



Nutzerservice des Universitätsrechenzentrums

Straße der Nationen 62, Raum 072 (Eingang am Hbf.), Tel. 0371/531-1656
Reichenhainer Straße 70, Raum B405 (Turmbau), Tel. 0371/531-3705
Öffnungszeiten: Mo-Fr 8:45 -- 11:30 Uhr, Mo, Die, Do, Fr 12:45 -- 18:00 Uhr
Helpdesk: hilfe@hrz.tu-chemnitz.de



Foto: "Sardinien", C. Ziegler

Wir wünschen unseren Nutzern erlebnisreiche und erholsame Urlaubstage

Impressum

Herausgeber:
TU Chemnitz
Universitätsrechenzentrum
Str. der Nationen 62
09111 Chemnitz
Leiter: Prof. Dr. U. Hübner
E-mail: huebner@hrz.tu-chemnitz.de

Redaktion:
Dipl.-Math. Ursula Riedel

Redaktionsbeirat:
Dipl.-Math. Matthias Clauß
Dipl.-Inform. Frank Richter
Dr. Wolfgang Riedel

Redaktionsschluss: 06.04.2004

Anmerkungen: Bezeichnungen hier genannter Erzeugnisse, die auch eingetragene Warenzeichen sind, wurden nicht besonders gekennzeichnet. Eine fehlende Kennzeichnung heißt nicht, dass die Bezeichnung ein freies Warenzeichen ist. Die Beiträge enthalten Links zu anderen Seiten im Internet. Gemäß einem Urteil des Landgerichts Hamburg vom 12. Mai 1998 wird hiermit erklärt, dass wir keinen Einfluss auf die Gestaltung und auf die Inhalte der referenzierten Seiten haben. Wir distanzieren uns hiermit ausdrücklich von allen Inhalten aller referenzierten Seiten.

Mitteilungen des URZ

2/2004

In dieser Ausgabe

- Nutzung der Computerpools
- Unicode - eine neue Art der Zeichenkodierung
- Sicheres Programmieren mit PHP (Teil 2)
- NIDS im Campusnetz
- MONARCH
- Achtung, Mail-Würmer!
- Kurzinformationen



Nutzung der Computerpools

Durch die Plattformupgrades in den Ausbildungspools stehen eine Reihe von neuen Anwendungen zur Verfügung. Auch die Hinweise für unsere Nutzer, welche Applikationen das URZ empfiehlt, sind dabei überarbeitet worden. Dieser Artikel beschreibt den aktuellen Stand der verfügbaren und empfohlenen Softwareprodukte und zeigt, welche Unterstützung das URZ bei Problemen bei der Nutzung von Software leistet.

Stand der Nutzungsmöglichkeiten

In den letzten Wochen hat das URZ in den Ausbildungs-Pools Plattformupgrades durchgeführt, um den Nutzern für die nächsten Monate und Jahre wieder aktuelle und attraktive Nutzungsbedingungen bieten zu können. Nach dazu erforderlichen Hardware-Aufrüstungen wurden auf den Poolrechnern **Linux Fedora Core 1** und **Windows XP** installiert. Unter <http://www.tu-chemnitz.de/urz/pools/> ist eine aktuelle Übersicht der verfügbaren Poolräume mit der jeweiligen Ausstattung zu finden. Die konkreten Informationen zu den installierten Plattformen sind auf den Webseiten <http://www.tu-chemnitz.de/urz/linux/> (zu Linux) und <http://www.tu-chemnitz.de/urz/xp/> (zu Windows XP) zusammengestellt. Dabei möchten wir insbesondere auf die Informationssammlungen "FAQ" (Frequently Asked Questions) verweisen, in denen (für beide Plattformen) die wichtigsten Anwenderfragen benannt und Lösungen beschrieben sind.

Anwendungen unter Linux

Auf den Poolrechnern sind unter Linux Fedora Core 1 derzeit knapp 1600 Softwareprodukte installiert (<http://www.tu-chemnitz.de/urz/anwendungen/linux/liste.html>), der überwiegende Teil davon als rpm-Paket. Allerdings ist unter Linux die Grenze zwischen "Anwendung" und "systemnahen Werkzeug" ziemlich verschwommen, aber trotzdem zeigt die genannte Zahl, dass der Nutzer eine gewaltige Menge von Software zur Verfügung hat.

