003.

[CC 1999]
Common Criteria Implementation Board (CCIB) und International Organisation for Standardization (ISO): *Common Criteria 2.1: Common Criteria for Information Technology Security Evaluation*. ISO/IEC 15408, 1999.

[CCC 2002]
Chaos Computer Club e.V. (Veranstalter): *19. Chaos Communication Congress: Out Of Order*. Veranstaltet am 27., 28. und 29. Dezember 2002. Internet http://www.ccc.de/congress/2002/, Download: 01.08.2003.

[CCSS 2001]
Computer Security Institute; San Francisco's Federal Bureau of Investigation Computer Crime Squad: *2001 CSI/FBI Computer Crime and Security Survey*. Internet http://www.itsecure.com.au/resources/files/2001fbiccss.pdf, Download: 20.04.2003.

[CERT-Bund 2003]
Werner, Tillmann i.A. CERT-Bund, certbund@bsi.bund.de: *AW: CERT-Bund: Methoden der Vorfallserkennung und -analyse*. Persönliche Email: 28.04.2003.

[CERT/CC 2003]
CERT Coordination Center: *CERT/CC Statistics 1988-2002: Number of incidents reported*. 1997. Internet http://www.cert.org/ stats/cert_stats.html, Download: 10.04.2003.

[CNN 2000]
CNN.com (Hrsg.): *Cyber-attacks batter Web heavyweights: Strikes on eBay, Amazon, CNN.com follow Monday Yahoo! attack*. Internet http://www.cnn.com/2000/TECH/computing/02/09/cyber.attacks.01/, Download 30.08.2003.

[Cogswell & Russinovich 2003]
Cogswell, Bryce; Russinovich, Mark (Hrsg.): *Sysinternals Freeware*. Internet http://www.sysinternals.com/ntw2k/utilities.shtml, Download: 28.08.2003.

[ComTechDoc 2001]
The Computer Technology Documentation Project (Hrsg.): *Windows 2000 User Profiles*. Internet http://www.comptechdoc.org/os/windows/win2k/win2kusers.html, Download: 24.08.2003.

[CVS 2003]
CollabNet, Inc. (Hrsg.): *Version Management with CVS*. Internet http://www.cvshome.org/docs/manual/cvs-1.11.6/ cvs_1.html#SEC1, Download: 29.06.2003.

[DevPortal 2000]
Software Exploration, Ltd. (Hrsg.): *DevPortal™ White Paper*. Internet http://www.software-exploration.com/docs/DevPortalWhitePaper.doc, Download: 29.06.2003.

[Elder & Wolfe 2001]
Elder, D.; Wolfe, Robert: *Approaches to Policy Analysis*. Internet http://qsilver.queensu.ca/~wolfer/General/Analysis.html, Download: 10.04.2003.

[Elnitiarta & Han 1999]
Elnitiarta, Raul; Han, Wason; Symantec (Hrsg.): *BackOrifice2K.Trojan*. Internet http://www.symantec.com/avcenter/venc/data/back.orifice.2000.trojan.html, Download: 19.09.2003.

[Eppstein 1995]
Eppstein, David: *ICS 161: Design and Analysis of Algorithms*. Internet http://www.ics.uci.edu/~eppstein/161/960229.html, Download: 01.09.2003.

[Erdelyi 2003]
Erdelyi, Gergely: *F-Secure Virus Descriptions: Lovesan*. Internet http://www.europe.f-secure.com/v-descs/msblast.shtml, Download: 11.09.2003.

[Ethereal 2003]
Combs, Gerald et al. (Hrsg.): *Ethereal: Sniffing the glue that holds the Internet together*. Internet http://www.ethereal.com, Download: 29.08.2003.

[Feldmann & Senoucci 2002]
Feldmann, Lutz; Senoucci, Karim (Hrsg.): *Sicherer ins Internet: Einführung und Informationen zur Konfiguration gängiger Browser und Betriebssysteme*. Internet http://agn-www.informatik.uni-hamburg.de/papers/publications/hct/NTC.pdf, Download: 30.08.2003.

[Filo 2000]
Basta Computing (Hrsg.): *Filo: File and Folder Properties Editor*. Internet http://www.basta.com/ProdFilo.htm, Download: 23.09.2003.

[Fox 2003]
Fox, Dirk: *Kampagnen steigern Security-Bewusstsein*. Artikel in der Computer Zeitung 34. Jahrgang, Nummer 37, 08.09.2003.

[Freitag 2000]
Freitag, Sönke: *Webbasiertes Auffinden maliziöser Software mit fortschrittlichen heuristischen Verfahren („MWC - Malware Crawler")*. Diplomarbeit, Universität Hamburg, 2000.

[Fyodor 2003]
insecure.org (Hrsg.): *Top 75 Security Tools*. Internet http://www.insecure.org/tools.html, Download: 01.09.2003

[GBI 2001]
Brunnstein, Klaus: *Gestaltbarkeit und Beherrschbarkeit von Informatiksystemen (GBI)*. Vorlesungsskript, Universität Hamburg, 2001.

[Granger 2001]
Granger, Sarah: *Social Engineering Fundamentals, Part I: Hacker Tactics*. Internet http://www.securityfocus.com/infocus/1527, Download: 30.08.2003.

[Grigsby 1999]
Grigsby, Lee: *Automating Windows NT Setup Deployment Guide Supplement: Advanced Sysdiff. WhitePaper*. Redmond: Microsoft Corporation, 1999. Internet http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/winntas/deploy/depopt/advsysdf.asp, Download: 24.08.2003.

[Heyne 2003]
Heyne, Frank: *LADS – List Alternate Data Streams*. Internet http://www.heysoft.de/Frames/f_sw_la_de.htm, Download: 01.08.2003.

[Hoherz 2003]
Hoherz, Benjamin: *Einfluss von Incident Response auf die Erstellung von Notfallkonzepten*. Diplomarbeit, Universität Hamburg, 2003.

[Hoherz, Krüger, Menne & Michaelsen 2001]
Hoherz, Benjamin; Krüger, Silvio; Menne, Jan; Michaelsen, Nils: *Intrusion Detection Systeme in Firewalls*. Baccalaureatsarbeit, Universität Hamburg, 2003.

[Howard 1997]
Howard, John D.: *An Analysis Of Security Incidents On The Internet: 1989 - 1995*. Doktorarbeit, Carnegie Mellon University. 1997. Internet http://www.cert.org/research/JHThesis/Start.html, Download: 15.04.2003.

[Howard & Longstaff 1998]
Howard, John D.; Longstaff, Thomas A.: *A Common Language for Computer Security Incidents*. Sandia National Laboratories, 1998.

[Hypponen & Erdelyi 2003]
Hypponen, Mikko; Erdelyi, Gergely: *F-Secure Computer Virus Information Pages: Slammer*. Internet http://www.europe.f-secure.com/v-descs/mssqlm.shtml, Download: 30.08.2003

[Jackson & Hruska 1992]
Jackson, K. M.; Hruska, Jan (Hrsg.): *Computer Security Reference Book*. Butterworth Heinemann, 1992.

[Inf.-Duden 1993]
Lektorat des B.I.-Wissenschaftsverlags (Hrsg.): *Duden: Informatik*. 2. Aufl. Mannheim: Dudenverlag, 1993.

[IRT 2003a]
Incident Response Team (Hrsg.): *Incident Response: Labor zur Untersuchung von Computer-Vorfällen*. Beitrag zur Messe „Hamburger Computer Tage". Universität Hamburg, 2003. Internet http://commsy.informatik.uni-hamburg.de/commsy.php/IR@HCT2003.pps?cid=100 &mod=material&fct=getfile& iid=3283, Download: 06.08.2003

[IRT 2003b]
Buchwald, Christoph; Rößner, Christoph; Incident Response Team (Hrsg.): *Incident Response: Labor zur Untersuchung von Computer-Vorfällen*. Vortrag im Projektseminar „Aktuelle Probleme der IT/Netzsicherheit". Universität Hamburg, 2003.

[ISO 7498-2]
International Organisation for Standardization: *Information Processing Systems: Open Systems Interconnection Reference Model. Teil 2: Security Architecture.* ISO/IEC 7498-2, zitiert in [Shirey 2000].