Das URZ bemüht sich deshalb seit längerem, den Nutzern (insbesondere den Linux-Einsteigern und den unerfahrenen Anwendern) einen Weg durch diese verwirrende Vielfalt zu zeigen. Auf der Webseite http://www.tu-chemnitz.de/urz/anwendungen/linux/index_fc1.html findet sich eine Einteilung der wichtigsten Programme u.a. in solche für "Einsteiger" und solche für "Fortgeschrittene Nutzer". Auf den verlinkten Seiten sind die Produkte genannt, die das URZ als wichtig und geeignet für den jeweils beschriebenen Einsatzzweck betrachtet. Für jede Anwendung ist dabei eine Webseite verlinkt, auf der genauere Informationen über die betreffende Software, ihre Nutzbarkeit und weiterführende Dokumentation beschrieben sind (einige noch fehlende Seiten stehen in Kürze zur Verfügung).

Auf der "Einsteiger-Seite" wird deutlich, dass das URZ zwei Hauptanwendungen favorisiert: *Mozilla* für alle Netzdienste (hauptsächlich Webbrowser und E-Mail-Client) und *OpenOffice* als Standard-Büroapplikation. Das sind aus unserer Sicht die "Killerapplikationen" in den Pools - nicht weil diese Software besonders ressourcenhungrig

Wir verabschieden Erich Meißner

Vielleicht hat ihn der Eine oder Andere an unserer Uni schon vermisst, unseren Kollegen Erich Meißner. Als Techniker des URZ war er im Rahmen des "Hardware-Reparaturservice" in den letzten Jahren fast überall in unserer Uni, wo Computer im Einsatz sind.

Seit fast 40 Jahren hat sich Herr Meißner - gemeinsam mit seinen jeweiligen Kollegen - um "die Rechentechnik" in unserer Einrichtung gekümmert. Wenn man ihm beim "Kramen in Erinnerungen" zugehört hat, konnte man viel Interessantes erfahren, schließlich hat er fast das gesamte Spektrum an Computer- bzw. Rechentechnik (wie es z.B. in den 70-er Jahre noch hieß) betreut.

Angefangen hat es mit dem ZRA1, dem ersten Computer an unserer Uni, dann kamen die ODRA, der R 300 und die Rechner der ESER-Reihe - um nur die Wichtigsten zu nennen. Für die technische Verfügbarkeit der gesamten Palette der bis 1990 an der damaligen Technischen Hochschule vorhandenen Computertechnik war Herr Meißner mit verantwortlich. Mit der Etablierung des Universitätsrechenzentrums als zentrale Einrichtung der TU 1991 wurden die Aufgaben in der Gruppe Technik spezialisiert. Herr Meißner war seitdem speziell für die Klima- und Drucktechnik verantwortlich und war im Hardwareservice des URZ tätig. Wir danken Herrn Meißner für seinen unermüdlichen Einsatz.



Seit dem 1. März genießt Herr Meißner nun zunächst die Ruhephase der Altersteilzeit und hat sich vorgenommen, mit dem Wohnmobil noch viel von Europa kennenzulernen. Wir wünschen ihm dazu eine stabile Gesundheit und viele schöne Erlebnisse und Eindrücke.

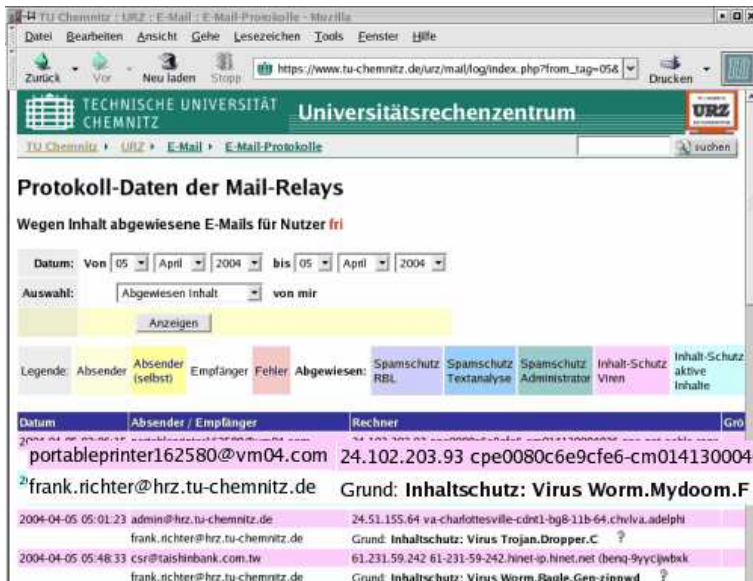
Erläuterung der Abkürzungen:

ZRA1 - Zeiss-Rechenautomat, erster industriell hergestellter Rechner der DDR

R 300 - Robotron 300 aus dem VEB Elektronische Rechenmaschinen Karl-Marx-Stadt

ESER - "Einheitliches System Elektronischer Rechenmaschinen" im RGW (IBM-kompatibel)

Ursula Riedel, April 2004



Die OpenSource-Virensoftware ClamAntiVirus ist kürzlich durch eine Studie als "wenig wirksam" eingestuft worden. Diese Einschätzung können wir nicht teilen. Freilich fehlen viele ältere, kaum noch relevante Virenkennungen. Aber bei akutem Auftreten von neuen E-Mail-Viren stehen sehr schnell aktuelle Viren-Signaturen zum Erkennen bereit - meist schneller als bei kommerziellen Anbietern. So werden an den Mail-Relays der TU täglich zwischen 1000 und 5000 E-Mails mit Viren unschädlich gemacht.

Für Sie als E-Mail-Benutzer bleibt trotz des zentralen Virenfilters die Sorgfaltspflicht erhalten:

- Im Mail-Programm: **Vorm Klicken überlegen!** Öffnen Sie Mail-Anhänge nur dann, wenn Sie vom Kommunikationspartner auch einen Anhang erwarten.
- Setzen Sie Antivirensoftware mit aktuellen Virenkennungen ein.

Siehe auch: <http://www.tu-chemnitz.de/urz/viren/>

Frank Richter, April 2004

wäre, sondern weil der überwiegende Teil aller Nutzer lediglich genau diese beiden Anwendungen benötigt bzw. benutzt. Das Interessante dabei ist noch, dass diese beiden Anwendungen plattformunabhängig existieren, sie können in identischer Art und Weise sowohl unter Windows als auch unter Linux betrieben werden.

Die Übersichtsseite für (schon etwas) fortgeschrittene Nutzer benennt u.a. Software für Multimedia-Anwendungen (Player, Bildbetrachtung und -bearbeitung), Viewer für verschiedene Datenformate (z.B. PDF, Postscript, RTF, DOC) aber auch fachgebietspezifische Applikationen für Mathematik und Konstruktion (CAD).

Anwendungen unter Windows

Unter Windows ist traditionell der Anteil kommerzieller Software am Gesamtangebot wesentlich größer als die verfügbare freie Software. Hieraus entsteht aber bei dem stattgefundenen Plattformupgrade ein ernstes Problem: die Übernahme aller bisher unter Windows NT verfügbaren Produkte nach Windows XP würde (im Zusammenhang mit den dabei notwendigen Softwareupdates) enorme Kosten verursachen. Außerdem wird die Situation dadurch verschärft, dass die großen Softwarehersteller fast ausnahmslos keine geteilte Lizenznutzung (*shared license* oder *floating license*) unter Windows erlauben. Alternativ müsste man für alle Rechner, von denen aus potentiell der Zugriff auf die betreffende Software möglich wäre, Lizenzen erwerben. In der Regel ist das aus finanzieller Sicht ausgeschlossen - das URZ verfügt nur über sehr eingeschränkte Gelder für Software.

Wir orientieren deshalb darauf, auch unter Windows (kosten)freie Software einzusetzen. Wichtige Programme sind bereits verfügbar, die Bereitstellung weiterer wird geprüft. Als Hauptapplikationen stehen somit unter Windows XP Mozilla und OpenOffice zur Verfügung. Damit wird nochmal offensichtlich: wer nur diese beiden Anwendungen benutzt, kann das gut auch unter Linux tun ...

Die beiden wichtigsten verfügbaren kommerziellen Softwareprodukte sind MS Office und CorelDraw, da OpenOffice nach wie vor keine 100%ig äquivalente Alternative darstellt. Leider werden von vielen Lehrbeauftragten die Unterlagen für die Studenten im Intranet in proprietären Datenformaten (wie Powerpoint) abgelegt, so dass die Benutzung entsprechender kommerzieller Software notwendig ist, um mit Sicherheit das Dokument inhaltlich und optisch richtig zu erhalten.

Die Liste der unter Windows XP in den Pools verfügbaren Anwendungen ist im Intranet unter <http://www.tu-chemnitz.de/urz/anwendungen/sw-xppools.html> veröffentlicht. Dabei soll nochmal betont werden, dass diese Liste den aktuellen Stand widerspiegelt. Wir arbeiten ständig daran, weitere Anwendungen verfügbar zu machen.