[ISS 2000645]
Internet Security Systems (Hrsg.): *2000645:* HTTP *URL with double-encoded ../*. Internet: http://www.iss.net/security_center/advice/Intrusions/2000645/default.htm, Download: 24.08.2003.

[Janz 2000]
Janz, André: *Reverse Engineering: Rechtliche Rahmenbedingungen und praktische Möglichkeiten.* Studienarbeit, Universität Hamburg, 2000.

[JargonFile 2002]
Raymond, Erik S. (Hrsg.): *Jargon File: The New Hacker's Dictionary. Version 4.3.3*. Internet http://catb.org/~esr/jargon/jargon.html, Download: 17.04.2003.

[Kefk 2003]
Kefk (Hrsg.): *Verfügbarkeit*. Internet http://www.kefk.net/Hardware/Verfügbarkeit/ index.asp, Download: 30.08.2003.

[Kerner 1995]
Kerner, Helmut: *Rechnernetze nach* OSI. 3. Aufl. Bonn: Addson-Wesley, 1995.

[Knepper 2003]
Knepper, Thomas: *Analyseverfahren: Treiberentwicklung*. Projektdiskussion am 08.04.2003.

[Kofler 1998]
Kofler, Michael: Visual Basic 6: *Programmiertechniken, Datenbanken, Internet*. Bonn: Addison-Wesley-Longman, 1998.

[KonTraG 1998]
Deutscher Bundestag (Gesetzgeber): *Gesetz zur Kontrolle und Transparenz im Unternehmensbereich (KonTraG)*. Berlin: 1998. Internet http://www.sicherheitsforum-bw.de/downloads/KonTraG.pdf, Download 30.08.2003.

[Kossakowski 1995]
Kossakowski, Klaus-Peter: *Glossary of Computer Security Incident Handling Terms and Abbreviations*. DFN-CERT-Internet http://www.dfn-cert.de/eng/pre99papers/certterm.html, Download: 10.04.2003.

[Kossakowski 2000]
Kossakowski, Klaus-Peter: *Information Technology Incident Response Capabilities*. Hamburg: Libris Books On Demand, 2000.

[Kozierok 2001a]
Kozierok, Charles M.: *PC Operating System and File System Cross-Reference*. The PC Guide: Internet http://www.pcguide.com/ref/hdd/file/cross.htm, Download: 09.09.2003.

[Kozierok 2001b]
Kozierok, Charles M.: FAT *File System Disk Volume Structures*. The PC Guide: Internet http://www.pcguide.com/ref/hdd/file/fat.htm, Download: 30.06.2003.

[Kozierok 2001c]
Kozierok, Charles M.: *High Performance File System (HPFS)*. The PC Guide: Internet http://www.pcguide.com/ref/hdd/file/file_HPFS.htm, Download: 09.09.2003.

[Kozierok 2001d]
Kozierok, Charles M.: *New Technology File System (*NTFS*)*. The PC Guide: Internet http://www.pcguide.com/ref/hdd/file/ntfs/, Download: 09.09.2003.

[Krugler 2003]
Krugler, Martin: *Client Side Cache*. Persönliches Gespräch: 27.08.2003.

[Lancaster 2002]
Lancaster, Tom: *Windows Security Administration Tips: Help with debugging*. Internet http://searchwin2000.techtarget.com/tip/1,289483,sid1_gci832105,00.html, Download: 08.07.2003.

[Lauritsen 1999]
Lauritsen, Jesper: *Jesper NT Tools*. Internet http://www.ibt.ku.dk/jesper/NTtools/, Download: 24.08.2003.

[Mackie, Roculan, Russell & Van Velzen 2001]
Mackie, Andrew; Roculan, Jensenne; Russell, Ryan; Van Velzen, Mario: *Nimda Worm Analysis. Version 2*. SecurityFocus (Hrsg.): Internet http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf, Download: 23.06.2003.

[Mandia & Prosise 2001]
Mandia, Kevin; Prosis, Chris: *Incident Response: Investigating Computer Crime*. Berkeley: Osborne / McGraw-Hill, 2001.

[McAfee 1999]
McAfee Security (Hrsg.): *Virus Profile: Back Orifice 2000*. Internet http://us.mcafee.com/virusInfo/default.asp?id=description&virus_k=10229, Download: 19.09.2003.

[Metterhausen 2003]
Metterhausen, Werner: *Sicherheitslücken in Windows NT/2000/XP*. Vortrag, Hannover: CEFIS, 18.03.2003.

[Meyer 1992]
Meyers Lexikonredaktion (Hrsg.): *Meyers Grosses Taschenlexikon*. 4. Aufl. Mannheim: BI-Taschenbuchverlag, 1992.

[Michaelsen 2003]
Michaelsen, Nils: *Penetrationstest – Möglichkeiten und Grenzen*. Diplomarbeit, Universität Hamburg, 2003.

[MS01-001]
Microsoft Corporation (Hrsg.): *Microsoft Security Bulletin MS01-001: Web Client Will Perform* NTLM *Authentication Regardless of Security Settings* .Internet http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-001.asp, Download: 16.09.2003.

[MS03-007]
Microsoft Corporation (Hrsg.): *Microsoft Security Bulletin MS03-007: Unchecked Buffer In Windows Component Could Cause Server Compromise*. Internet http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-007.asp, Download: 14.08.2003.

[MS COM+ CRM 2003]
Microsoft Corporation (Hrsg.): *Developer Network (MSDN): COM+ Compensating Resource Manager*. Internet http://msdn.microsoft.com /library/default.asp?url=/library/en-us/cossdk/htm/pgservices_crm_5len.asp, Download: 16.08.2003.

[MS CSC 2002]
Microsoft Corporation (Hrsg.): *Developer Network (MSDN): Application Specification for Microsoft Windows 2000 for Desktop Applications: Chapter 6: OnNow/ACPI Support*. Internet http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnw2kcli/html/W2Kcli_chapter6.asp, Download 06.08.2003.

[MS FAT 2000]
Microsoft Corporation (Hrsg.): *Microsoft Extensible Firmware Initiative:* FAT*32 File System Specification. Hardware Whitepaper.* Version 1.03 vom 06.12.2000. Internet http://www.microsoft.com/hwdev/download/hardware/FATGEN103.doc. Download: 04.06.2003.

[MS IIS 2001]
Microsoft Corporation (Hrsg.): IIS *Insider – October 2001.* Internet http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/insider/iisi1001.asp, Download: 06.08.2003.

[MS KB 101353]
Microsoft Corporation (Hrsg.): *Knowledge Base 101353: Windows NT Supports Multiple Data Streams Inconsistently.* Internet http://support.microsoft.com/?kbid=101353, Download: 01.08.2003.

[MS KB 154174]
Microsoft Corporation (Hrsg.): *Knowledge Base 154174: Windows NT und Windows 95 hängen bei ungültigen* ICMP*-Datagramm-Fragmenten.* Internet http://support.microsoft.com/?kbid=154174, Download: 24.08.2003.

[MS KB 171002]
Microsoft Corporation (Hrsg.): *Knowledge Base 171002*: *Das Speichern von Ansichteinstellungen in Windows*. Internet http://support.microsoft.com/?kbid=171002, Download: 23.09.2003.

[MS KB 235994]
Microsoft Corporation (Hrsg.): *Microsoft Knowledge Base Article 235994: How Windows NT Saves Window Size and Location Parameters*. Internet http://support.microsoft.com/?kbid=235994, Download: 17.09.2003.

[MS KB 277675]
Microsoft Corporation (Hrsg.): *Knowledge Base 277675: How to Collect Local Security Authority Application Programming Interface Logging for Troubleshooting.* Internet http://support.microsoft.com/?kbid= 277675, Download: 17.08.2003.

[MS KB 295534]
Microsoft Corporation (Hrsg.): *Knowledge Base 295534: Überflüssiger Decodierungsvorgang kann Ausführung von Befehlen ermöglichen.* Internet http://support.microsoft.com/?kbid=295534, Download: 24.08.2003.

[MS KB 303204]
Microsoft Corporation (Hrsg.): *Knowledge Base 303204: Protokolldatei im Taskplaner kann falsch formatiert und schlecht lesbar sein.* Internet http://support.microsoft.com/?kbid=303204, Download: 24.08.2003.

[MS MMC 1998]
Microsoft Corporation (Hrsg.): *Microsoft Windows Management Instrumentation: Advantages to Developers*. Internet http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmi/html/msdn_wmiadvantages.asp, Download: 16.08.2003.

[MS MMC 2001]
Microsoft Corporation (Hrsg.): *Microsoft Management Console: Ereignisanzeige*. Online-Hilfe, 2001.

[MS OCA 2003]
Microsoft Corporation (Hrsg.): *Microsoft Online Crash Analysis: Microsoft-Absturzanalyse*. Internet http://oca.microsoft.com/de/, Download: 29.08.2003.

[MS RPC 2003]
Microsoft Corporation (Hrsg.): *Using* RPC *Protocols*. Internet http://www.microsoft.com/technet/treeview/default.asp?url=/technet/proddocs/en-us/isafp1/html/RPCAboutFilter.asp, Download 15.09.2003.

[MS RSM 2002]
Microsoft Corporation (Hrsg.): *Developer Network (MSDN): Windows Base Services: Removable Storage Manager*. Internet http://msdn.microsoft.com/library/default.asp?url=/library/en-us/devio/base/rsm_architecture.asp, Download: 06.08.2003.

[MS WMI 1999]
Microsoft Corporation (Hrsg.): *Windows Management Instrumentation: International Support Overview*. Whitepaper. 1999. Internet http://www.microsoft.com/windows2000/docs/WMIgloba.doc. Download: 16.08.2003.

[MSDN 2002]
Microsoft Corporation (Hrsg.): *Microsoft Developer Network (MSDN): Windows System Information: Registry Hives*. Internet http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sysinfo/base/registry_hives.asp, Download 06.08.2003.

[Mutlu, Schnell & Yüksel 2001]
Mutlu, Sibel; Schnell, Axel; Yüksel, Emine: *Intrusion Detection als ergänzender Sicherheitsmechanismus am Beispiel von UNIX*. Diplomarbeit, Universität Hamburg, 2001.

[Myers 1986]
Myers, Eugene W.: *An O(ND) Difference Algorithm and Its Variations*. Internet http://apinkin.net/space/DifferenceEngine, Download: 29.05.2003.

[MySQL 2003]
MySQL AB (Hrsg.): *MySQL-Referenzhandbuch*. Internet http://www.mysql.com/doc/de/index.html, Download 24.08.2003.

[P3 1999]
Neumann, Bernd: *Praktische Informatik 3: Algorithmen* & *Datenstrukturen (P3)*. Vorlesungsskript, Universität Hamburg, 1999.

[Pfleeger 2000]
Pfleeger, Charles P.: *Security in Computing*. 2. Aufl. New York: Prentice-Hall, 2000.

[Podrezov 2003]
Podrezov, Alexey: *F-Secure Computer Virus Information Pages: BO2k*. Internet http://www.europe.f-secure.com/v-descs/bo2k.shtml, Download: 01.08.2003.

[Podrezov & Hypponen 2001]
Podrezov, Alexey; Hypponen, Mikko: *F-Secure Computer Virus Information Pages: NetBus*. Internet http://www.europe.f-secure.com/v-descs/netbus.shtml, Download: 01.08.2003.

[Schiffmann & Schmitz 1996]
Schiffmann, Wolfram; Schmitz, Robert: *Technische Informatik 1: Grundlagen der digitalen Elektronik*. 3. Aufl. Karlsruhe: Springer-Verlag, 1996.

[Schlingloff 2001]
Schlingloff, Bernd-Holger: *Informationssicherheit I*. Vorlesungsskript, Universität Bremen, 2001. Internet http://www.informatik.uni-bremen.de/~hs/Lehre/scriptum.pdf, Download: 06.04.2003.

[Seedorf 2002]
Seedorf, Jan: *Verfahren zur Qualitätsbestimmung der Erkennung von bösartiger Software*. Diplomarbeit, Universität Hamburg, 2002.

[ShellIconCache 2003]
Radsoft.net (Hrsg.): *ShellIconCache*. Internet http://www.radsoft.net/workshop/assorted/mush.html, Download: 06.08.2003.

[Shirey 2000]
Shirey, Robert W.: RFC *2828: Internet Security Glossary*. Internet http://www.rfc-editor.org/rfc/rfc2828.txt, Download: 10.04.2003.

[Shoch & Hepps 1982]
Shoch, John F.; Hepps, Jon A.: *The 'Worm' Programs – Early Experience with a Distributed Computation*. Communications of the ACM, Volume 25, 1982, Internet: http://portal.acm.org/citation.cfm?id=358455&coll=portal&dl=ACM&CFID=11459021&CFTOKEN=69490695 Download: 28.07.2003.

[Smith 2001]
Smith, Robert: *Troubleshooting by Using the Setupapi.log File. Windows 2000 Server WhitePaper*. Redmond: Microsoft Corporation, 2001. Internet http://www.microsoft.com/windows2000/docs/Setupapilog.doc, Download: 06.08.2003.

[Soller 2002]
Soller, René: *Grafische Darstellung und Analyse der Gruppendependenzen von Websites mit maliziösen Inhalten*. Diplomarbeit, Universität Hamburg, 2002.

[Stallings 2000]
Stallings, William: *Computer Organization and Architecture: Designing for Performance*. 5. Aufl. Upper Saddle River: Prentice-Hall, 2000.

[Stephany 2001]
Stephany, Markus: *mirkes.de: dumphive*. Internet: http://www.mirkes.de/de/delphi/samples/dumphive.php, Download: 15.09.2003.

[STE 2000]
Floyd, Christiane; Oberquelle, Horst: *Software-Technik und -Ergonomie (STE)*. Vorlesungsskript, Universität Hamburg, 2000.

[Stoll 1999]
Stoll, Clifford: *Kuckucksei*. 3. Aufl. Frankfurt am Main: S. Fischer Verlag GmbH, 1999.

[Sun 1985]
Sun Mircosystems (Hrsg.): *Sun Cluster 3.0 Services: A Holistic Approach to Maximizing Uptime*. Internet http://www.sun.com/service/products/suncluster/index.html, Download 26.08.2003.

[T1 1998]
Heide, Klaus von der: *Technische Informatik 1: Digitale Systeme 1 (T1)*. Vorlesungsaufzeichnungen, Universität Hamburg, 1998.

[Tanenbaum 1998]
Tanenbaum, Andrew S.: *Computernetzwerke*. 3. Aufl. München: Prentice-Hall, 1998.

[Tasch 2003]
Tasch, Bertram: *Tasch*. Internet http://www.tasch.de/Cleaner/Tasch.cmd.txt, Download: 15.09.2003.

[TCSEC 1985]
Department of Defense (Hrsg.): *Trusted Computer System Evaluation Criteria*. Internet http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html, Download: 09.06.2003.

[TogetherSoft 2002]
Borland Software Corporation (Hrsg.): *Practical* UML*: A Hands-On Introduction for Developers*. Internet http://www.togethersoft.com/services/practical_guides/umlonlinecourse/, Download: 17.05.2003.

[Tripwire 1992]
Tripwire, Inc. (Hrsg.): *Tripwire.org: Questions and Answers*. Internet http://www.tripwire.org/qanda/index.php#1, Download: 29.06.2003.

[van Wyk & Forno 2001]
van Wyk, Kenneth R.; Forno, Richard: *Incident Response*. Sebastopol: O'Reilly, 2001.

[Wack 1991]
Wack, John P.: *Establishing a Computer Security Incident Response Capability (CSIRC)*. NIST Special Publication 800-3. Gaithersburg, Md.: US National Institute of Standards and Technology, 1991. Internet http://csrc.nist.gov/publications/nistpubs/800-3/800-3.pdf, Download: 06.08.2003.

[WatzNew 2003]
A.I.Studios (Hrsg.): *WatzNew: Features*. Internet http://www.watznew.com/features.html, Download: 29.08.2003.

[West-Brown, Stikvoort & Kossakowski 1998]
West-Brown, Moira; Stikvoort, Don; Kossakowski, Klaus-Peter: *Handbook for Computer Security Incident Response Teams (CSIRTs)*. Carnegie Mellon University, 1998.