Anwenderunterstützung - FAQ

Das URZ betrachtet es als eine seiner wichtigsten Aufgaben, die Nutzer der Computerpools bei der Nutzung der verfügbaren Software zu beraten und Hilfestellung bei Problemen zu geben. Aus diesem Grund bieten wir die o.g. Software-Übersichtsseiten und die applikationsspezifischen Webseiten an. Weiterhin haben wir damit begonnen, Sammlungen von oft gestellten Fragen (und deren Antworten) zu veröf-

fentlichen. Diese FAQ zu Anwendungssoftware sind im Intranet unter <http://www.tu-chemnitz.de/urz/anwendungen/faq.html> erreichbar. Im Moment befindet sich diese Seite noch im Aufbau bzw. der Umorganisation. Wir arbeiten daran, dass unsere Nutzer in Kürze eine detaillierte und hilfreiche Übersicht zu den wichtigsten Anwendungsproblemen vorfinden.

Besonders hingewiesen sei in diesem Zusammenhang noch auf die FAQ zu Unicode (<http://www.tu-chemnitz.de/urz/linux/faq/unicode.html>), in der wichtige Probleme, die im Zusammenhang mit der neuen Codebasis unter Linux auftreten können, zusammengetragen sind (siehe auch Artikel zu Unicode in dieser Ausgabe).

Dr. Wolfgang Riedel, April 2004

Achtung, Mail-Würmer!

Neben Spam als unerwünschter, kommerzieller E-Mail macht sich zunehmend eine andere Plage breit: Die rasante Verbreitung von Schadprogrammen per E-Mail. "Klickfreudige" Benutzer öffnen im Mailprogramm ohne Überlegung E-Mail-Anhänge und infizieren somit Ihre PC-Systeme (fast ausschließlich Windows-PCs). Diese versenden nun selbst massenhaft E-Mails mit gefährlichem Anhang.

Dabei werden Empfängeradressen zufällig aus Dateien des Windows-Systems (Adressbuch, "Temporäre Internet-Dateien") verwendet. Aber auch die Absenderadresse wird willkürlich gewählt, wogegen man sich als Inhaber einer solchen Adresse leider nicht wehren kann.

Bei vielen Internet-Providern, in Firmen- und Hochschulnetzen wird der E-Mail-Verkehr durch Virenchecker geprüft. Leider erzeugen diese Prüfer häufig Hinweis-Mails ("Ihre E-Mail enthielt einen Virus") an den vermeintlichen Absender, der diese Mail in den meisten Fällen jedoch gar nicht selbst abgeschickt hat, sondern überhaupt nichts damit zu tun hat. Solche Warnungs-Mails sind für den Empfänger oft verwirrend und stiften Verunsicherung, von der Belastung der E-Mail-Infrastruktur und der Administratoren ganz zu schweigen.

In den Mail-Relays der TU Chemnitz setzen wir den Virenchecker **ClamAntiVirus** ein. Wenn eine E-Mail mit enthaltenem Virus identifiziert wird, wird diese E-Mail gelöscht - der vermeintliche Absender wird nicht informiert. Der Empfänger kann diese Aktion an Hand einer Eintragung in seinen Protokoll-Daten via WWW sehen:

<https://www.tu-chemnitz.de/urz/mail/log/>

Verweise:

- Deutsche Initiative für Netzwerkinformation
- DINI-Zertifikat Dokumenten- und Publikationsserver (pdf)
- MONARCH
- Recherche-Interface
- Swish-E

Sebastian Kratzert, Christoph Ziegler, April 2004

Unicode - eine neue Art der Zeichenkodierung

Die Darstellung von Textzeichen im Zusammenhang mit der Computertechnik war bisher in allen Betriebssystemen recht inhomogen. Um die Kodierung der Textzeichen zu vereinheitlichen, wurde der Unicode-Standard ISO-10646 entwickelt. Mit der Umstellung der vom URZ administrierten Linux-Rechner auf Fedora Core 1.0 erfolgt erstmals ein Umstieg auf Unicode im Format UTF-8. Der Artikel beschreibt Vorteile von Unicode, eventuelle Probleme beim Umstieg und Hintergrundinformationen für den interessierten Leser.

Warum eine neue Kodierung?