[WinDiff 2003]
Microsoft Corporation (Hrsg.): *WinDiff*. Internet http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tools/tools/windiff.asp, Download: 17.05.2003

[Züllighoven & Soltos 1998]
Züllighoven, Heinz; Soltos, Thomas: *Das objektorientierte Konstruktionshandbuch: Nach dem Werkzeug- und Materialansatz*. Hamburg: Dpunkt Verlag, 1998.

Abbildungsverzeichnis

Abkürzungsverzeichnis

ADS	Alternate Data Stream
ASCII	American Standard Code for Information Interchange
aVTC	antiViren Test Center
BIOS	Basic Input Output System
BPB	BIOS Parameter Block
CERT	Computer Emergency Response Team
CERT/CC	Computer Emergency Response Team / Coordination Center
CIM	Common Information Model
COM+	Component Services
CRM	Compensation Resource Manager (vgl. [MS COM+ CRM 2003])
CSC	Client Side Cache (vgl. [MS CSC 2002])
CSI	Computer Security Institute
CSIRT	Computer Security Incident Response Team
CVS	Concurrent Versions System (vgl. [CVS 2003])
DMTF	Desktop Management Force
DNS	Domain Name System
DoS	Denial of Service
EAL	Evalutation Assurance Level (vgl. [CC 1999: Teil 3: 6])
FAT	File Allocation Table (vgl. [Kozierok 2001b])
HIDS	host-based IDS (vgl. [Mutlu: 3.2.2])
ICMP	Internet Control Message Protocol (s. RFC 792)
IDS	Intrusion Detection System (vgl. [Mutlu et al. 2001: 3])
IHT	Incident Handling Team
IIS	Internet Information Services
IP	Internet Protocol (s. RFC 791)
IRT	Incident Response Team der Universität Hamburg
ISO	International Organisation for Standardization
IT	Informationstechnologie *oder* Information Technology
HPFS	High Performance File System (vgl. [Kozierok 2001c])
HTTP	Hypertext Transport Protocol (s. RFC 2616)
LCS	Longest Common Sequence

LSASS	Local Security Authority Service Shell
MMC	Microsoft Management Console
NIDS	Network-Based IDS (vgl. [Mutlu: 3.2.1])
NTFS	New Technology File System (vgl. [Kozierok 2001d])
NTLM	New Technology LAN Manager (vgl. [MS01-001])
OSI	Open System Interconnection
RFC	Request for Comment (s. http://www.rfc-editor.org)
RNG	Random Number Generators
RSM	Removable Storage Managers (vgl. [MS RSM 2002])
RPC	Remote Procedure Call
SES	Shortest Edit Script
TCP	Transmission Control Protocol (s. RFC 793)
TCP/IP	Protokoll-Familie zur Kommunikation über Netzwerke (vgl. [Tanenbaum 1998: 1.4.2])
UML	Unified Modeling Language (vgl. [TogetherSoft 2002])
UUID	Universally Unique Identifier
VB	Visual Basic
WAM	Werkzeug, Automat, Material (meist als *WAM-Ansatz*, vgl. [Züllighoven])
WBEM	Web-Based Enterprise Management
WMI	Windows Management Instrumentation (vgl. [MS WMI 1999])
WSH	Windows Scripting Host

Anlage A Hinweise für die CompareSys-Installation

Die Software CompareSys liegt auf einer CD (s. Anlage G) bei. Vor der Installation sollte überprüft werden, ob die Mindestanforderungen an das System erfüllt sind:

minimal:

- Pentium II 233 MHz
- 64 MB Arbeitsspeicher
- 10 MB freier Festplattenspeicher
- Microsoft Windows 98, Microsoft Windows NT 4.0 (SP6) oder höher
- Microsoft Internet Explorer 5.5 oder höher
- kompatible Maus

empfohlen:

- Pentium IV 1,80 GHz
- 512 MB Arbeitsspeicher (mit großer Auslagerungsdatei)
- 10 MB freier Festplattenspeicher
- Microsoft Windows XP Professional
- Microsoft Internet Explorer 6
- kompatible Maus

Die Installation wird durch Ausführung des Programmes setup.exe im Verzeichnis CompareSys gestartet. Weitere Hinweise sowie eine Nutzungsanleitung befinden sich in der Online-Hilfe, die nach Installation und Programmstart im Menü „?“ durch Anklicken von „Hilfe“ aufgerufen werden kann, sowie in den Dateien index.html und readme.html auf der CD.

Anlage B Auszug aus dem Report des Dateisystemmonitors bei Versuchen mit Nimda

Zeit	Prozess	Zugriffsprozedur	Zieldatei
17:22:49	EXA_000_.EXE:892	IRP_MJ_READ*	C:\WINNT\system32\ntdll.dll
17:22:49	EXA_000_.EXE:892	IRP_MJ_READ*	C:\$Mft
17:22:49	EXA_000_.EXE:892	FASTIO_QUERY_OPEN	C:\WINNT\System32\MAPI32.DLL
17:22:49	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\WINNT\System32\MAPI32.DLL
17:22:49	EXA_000_.EXE:892	FASTIO_QUERY_STD_INFO	C:\WINNT\System32\MAPI32.DLL
17:22:49	EXA_000_.EXE:892	IRP_MJ_READ*	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_READ*	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_READ*	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_READ*	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	FASTIO_QUERY_OPEN	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\WINNT\System32\ws2_32.dll
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\WINNT\System32\ws2_32.dll
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLOSE	C:\WINNT\System32\ws2_32.dll
17:22:50	EXA_000_.EXE:892	FASTIO_QUERY_OPEN	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\WINNT\System32\WS2HELP.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\WINNT\System32\WS2HELP.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLOSE	C:\WINNT\System32\WS2HELP.DLL
17:22:50	EXA_000_.EXE:892	FASTIO_QUERY_OPEN	C:\WINNT\System32\MAPI32.DLL
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp
17:22:50	System:8	IRP_MJ_CLOSE	C:
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLOSE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_SET_INFORMATION	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_WRITE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_SET_INFORMATION	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_SET_INFORMATION	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLOSE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_READ	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	FASTIO_READ	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	FASTIO_WRITE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLOSE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CREATE	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_SET_INFORMATION	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe
17:22:50	EXA_000_.EXE:892	IRP_MJ_CLEANUP	C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\mep2.tmp.exe

Anlage C Versuchsergebnisse bei Änderungen durch ordentlichen Gebrauch

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße[1]	neue Dateigröße[1]	urspr. Attribute[2]	neue Attribute[2]	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
1	WINNT\CSC\00000002	false	64	64	SA	SA	24.7.03 14:00	5.8.03 15:33	23.7.03 18:30	23.7.03 18:30
2	WINNT\CSC\csc1.tmp	false	64	64	SA	SA	23.7.03 18:30	24.7.03 14:00	23.7.03 18:30	23.7.03 18:30
3	WINNT\Debug\ipsecpa.log	false	0	0	A	A	24.7.03 17:00	5.8.03 15:33	23.7.03 18:30	23.7.03 18:30
4	WINNT\Debug\ipsecpa.log.last	false	0	0	A	A	24.7.03 14:40	24.7.03 17:00	23.7.03 18:30	23.7.03 18:30
5	WINNT\Debug\oakley.log	false	0	0	A	A	24.7.03 17:00	5.8.03 15:33	23.7.03 18:30	23.7.03 18:30
6	WINNT\Debug\oakley.log.sav	false	0	0	A	A	24.7.03 14:40	24.7.03 17:00	23.7.03 18:30	23.7.03 18:30
7	WINNT\Debug\PASSWD.LOG	false	0	0	A	A	24.7.03 16:59	5.8.03 15:32	23.7.03 18:05	23.7.03 18:05
8	WINNT\system32\config\AppEvent.Evt	false	65536	65536	A	A	24.7.03 15:31	5.8.03 15:39	23.7.03 18:30	23.7.03 18:30
9	WINNT\system32\config\SysEvent.Evt	false	65536	65536	A	A	24.7.03 15:31	5.8.03 15:39	23.7.03 18:30	23.7.03 18:30
10	Dokumente und Einstellungen\Administrator\ntuser.ini	false	192	192	SH	SHA	24.7.03 15:31	5.8.03 15:39	23.7.03 18:44	23.7.03 18:44
11	Pollog.txt	true	72	72	A	A	24.7.03 16:59	5.8.03 15:32	24.7.03 14:40	24.7.03 14:40
12	PollSt.txt	true	750	1125	A	A	24.7.03 16:59	5.8.03 15:32	24.7.03 14:40	24.7.03 14:40
13	Dokumente und Einstellungen\Administrator\NTUSER.DAT	true	225280	225280	HA	HA	24.7.03 17:04	5.8.03 15:39	23.7.03 18:44	23.7.03 18:44
14	Dokumente und Einstellungen\Administrator\ntuser.dat.LOG	true	1024	1024	HA	HA	24.7.03 17:04	5.8.03 15:39	23.7.03 18:44	23.7.03 18:44
15	mysql\data\mysql.err	true	30	639	A	A	24.7.03 17:00	5.8.03 15:39	24.7.03 17:00	24.7.03 17:00
16	WINNT\CSC\00000001	true	64	64	SA	SA	24.7.03 17:00	5.8.03 15:39	23.7.03 18:30	23.7.03 18:30
17	WINNT\SchedLgU.Txt	true	778	994	A	A	24.7.03 15:31	5.8.03 15:39	23.7.03 18:30	23.7.03 18:30
18	WINNT\setupapi.log	true	153823	159131	A	A	24.7.03 14:49	5.8.03 15:33	23.7.03 18:06	23.7.03 18:06
19	WINNT\ShellIconCache	true	364780	365316	H	H	24.7.03 15:31	5.8.03 15:39	24.7.03 15:31	24.7.03 15:31
20	WINNT\system32\config\SAM	true	24576	24576	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 18:05	23.7.03 18:05
21	WINNT\system32\config\SAM.LOG	true	1024	1024	HA	HA	24.7.03 17:04	5.8.03 15:39	23.7.03 18:05	23.7.03 18:05
22	WINNT\system32\config\SECURITY	true	28672	28672	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 18:05	23.7.03 18:05
23	WINNT\system32\config\SECURITY.LOG	true	1024	1024	HA	HA	24.7.03 17:04	5.8.03 15:39	23.7.03 18:05	23.7.03 18:05
24	WINNT\system32\config\SOFTWARE	true	6533120	6533120	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 19:03	23.7.03 19:03
25	WINNT\system32\config\software.LOG	true	1024	1024	HA	HA	24.7.03 17:04	5.8.03 15:39	23.7.03 18:03	23.7.03 18:03
26	WINNT\system32\config\SYSTEM	true	2043904	2052096	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 19:03	23.7.03 19:03
27	WINNT\system32\config\SYSTEM.ALT	true	2043904	2052096	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 18:05	23.7.03 18:05
28	WINNT\system32\inetsrv\MetaBase.bin	true	206335	206324	A	A	24.7.03 17:04	5.8.03 15:39	23.7.03 18:47	23.7.03 18:47
29	WINNT\system32\NtmsData\NTMSDATA	true	110592	102400	A	A	24.7.03 17:00	5.8.03 15:39	23.7.03 18:31	23.7.03 18:31

[1] in Bytes

[2] A = zu archivieren, H = versteckt (engl. *hidden*), R = schreibgeschützt (engl. *read-only*), S = Systemdatei

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße[1]	neue Dateigröße[1]	urspr. Attribute[2]	neue Attribute[2]	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
30	WINNT\system32\NtmsData\ NTMSDATA.BAK	true	110592	102400	A	A	24.7.03 17:01	5.8.03 15:39	23.7.03 18:31	23.7.03 18:31
31	WINNT\system32\NtmsData\NTMSIDX	true	89288	89288	A	A	24.7.03 17:01	5.8.03 15:39	23.7.03 18:31	23.7.03 18:31
32	WINNT\Tasks\SA.DAT	true	6	6	HA	HA	24.7.03 17:01	5.8.03 15:39	23.7.03 18:19	23.7.03 18:19
33	WINNT\inf\7-zip.PNF			56472		A		5.8.03 15:33		5.8.03 15:33
34	WINNT\system32\LogFiles\W3SVC1\ ex030805.log			11806		A		5.8.03 15:39		5.8.03 15:34

Anlage D Versuchsergebnisse bei Defacement und Platzierung einer Angriffssoftware

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
1	Dokumente und Einstellungen\Administrator\Cookies\index.dat	false	16384	16384	A	A	24.7.03 17:02	29.7.03 11:01	23.7.03 18:44	23.7.03 18:44
2	Dokumente und Einstellungen\Administrator \Lokale Einstellungen\Temporary Internet Files \Content.IE5\index.dat	false	49152	49152	A	A	24.7.03 17:02	29.7.03 11:01	23.7.03 18:44	23.7.03 18:44
3	Dokumente und Einstellungen\Administrator \Lokale Einstellungen\Verlauf\History.IE5 \index.dat	false	49152	49152	A	A	24.7.03 17:02	29.7.03 11:01	23.7.03 18:44	23.7.03 18:44
4	WINNT\CSC\00000002	false	64	64	SA	SA	24.7.03 14:00	29.7.03 10:57	23.7.03 18:30	23.7.03 18:30
5	WINNT\CSC\csc1.tmp	false	64	64	SA	SA	23.7.03 18:30	24.7.03 14:00	23.7.03 18:30	23.7.03 18:30
6	WINNT\Debug\ipsecpa.log	false	0	0	A	A	24.7.03 17:00	29.7.03 11:00	23.7.03 18:30	23.7.03 18:30
7	WINNT\Debug\ipsecpa.log.last	false	0	0	A	A	24.7.03 14:40	29.7.03 10:57	23.7.03 18:30	23.7.03 18:30
8	WINNT\Debug\oakley.log	false	0	0	A	A	24.7.03 17:00	29.7.03 11:00	23.7.03 18:30	23.7.03 18:30
9	WINNT\Debug\oakley.log.sav	false	0	0	A	A	24.7.03 14:40	29.7.03 10:57	23.7.03 18:30	23.7.03 18:30
10	WINNT\Debug\PASSWD.LOG	false	0	0	A	A	24.7.03 16:59	29.7.03 10:59	23.7.03 18:05	23.7.03 18:05
11	WINNT\system32\config\AppEvent.Evt	false	65536	65536	A	A	24.7.03 15:31	29.7.03 11:08	23.7.03 18:30	23.7.03 18:30
12	WINNT\system32\config\SysEvent.Evt	false	65536	65536	A	A	24.7.03 15:31	29.7.03 11:08	23.7.03 18:30	23.7.03 18:30
13	Dokumente und Einstellungen\Administrator \ntuser.ini	false	192	192	SH	SHA	24.7.03 15:31	29.7.03 11:08	23.7.03 18:44	23.7.03 18:44
14	Pollog.txt	true	72	72	A	A	24.7.03 16:59	29.7.03 11:00	24.7.03 14:40	24.7.03 14:40
15	PollSt.txt	true	750	1500	A	A	24.7.03 16:59	29.7.03 11:00	24.7.03 14:40	24.7.03 14:40
16	Dokumente und Einstellungen\Administrator \NTUSER.DAT	true	225280	225280	HA	HA	24.7.03 17:04	29.7.03 11:08	23.7.03 18:44	23.7.03 18:44
17	Dokumente und Einstellungen\Administrator \ntuser.dat.LOG	true	1024	1024	HA	HA	24.7.03 17:04	29.7.03 11:08	23.7.03 18:44	23.7.03 18:44
18	Inetpub\wwwroot\index.html	true	738	114	A	RA	19.5.03 19:53	29.7.03 11:03	24.7.03 14:54	24.7.03 14:54
19	mysql\data\mysql.err	true	30	1248	A	A	24.7.03 17:00	29.7.03 11:08	24.7.03 17:00	24.7.03 17:00
20	WINNT\CSC\00000001	true	64	64	SA	SA	24.7.03 17:00	29.7.03 11:08	23.7.03 18:30	23.7.03 18:30
21	WINNT\SchedLgU.Txt	true	778	1210	A	A	24.7.03 15:31	29.7.03 11:08	23.7.03 18:30	23.7.03 18:30
22	WINNT\setupapi.log	true	153823	161801	A	A	24.7.03 14:49	29.7.03 10:57	23.7.03 18:06	23.7.03 18:06
23	WINNT\system32\config\SAM	true	24576	24576	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 18:05	23.7.03 18:05
24	WINNT\system32\config\SAM.LOG	true	1024	1024	HA	HA	24.7.03 17:04	29.7.03 11:08	23.7.03 18:05	23.7.03 18:05
25	WINNT\system32\config\SECURITY	true	28672	28672	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 18:05	23.7.03 18:05
26	WINNT\system32\config\SECURITY.LOG	true	1024	1024	HA	HA	24.7.03 17:04	29.7.03 11:08	23.7.03 18:05	23.7.03 18:05
27	WINNT\system32\config\SOFTWARE	true	6533120	6533120	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 19:03	23.7.03 19:03
28	WINNT\system32\config\software.LOG	true	1024	1024	HA	HA	24.7.03 17:04	29.7.03 11:08	23.7.03 18:03	23.7.03 18:03
29	WINNT\system32\config\SYSTEM	true	2043904	2068480	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 19:03	23.7.03 19:03
30	WINNT\system32\config\SYSTEM.ALT	true	2043904	2068480	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 18:05	23.7.03 18:05