Die Einführung von Unicode nach ISO-10646 ist in der Computerwelt höchst sinnvoll und längst überfällig. Die bisherigen Kodierungen, d.h., wie die Zeichen zur Verarbeitung computergerecht aufbereitet wurden, waren nicht uneindeutig. Mit Unicode werden nun die unterschiedlichen sprachspezifischen Erweiterungen des bisher meist genutzten ISO8859-Standards und anderer Zeichenstandards durch einen einzigen, allumfassenden Standard ersetzt. Jedes in der Welt vorkommende Zeichen besitzt ein eindeutiges Äquivalent in der Kodierungsvorschrift!

Im Betriebssystem Fedora Core 1 wurde erstmals auf Unicode ISO-10646 im UTF-8-Format umgestellt. Näheres siehe unten. Das bedeutet eine gewisse Vorreiterrolle. Anderere Distributionen wie z.B. SuSE-Linux werden im nächsten Release ebenfalls umsteigen.

Was bedeutet der Wechsel auf Unicode?

Die Umstellung auf eine neue Kodierung erfordert von Betriebssystementwicklern, Anwendungsprogrammierern und Systemadministratoren große Anstrengungen, besonders, um auch die Nutzung von (älteren) in anderer Kodierung gespeicherten Texten zu gewährleisten. Obwohl innerhalb des neuen Systems die Behandlung von Textzeichen problemlos verlaufen sollte, können objektiv nicht alle Fälle behandelt werden.

Betroffen von der neuen Kodierung sind alle die Stellen, wo im Rechner auf Zeichen(ketten) zurückgegriffen wird:

- Textdateien (Beschreibungen in einem Textformat, Sourcecode, Scripts, HTML-Files, E-Mails, ...)
- Dateinamen
- Standardinput, -output, -error, Pipes
- Environmentvariable
- Cut&Paste-Buffer
- Remote-Zugriff auf andere Rechner: Telnet, ssh, Modem, Serial-Port-Verbindungen zu Terminalemulatoren

wobei für die gewöhnliche Nutzung besonders die ersten beiden Anstriche von Bedeutung sind.

Kritische Arbeitsweisen entstehen durch die Historie und das Nebeneinanderbestehen alter und neuer Systeme:

- wenn im neuen System weiterhin auf alte Dateien zurückgegriffen wird
- wenn weiterhin in einem alten und neuen System mit den gleichen Daten gearbeitet wird
 - das alte System erkennt die neue Kodierung nicht oder der zur Visualisierung nötige Zeichensatz ist nicht vorhanden
 - das neue System kann die alten Daten nicht ordentlich verarbeiten

Textdateien

Was funktioniert und was nicht, hängt von der Intelligenz der Programme ab, die mit den Daten umgehen:

- Es gibt viele Programme, die die Kodierung der Textdateien (ISO8859-x, UTF-8, UTF-16, ...) erkennen und dann entsprechend interpretieren. Die Desktopsysteme KDE und GNOME können gut mit der neuen Kodierungsart UTF-8 umgehen.
- Bestimmte Programme lassen sich die Kodierung durch Umgebungsvariable vorgeben oder suchen im Dokument nach einer expliziten Angabe (z.B. Webbrowser)
- Der problematischste Fall ist, dass eine Applikation eine feste Annahme über die zu verarbeitenden Daten trifft oder die neue Kodierung nicht versteht. Bei diesen Applikationen sollte man auf Alternativen ausweichen.

Dateinamen

Ein Vorteil ist, dass in Linux-/Unix-Systemen der Unicode in UTF-8-Kodierung benutzt wird. Das bedeutet, dass alle ASCII-Zeichen (das sind die Groß- und Kleinbuchstaben, Ziffern, einige Sonderzeichen wie Punkt, Komma, ... , aber nicht die Umlaute) die gleiche Kodierung von 1 Byte Länge besitzen wie bisher in ISO-8859-x!

Innerhalb von Dateinamen führen Nicht-ASCII-Zeichen in der alten ISO8859-Kodierung unweigerlich zu falschen Darstellungen in einem System mit UTF-8-Zeichen und umgekehrt. Eine Umbenennung ist hier unumgänglich.

Wo hole ich mir Rat bei Problemen?

- Hinweise und Problemlösungen bezüglich Unicode auf den vom URZ administrierten Rechnern sind in einer speziellen URZ-Unicode-FAQ-Liste <http://www.tu-chemnitz.de/urz/linux/faq/unicode.html> zusammengestellt.
- Wenn in der URZ-Unicode-FAQ-Liste das Problem noch nicht aufgenommen ist, dann bitte eine Mail an hilfe@hrz.tu-chemnitz.de mit dem Vermerk "UNICODE":

"Grundlagen" drei neue Dokumentationen erstellt: "Leitlinien", "Rechtliche Aspekte" und "Technologie". Diese Texte beinhalten die Zusammenfassung bisher an verschiedenen Stellen existierender Informationen unter dem Fokus der MONARCH-Technologie.