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
31	WINNT\system32\inetsrv\MetaBase.bin	true	206335	206329	A	A	24.7.03 17:04	29.7.03 11:08	23.7.03 18:47	23.7.03 18:47
32	WINNT\system32\NtmsData\NTMSDATA	true	110592	102400	A	A	24.7.03 17:00	29.7.03 11:08	23.7.03 18:31	23.7.03 18:31
33	WINNT\system32\NtmsData \NTMSDATA.BAK	true	110592	102400	A	A	24.7.03 17:01	29.7.03 11:08	23.7.03 18:31	23.7.03 18:31
34	WINNT\system32\NtmsData\NTMSIDX	true	89288	89288	A	A	24.7.03 17:00	29.7.03 11:08	23.7.03 18:31	23.7.03 18:31
35	WINNT\Tasks\SA.DAT	true	6	6	HA	HA	24.7.03 17:00	29.7.03 11:08	23.7.03 18:19	23.7.03 18:19
36	bo2k.exe			167936		A		29.7.03 11:05		29.7.03 11:05
37	WINNT\inf\7-zip.PNF			56472		A		29.7.03 10:57		29.7.03 10:57
38	WINNT\system32\LogFiles\W3SVC1 \ex030729.log			65536		A		29.7.03 11:00		29.7.03 11:00
39	WINNT\system32\UMGR32.EXE			167936		A		29.7.03 11:05		29.7.03 11:06

Anlage E Versuchsergebnisse bei Schaffung einer Benutzerkennung und Defacement

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
1	Dokumente und Einstellungen\Administrator \Cookies\index.dat	false	16384	16384	A	A	8.8.03 12:51	8.8.03 15:59	23.7.03 18:44	23.7.03 18:44
2	Dokumente und Einstellungen\Administrator \Lokale Einstellungen\Temporary Internet iles\Content.IE5\index.dat	false	49152	49152	A	A	8.8.03 12:51	8.8.03 15:59	23.7.03 18:44	23.7.03 18:44
3	Dokumente und Einstellungen\Administrator \Lokale Einstellungen\Verlauf\History.IE5\ index.dat	false	49152	49153	A	A	8.8.03 1 2:51	8.8.03 15:59	23.7.03 18:44	23.7.03 18:44
4	Dokumente und Einstellungen\Administrator \ntuser.ini	false	192	192	SHA	SHA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:44	23.7.03 18:44
5	Dokumente und Einstellungen\Default User\ Cookies\index.dat	false	16384	16384	A	A	8.8.03 1 2:50	8.8.03 15:22	8.8.03 11:32	8.8.03 11:32
6	Dokumente und Einstellungen\Default User\ Lokale Einstellungen\Temporary Internet Files\Content.IE5\index.dat	false	32768	32768	A	A	8.8.03 1 2:50	8.8.03 15:22	8.8.03 11:32	8.8.03 11:32
7	Dokumente und Einstellungen\Default User\ Lokale Einstellungen\Verlauf\History.IE5\ index.dat	false	16384	16384	A	A	8.8.03 1 2:50	8.8.03 15:22	8.8.03 11:32	8.8.03 11:32
8	WINNT\CSC\00000001	false	64	64	SA	SA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:30	23.7.03 18:30
9	WINNT\Debug\ipsecpa.log	false	0	0	A	A	8.8.03 1 2:50	8.8.03 15:22	23.7.03 18:30	23.7.03 18:30
10	WINNT\Debug\ipsecpa.log.last	false	0	0	A	A	8.8.03 1 1:37	8.8.03 14:56	23.7.03 18:30	23.7.03 18:30
11	WINNT\Debug\oakley.log	false	0	0	A	A	8.8.03 1 2:50	8.8.03 15:22	23.7.03 18:30	23.7.03 18:30
12	WINNT\Debug\oakley.log.sav	false	0	0	A	A	8.8.03 1 1:37	8.8.03 14:56	23.7.03 18:30	23.7.03 18:30
13	WINNT\Debug\PASSWD.LOG	false	0	0	A	A	8.8.03 1 2:49	8.8.03 15:22	23.7.03 18:05	23.7.03 18:05
14	WINNT\Registration\{02D4B3F1-FD88-11D1-960D-00805FC79235}.crmlog	false	1048576	1048576	A	A	6.8.03 1 4:29	8.8.03 15:23	23.7.03 18:16	23.7.03 18:16
15	WINNT\system32\inetsrv\MetaBase.bin	false	206667	206667	A	A	8.8.03 1 2:53	8.8.03 16:09	23.7.03 18:47	23.7.03 18:47
16	WINNT\system32\NtmsData\NTMSIDX	false	91736	91736	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:31	23.7.03 18:31
17	WINNT\system32\wbem\Repository\ $WinMgmt.CFG	false	12	12	A	A	8.8.03 1 2:50	8.8.03 15:23	23.7.03 18:19	23.7.03 18:19
18	WINNT\system32\wbem\Repository\ CIM.REP	false	4587520	4587520	A	A	8.8.03 1 2:50	8.8.03 15:23	23.7.03 18:19	23.7.03 18:19
19	WINNT\Tasks\SA.DAT	false	6	6	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:19	23.7.03 18:19
20	Dokumente und Einstellungen\Administrator \NTUSER.DAT	true	249856	249856	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:44	23.7.03 18:44
21	Dokumente und Einstellungen\Administrator \ntuser.dat.LOG	true	1024	1024	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:44	23.7.03 18:44
22	Dokumente und Einstellungen\All Users\ Startmenü\Programme\Zubehör\ Unterhaltungsmedien\ Windows Media Player.lnk	true	665	683	A	A	23.7.03 18:44	8.8.03 15:19	23.7.03 18:19	23.7.03 18:19
23	Inetpub\mysql\mysql.err	true	1825	2061	A	A	8.8.03 1 2:53	8.8.03 16:15	24.7.03 14:59	24.7.03 14:59
24	Inetpub\wwwroot\index.html	true	738	84	A	A	19.5.03 19:53	8.8.03 15:52	24.7.03 14:54	8.8.03 15:52
25	Pollog.txt	true	72	72	A	A	8.8.03 1 2:49	8.8.03 15:22	24.7.03 14:54	24.7.03 14:54
26	PollSt.txt	true	4875	5625	A	A	8.8.03 1 2:49	8.8.03 15:22	24.7.03 14:54	24.7.03 14:54
27	WINNT\Debug\UserMode\userenv.log	true	1262	1748	A	A	24.7.03 14:37	8.8.03 15:19	24.7.03 14:36	24.7.03 14:36
28	WINNT\inf\mplayer2.inf	true	61616	35090	A	A	22.7.02 12:05	10.12.99 13:00	8.8.03 11:27	8.8.03 11:27

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
29	WINNT\inf\mplayer2.PNF	true	32548	53804	A	A	24.7.03 14:40	8.8.03 15:19	23.7.03 18:07	23.7.03 18:07
30	WINNT\OEWABLog.txt	true	1169	1514	A	A	8.8.03 1 1:34	8.8.03 15:19	23.7.03 18:19	23.7.03 18:19
31	WINNT\SchedLgU.Txt	true	2730	3162	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:30	23.7.03 18:30
32	WINNT\security\logs\scepol.log	true	2784	4640	A	A	8.8.03 1 1:31	8.8.03 11:31	8.8.03 11:31	8.8.03 11:31
33	WINNT\setupapi.log	true	162273	163293	A	A	8.8.03 1 1:26	8.8.03 15:19	23.7.03 18:06	23.7.03 18:06
34	WINNT\ShellIconCache	true	277068	375416	H	H	8.8.03 1 2:52	8.8.03 16:15	24.7.03 15:31	24.7.03 15:31
35	WINNT\system32\config\AppEvent.Evt	true	524288	65536	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:30	23.7.03 18:30
36	WINNT\system32\config\DEFAULT	true	12280	122880	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 19:03	23.7.03 19:03
37	WINNT\system32\config\default.LOG	true	1024	1024	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:03	23.7.03 18:03
38	WINNT\system32\config\SAM	true	24576	24576	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:05	23.7.03 18:05
39	WINNT\system32\config\SAM.LOG	true	1024	1024	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:05	23.7.03 18:05
40	WINNT\system32\config\SECURITY	true	28672	28672	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:05	23.7.03 18:05
41	WINNT\system32\config\SECURITY.LOG	true	1024	1024	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:05	23.7.03 18:05
42	WINNT\system32\config\SOFTWARE	true	6737920	6742016	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 19:03	23.7.03 19:03
43	WINNT\system32\config\software.LOG	true	1024	1024	HA	HA	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:03	23.7.03 18:03
44	WINNT\system32\config\SysEvent.Evt	true	65536	65536	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:30	23.7.03 18:30
45	WINNT\system32\config\SYSTEM	true	2347008	2351104	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 19:03	23.7.03 19:03
46	WINNT\system32\config\SYSTEM.ALT	true	2347008	2351104	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:05	23.7.03 18:05
47	WINNT\system32\LogFiles\W3SVC1\ ex030808.log	true	691	65536	A	A	8.8.03 1 1:12	8.8.03 16:05	8.8.03 11:07	8.8.03 11:07
48	WINNT\system32\NtmsData\NTMSDATA	true	110592	100592	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:31	23.7.03 18:31
49	WINNT\system32\NtmsData\ NTMSDATA.BAK	true	110592	110592	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:31	23.7.03 18:31
50	WINNT\system32\wbem\Logs\ wbemcore.log	true	44461	44685	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:18	23.7.03 18:18
51	WINNT\system32\wbem\Logs\ WinMgmt.log	true	580	748	A	A	8.8.03 1 2:53	8.8.03 16:15	23.7.03 18:19	23.7.03 18:19
52	WINNT\system32\wbem\Repository\ CIM.REC	true	1140850	1140848	A	A	8.8.03 1 2:52	8.8.03 15:23	6.8.03 14:37	6.8.03 14:37
53	WINNT\system32\LogFiles\W3SVC1\ ex030724.log		65536		A		24.7.03 15:31		24.7.03 14:08	
54	WINNT\system32\LogFiles\W3SVC1\ ex030730.log		2097152		A		30.7.03 16:42		30.7.03 15:13	
55	WINNT\system32\LogFiles\W3SVC1\ ex030731.log		137761		A		31.7.03 20:23		31.7.03 16:42	
56	WINNT\system32\LogFiles\W3SVC1\ ex030801.log		8414		A		1.8.03 8 :18		1.8.03 8:10	
57	WINNT\system32\LogFiles\W3SVC1\ ex030806.log		65536		A		6.8.03 1 4:45		6.8.03 14:29	
58	Dokumente und Einstellungen\hacker\ Anwendungsdaten\Microsoft\ Internet Explorer\brndlog.bak			141		A		23.7.03 18:19		8.8.03 15:19
59	Dokumente und Einstellungen\hacker\ Anwendungsdaten\Microsoft\ Internet Explorer\brndlog.txt			9422		A		8.8.03 15:19		8.8.03 15:19

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
60	Dokumente und Einstellungen\hacker\ Anwendungsdaten\Microsoft\ Internet Explorer\Quick Launch\Desktop anz eigen.scf			79		A		8.8.03 15:19		8.8.03 15:19
61	Dokumente und Einstellungen\hacker\ Anwendungsdaten\Microsoft\ Internet Explorer\Quick Launch\ Internet Explorer Browser starten.lnk			621		A		8.8.03 15:19		8.8.03 15:19
62	Dokumente und Einstellungen\hacker\ Anwendungsdaten\Microsoft\ Internet Explorer\Quick Launch\ Outlook Express starten.lnk			2093		R		8.8.03 15:19		8.8.03 15:19
63	Dokumente und Einstellungen\hacker\ Cookies\index.dat			16384		A		8.8.03 15:19		8.8.03 15:19
64	Dokumente und Einstellungen\hacker\ Desktop\ Verbindung mit dem Internet herstellen.LNK			766		A		8.8.03 15:19		8.8.03 15:19
65	Dokumente und Einstellungen\hacker\ Eigene Dateien\Eigene Bilder\Beispiel.jpg			9894		SH		23.7.03 18:18		8.8.03 15:19
66	Dokumente und Einstellungen\hacker\ Eigene Dateien\Eigene Bilder\Desktop.ini			434		SH		8.8.03 15:19		8.8.03 15:19
67	Dokumente und Einstellungen\hacker\ Favoriten\Desktop.ini			83		A		8.8.03 15:19		8.8.03 15:19
68	Dokumente und Einstellungen\hacker\ Favoriten\Links\Kostenlose Hotmail.url			113		A		8.8.03 15:19		8.8.03 15:19
69	Dokumente und Einstellungen\hacker\ Favoriten\Links\Links anpassen.url			121		A		8.8.03 15:19		8.8.03 15:19
70	Dokumente und Einstellungen\hacker\ Favoriten\Links\Windows.url			113		A		8.8.03 15:19		8.8.03 15:19
71	Dokumente und Einstellungen\hacker\ Favoriten\Media\Aktuelles.url			122		A		23.7.03 18:19		8.8.03 15:19
72	Dokumente und Einstellungen\hacker\ Favoriten\Media\Internet Radio Guide.url			127		A		23.7.03 18:19		8.8.03 15:19
73	Dokumente und Einstellungen\hacker\ Favoriten\Media\ Windows Medienübersicht.url			122		A		23.7.03 18:19		8.8.03 15:19
74	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Aktuelles.url			122		A		8.8.03 15:19		8.8.03 15:19
75	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Bloomberg.url					A		8.8.03 15:19		8.8.03 15:19
76	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Capitol Records.url			129		A		8.8.03 15:19		8.8.03 15:19
77	Dokumente und Einstellungen\hacker\ Favoriten\Medium\CBS.url			127		A		8.8.03 15:19		8.8.03 15:19
78	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ CNBC Dow Jones Business Video.url			123		A		8.8.03 15:19		8.8.03 15:19
79	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ CNET Today - Technology News.url			124		A		8.8.03 15:19		8.8.03 15:19
80	Dokumente und Einstellungen\hacker\ Favoriten\Medium\CNN Videoselect.url			123		A		8.8.03 15:19		8.8.03 15:19
81	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Disney.url			126		A		8.8.03 15:19		8.8.03 15:19
82	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ESPN Sports.url			124		A		8.8.03 15:19		8.8.03 15:19
83	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Fox News.url			127		A		8.8.03 15:19		8.8.03 15:19
84	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Fox Sports.url			129		A		8.8.03 15:19		8.8.03 15:19
85	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Hollywood Online.url			129		A		8.8.03 15:19		8.8.03 15:19
86	Dokumente und Einstellungen\hacker\ Favoriten\Medium\Internet Radio Guide.url			127		A		8.8.03 15:19		8.8.03 15:19
87	Dokumente und Einstellungen\hacker\ Favoriten\Medium\MSNBC.url			125		A		8.8.03 15:19		8.8.03 15:19