Neue Hardware

Als Server werden jetzt zwei Rechner mit folgender Charakteristik eingesetzt:

- AMD Dual Athlon MP 2200+
- 2GB RAM
- 2 Stromzuführungen
- GBit-Ethernet (level4-switching)

Dies ist die Basis für folgende Verbesserungen:

- Redundanz - bei Ausfall eines Servers muss MONARCH weiterlaufen
- Lastverteilung - durch Nutzung der level4-switching-Technologie ist eine Verteilung der Anforderungen auf zwei Server möglich
- Geschwindigkeit - Recherchieren sowie Archivieren profitieren wesentlich von der hohen (und dualen) Prozessorgeschwindigkeit

Neue Recherche-Technologie

Die Vorteile der neuen Recherche-Software Swish-E sind:

- Recherche nach einzelnen "Suchfeldern"
- inkrementelle Indizierung
- einfaches Eingabeformat für die Indizierung
- Perl-Schnittstelle
- keine externe Datenbank notwendig

Das sichtbare Ergebnis ist ein völlig neues und modernes Nutzungsinterface.

Neben der Umstellung auf Swish-E wurde auch die Volltext-Recherche verändert. Diese wird jetzt extern auf Google-Basis realisiert. Der Vorteil besteht in einer Nachnutzung des beim Betreiber der Suchmaschine vorliegenden Knowhows in Bezug auf die Indizierung von Volltexten unterschiedlicher Datenformate, wie pdf (mit den verschiedenen Versionen), postscript o.a.m.

Eine Konsequenz der MONARCH-Verbesserungen dürften häufiger archivierende Autoren festgestellt haben: Die automatische Archivierung wird jetzt stündlich aktiviert (werktags zwischen 7 und 18 Uhr), gegenüber früher einmal täglich. Somit wird die Archivierungsbestätigung wie auch die Reservierungsbestätigung in einem akzeptableren Zeitraum empfangen.

DINI-Zertifikat

DINI unterstützt den von Wissenschaftsrat sowie Hochschulrektorenkonferenz geforderten Aufbau von Dokumenten- und Publikationsservern.

Mit der Vergabe eines Zertifikats wird eine Qualitätskontrolle für Dokumenten- und Publikationsserver ermöglicht.

Die Erteilung eines Zertifikats ist an die Erfüllung von Mindestanforderungen geknüpft. Diese sind unter folgenden Zielstellungen in einem entsprechenden Papier enthalten:

- "detaillierte Beschreibung der Anforderungen an einen Dokumenten- und Publikationsserver"
- "Aufzeigen von Entwicklungsrichtlinien bei der Gestaltung von Servern und beim Austausch von Informationen über diese Server"
- "eine für Nutzende und Betreiber sichtbare Dokumentation der Einhaltung von Standards und Empfehlungen"

Die aktuelle Version von MONARCH

Die aktuelle Version, mit der die Zertifizierung beantragt wurde, ist MONARCH 2.3.1. Diese wurde am 1. Dezember 2003 in Betrieb genommen. Die Basisversion 2.3 war am 1. Oktober 2003 in die Nutzung genommen worden.

Was sind die wesentlichen Inhalte bzw. Änderungen in MONARCH 2.3 gegenüber der Vorgänger-Version 2.2.1?

- Portierung von MONARCH auf eine neue Hardwareplattform
Die seit 1997 verwendete Servertechnik auf Basis SUN-SPARC wurde ersetzt durch leistungsfähigere und ausfallsichere PC-Server-Technologie. Das bisher unter dem Betriebssystem Solaris laufende MONARCH wurde auf Linux umgestellt.
- Umstellung der Recherche-Technologie
Die unter Solaris auf glimpse basierende MONARCH-eigene Recherche wurde auf Swish-E umgestellt.
- Neues Layout der Web-Seiten
Das Layout der MONARCH-Seiten wurde gemäß der aktuellen Vorgaben seitens der TU und des URZ-Regulariums modernisiert.

Ab MONARCH 2.3.1

- Dokumentbezogene Nutzungsstatistik den Dokument-Metadaten hinzugefügt
Die schon länger existierende Nutzungsstatistik je Dokument (Anzahl Zugriffe) wurde der jeweiligen Dokument-Startseite hinzugefügt.
- Erweiterung der MONARCH-Beschreibung
Gemäß den Anforderungen an eine Zertifizierung wurden unter der Rubrik

problemkürzel " schicken.

Vertiefende Betrachtungen zur Zeichenbehandlung in Computern

Textzeichenkodierung in Computern

Textzeichen in einer für den Menschen lesbaren Form sind für die Verarbeitung im Computer ungeeignet. Sie müssen deshalb in eine computerverständliche Form gebracht werden. Konkret heißt das, es müssen für alle Textzeichen computerverarbeitbare Äquivalente (Kodes) vereinbart sein. Das gilt sowohl für die interne Verarbeitung im Computer selbst als auch für die Speicherung auf Datenträgern wie Festplatten, Disketten, CD's usw..

Man kann den Weg eines Zeichens vereinfacht so beschreiben:

- Eingabe:
 - über die Tastatur des Computers oder Analyse mittels Texterkennungssystem -> Erzeugen des internen Codes oder
 - Lesen eines Zeichens von einem Datenträger.
- Verarbeiten im Computer
- Ausgabe
 - Speichern in kodierter Form auf einem Datenträger oder
 - Visualisierung durch Rückkonvertierung des internen Codes in eine für Menschen lesbare Darstellung (z.B. für Drucker, Monitor)

Bei der Tastatureingabe kommt man durch die begrenzte Anzahl der Tasten um eine sprachabhängige Zuordnung der Tasten zu den Zeichen (genauer: ihrer internen Kodierung) nicht herum.

Die Visualisierung der internen Zeichenkodes auf Bildschirm und Drucker kann man sich vereinfacht so vorstellen, dass auf Grund des internen Codes ein darstellbares Element aus einer Zeichensatzdatei ausgewählt wird (ggf. unter Hinzufügung von Attributen wie Größe, Farbe, usw.).

Ein zentrales Problem stellt die Kodierung der Textzeichen dar. Leider entstanden mehrere Kodierungsvorschriften, die in unterschiedlichen Standards vereinbart sind und in den verschiedensten Systemen auch verwendet wurden und werden.

Wie wurden Texte bisher kodiert?

Wegen des geringen Leistungsvermögens ehemaliger Computer (Speicherkapazität, Rechengeschwindigkeit,...) hatte sich der Standard ISO8859 durchgesetzt:

- Mit Ausnahme von Zeichen aus dem asiatischen Raum wurde zur Kodierung eines Zeichens immer 1 Byte benutzt. Damit sind maximal 256 verschiedene Zustände darstellbar.

Die kleinste Einheit in einem Rechner ist das Bit, dass genau die 2 Zustände "0" oder "1" annehmen kann. Die kleinste adressierbare Einheit in einem Rechner ist das Byte. Es besteht aus 8 Bit. Da jedes Bit 2 Zustände einnehmen kann, existieren pro Byte $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 256$ unterschiedliche Kombinationsmöglichkeiten.

- Die 256 Kombinationen reichen nur zur Darstellung weniger Textzeichen. Deshalb hat man sich folgendermaßen geeinigt:
 - Die Bedeutung der ersten 128 Zeichen, dem sogenannte ASCII-Zeichensatz, ist festgelegt. Dafür werden nur sieben der acht Bits eines Bytes benutzt. Das erste Bit ist immer Null. Enthalten sind alle Textzeichen des englischen Sprachraumes, u.a. das komplette lateinische Alphabet (A-Z, a-z), die Ziffern und einige Sonderzeichen wie der Unterstrich, das Fragezeichen usw.
 - Die restlichen 128 Kombinationen (mit erstem Bit = 1) sind mehrfach belegt! Die Bedeutung des Codes kann nur in Verbindung mit der eingestellten Kodierungsvorschrift erkannt werden.

Beispiele für die unterschiedlichen Varianten des Standards ISO-8859-n:

Bitkombination	ISO-8859-1 westeuropäisch	ISO-8859-5 kyrillisch	ISO8859-7 neugriechisch	ISO-8859-15 Euroland
01100001	kleines "a"	kleines "a"	kleines "a"	kleines "a"
11100100	kleines "ä"	kleines russisches "ѐ"	kleines griechisches "Delta"	kleines "ä"
10100100	allg. Währungssymbol	Eurozeichen	unbelegt	Eurozeichen