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
88	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ MUSICVIDEOS.COM.url			131		A		8.8.03 15:19		8.8.03 15:19
89	Dokumente und Einstellungen\hacker\ Favoriten\Medium\NBC VideoSeeker.url			123		A		8.8.03 15:19		8.8.03 15:19
90	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ TV Guide Entertainment Network.url			127		A		8.8.03 15:19		8.8.03 15:19
91	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ Universal Studios Online.url			136		A		8.8.03 15:19		8.8.03 15:19
92	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ Warner Bros. Hip Clips.url			130		A		8.8.03 15:19		8.8.03 15:19
93	Dokumente und Einstellungen\hacker\ Favoriten\Medium\ Windows Media Showcase.url			122		A		8.8.03 15:19		8.8.03 15:19
94	Dokumente und Einstellungen\hacker\ Favoriten\MSN.url			121		A		8.8.03 15:19		8.8.03 15:19
95	Dokumente und Einstellungen\hacker\ Favoriten\Radio Station Guide.url			197		A		8.8.03 15:19		8.8.03 15:19
96	Dokumente und Einstellungen\hacker\ Favoriten\Webereignisse.url			197		A		8.8.03 15:19		8.8.03 15:19
97	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\Anwendungsdaten\ Microsoft\Windows\UsrClass.dat			8192		HA		8.8.03 15:19		8.8.03 15:19
98	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\Anwendungsdaten\ Microsoft\Windows\UsrClass.dat.LOG			1024		HA		8.8.03 15:19		8.8.03 15:19
99	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ 8D4F69W7\desktop.ini			67		SHA		8.8.03 11:32		8.8.03 15:19
100	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ desktop.ini			67		SHA		8.8.03 11:32		8.8.03 15:19
101	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ GL23KVG5\desktop.ini			67		SHA		8.8.03 11:32		8.8.03 15:19
102	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ index.dat			32768		A		8.8.03 15:19		8.8.03 15:19
103	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ STIJ8XYB\desktop.ini			67		SHA		8.8.03 11:32		8.8.03 15:19
104	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\Content.IE5\ WXANODE3\desktop.ini			67		SHA		8.8.03 11:32		8.8.03 15:19
105	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\ Temporary Internet Files\desktop.ini			67		SH		8.8.03 15:19		8.8.03 15:19
106	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\Verlauf\desktop.ini			113		SH		8.8.03 15:19		8.8.03 15:19
107	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\Verlauf\History.IE5\ desktop.ini			113		SHA		8.8.03 11:32		8.8.03 15:19
108	Dokumente und Einstellungen\hacker\ Lokale Einstellungen\Verlauf\History.IE5\ index.dat			32768		A		8.8.03 15:19		8.8.03 15:19
109	Dokumente und Einstellungen\hacker\ NTUSER.DAT			172032		HA		8.8.03 15:19		8.8.03 15:19
110	Dokumente und Einstellungen\hacker\ ntuser.dat.LOG			1024		HA		8.8.03 15:19		8.8.03 15:19

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
111	Dokumente und Einstellungen\hacker \ntuser.ini			192		SHA		8.8.03 15:19		8.8.03 15:19
112	Dokumente und Einstellungen\hacker\Recent \Desktop.ini			124		SH		8.8.03 15:19		8.8.03 15:19
113	Dokumente und Einstellungen\hacker\ SendTo\3½-Diskette (A).lnk			129		A		23.7.03 18:06		8.8.03 15:19
114	Dokumente und Einstellungen\hacker\ SendTo \Desktop (Verknüpfung erstellen).DeskLink			0		A		23.7.03 18:18		8.8.03 15:19
115	Dokumente und Einstellungen\hacker\ SendTo\Eigene Dateien.mydocs			0		A		23.7.03 18:18		8.8.03 15:19
116	Dokumente und Einstellungen\hacker\ SendTo\E-Mail-Empfänger.MAPIMail			0		A		23.7.03 18:18		8.8.03 15:19
117	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Internet Explorer.lnk			609		A		8.8.03 15:19		8.8.03 15:19
118	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Outlook Express.lnk			579		A		8.8.03 15:19		8.8.03 15:19
119	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Adressbuch.lnk			635		A		8.8.03 15:19		8.8.03 15:19
120	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\Editor.lnk			1379		A		23.7.03 18:20		8.8.03 15:19
121	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Eingabeaufforderung.lnk			1357		A		23.7.03 18:20		8.8.03 15:19
122	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Eingabehilfen\Bildschirmlupe.lnk			1387		A		23.7.03 18:20		8.8.03 15:19
123	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Eingabehilfen\Bildschirmtastatur.lnk			1381		A		23.7.03 18:20		8.8.03 15:19
124	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Eingabehilfen\Hilfsprogramm-Manager.lnk			1361		A		23.7.03 18:20		8.8.03 15:19
125	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\Imaging.lnk			770		A		23.7.03 18:18		8.8.03 15:19
126	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Synchronisieren.lnk			1445		A		23.7.03 18:20		8.8.03 15:19
127	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Systemprogramme\Datenträgerbereinigung.lnk			1381		A		23.7.03 18:18		8.8.03 15:19
128	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Systemprogramme\Erste Schritte.lnk			1368		A		23.7.03 18:18		8.8.03 15:19
129	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Systemprogramme\Geplante Tasks.lnk			1640		A		23.7.03 18:18		8.8.03 15:19
130	Dokumente und Einstellungen\hacker\ Startmenü\Programme\Zubehör\ Windows-Explorer.lnk			1314		A		23.7.03 18:18		8.8.03 15:19
131	Dokumente und Einstellungen\hacker\ Vorlagen\amipro.sam			4570		A		10.12.99 13:00		8.8.03 15:19
132	Dokumente und Einstellungen\hacker\ Vorlagen\excel.xls			5632		A		10.12.99 13:00		8.8.03 15:19
133	Dokumente und Einstellungen\hacker\ Vorlagen\excel4.xls			1518		A		10.12.99 13:00		8.8.03 15:19
134	Dokumente und Einstellungen\hacker\ Vorlagen\lotus.wk4			2448		A		10.12.99 13:00		8.8.03 15:19
135	Dokumente und Einstellungen\hacker\ Vorlagen\powerpnt.ppt			12288		A		10.12.99 13:00		8.8.03 15:19
136	Dokumente und Einstellungen\hacker\ Vorlagen\presenta.shw			461		A		10.12.99 13:00		8.8.03 15:19
137	Dokumente und Einstellungen\hacker\ Vorlagen\quattro.wb2			4017		A		10.12.99 13:00		8.8.03 15:19
138	Dokumente und Einstellungen\hacker\ Vorlagen\sndrec.wav			58		A		10.12.99 13:00		8.8.03 15:19

Nr.	Dateiname	Dateiinhalt geändert	urspr. Dateigröße	neue Dateigröße	urspr. Attribute	neue Attribute	urspr. Änderungs-datum	neues Änderungs-datum	urspr. Erstellungs-datum	neues Erstellungs-datum
139	Dokumente und Einstellungen\hacker\ Vorlagen\winword.doc			4608		A		10.12.99 13:00		8.8.03 15:19
140	Dokumente und Einstellungen\hacker\ Vorlagen\winword2.doc			1769		A		10.12.99 13:00		8.8.03 15:19
141	Dokumente und Einstellungen\hacker \Vorlagen\wordpfct.wpd			30		RA		10.12.99 13:00		8.8.03 15:19
142	Dokumente und Einstellungen\hacker\ Vorlagen\wordpfct.wpg			57		RA		10.12.99 13:00		8.8.03 15:19
143	Inetpub\wwwroot\index.orig			738		A		19.5.03 19:53		24.7.03 14:54
144	WINNT\system32\Microsoft\Crypto\ntt\ ELDUMP.EXE			122880		A		14.12.98 15:29		8.8.03 15:56
145	WINNT\system32\Microsoft\Crypto\ntt\ ELSAVCLR.EXE			49152		A		16.12.98 9:17		8.8.03 15:56
146	WINNT\system32\Microsoft\Crypto\ntt\ ELSAVE.EXE			33792		A		7.9.98 12:03		8.8.03 15:56
147	WINNT\system32\Microsoft\Crypto\ntt\ NETVIEWX.EXE			40960		A		3.2.99 13:33		8.8.03 15:56
148	WINNT\system32\Microsoft\Crypto\ntt\ SRVBOOT.BAT			3158		A		3.2.99 16:35		8.8.03 15:56
149	WINNT\system32\Microsoft\Crypto\ntt\ USERSES.BAT			5947		A		5.2.99 13:47		8.8.03 15:56
150	WINNT\system32\Microsoft\Crypto\ntt\ WSSES.BAT			6147		A		5.2.99 13:47		8.8.03 15:56

Anlage F Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Jan Menne