- Der entscheidende Nachteil ist, dass zur korrekten Interpretation der Kodierungen zusätzliche Angaben (mindestens der zur Kodierung benutzte Standard) notwendig sind.
- Zur Visualisierung müssen für jede Version des Standards Zeichensatzdateien zur Verfügung stehen, damit eine korrekte Darstellung des internen Codes erfolgen kann
- In reinen Textdateien können z.B. Schriften mit gleichzeitig russischen und griechischen Schriftzeichen nicht dargestellt werden.
- Die Interpretation von Zeichenkodierungen mit erstem Bit = 1 in Dateinamen

MONARCH-Zertifizierung

Der Dokumenten- und Publikationsserver MONARCH wurde DINI-zertifiziert. Erläutert werden die Ziele dieser Zertifizierung. Weiterhin werden die mit der Version MONARCH 2.3 realisierten Verbesserungen vorgestellt.



Im Januar 2004 hat die TU Chemnitz als erste Einrichtung einen Antrag auf Zertifizierung eines Dokumenten- und Publikationsserver bei DINI e.V. gestellt.

Seit dem 17. März 2004 ist das Multimedia ONline ARchiv CHemnitz (**MONARCH**) berechtigt, dieses Logo zu tragen.

DINI

DINI ist die Deutsche Initiative für Netzwerk Information e.V., eine Initiative der folgende vier Partnerorganisationen angehören:

- Arbeitsgemeinschaft der Medienzentren der deutschen Hochschulen
- Deutscher Bibliotheksverband Sektion 4: Wissenschaftliche Universallibliotheken
- Zentren für Kommunikation und Informationsverarbeitung in Lehre und Forschung e.V.
- Fachgesellschaften

Mitglieder von DINI sind vor allem Bibliotheken, Rechenzentren, Medienzentren oder sonstige Serviceeinrichtungen.

Fazit

Aufbau und Betrieb eines NIDS im Campusnetz ist eine notwendige aber auch komplexe Aufgabe und löst in erster Linie **ein** Problem: Die Analyse des Netzwerkverkehrs auf sicherheitstechnischer Grundlage ermöglicht das Erkennen von Gefahren für den Betrieb des Campusnetzes und bietet damit der Netznutzern zusätzlichen Schutz durch reaktive Maßnahmen. Daraus resultierende noch offene Probleme:

- Die Information der betroffenen Nutzer vom Netz getrennter Endsysteme gestaltet sich äußerst schwierig, da dem URZ in den meisten Fällen die Kenntnis über die jeweils aktuellen Systemverantwortlichen fehlt.
- Der Aufwand im URZ, befallene Endgeräte zu isolieren und nach erfolgter Behandlung wieder in das Netzwerk zu integrieren, ist beträchtlich.
- Das Abschalten betroffener Endgeräte vom Netzwerk erschwert die Behandlung der Systeme durch den Nutzer.

An der Lösung dieser Probleme wird zur Zeit gearbeitet und wird Thema eines weiteren Artikels sein.

Thomas Schier, April 2004

hängt ebenfalls vom eingestellten Standard ab!

Unicode - eine eindeutige Kodierungsform von Zeichen

Mit den heute immens gestiegenen Speichergrößen und Verarbeitungsgeschwindigkeiten in Computern kann auf Zeichenkodierungen übergegangen werden, die nicht auf 1 Byte beschränkt sind.

Nach unabhängigen Vorarbeiten entwickeln seit 1991 zwei Gremien (Unicode-Consortium und eine Arbeitsgruppe der ISO) gemeinsam den Standard ISO-10646 für einen universellen Zeichensatz (Universal Character Set - UCS, Synonym: Unicode), der u.a. für j e d e s Zeichen in der Welt einen e i n d e u t i g e n Kode und offiziellen Namen festlegt:

- 32 Bit = 4 Byte werden per Definition zur Kodierung eines Zeichens genutzt
- In der Praxis reichen momentan 2 Bytes (mit mehr als 65000 Zeichen!) aus,
- Die ASCII-Zeichen behalten ihren ursprünglichen Kodewert
- Die restlichen in einem Byte darstellbaren Zeichen (erstes Bit = 1) entsprechen dem Standard ISO-8859-1
- Der Zeichenbegriff versteht sich allgemein, d.h. auch mathematische und technische Symbole, Runen, Blindenschrift usw. gehören dazu.

Beispiele für Zeichenkodierungen (Kodes in 2 Bytes dargestellt):

Zeichen	offizielle Bezeichnung nach ISO-10646	Bitkombination nach ISO-10646
kleines "a"	LATIN SMALL LETTER A	00000000 01100001
kleines "ä"	LATIN CAPITAL LETTER A WITH DIAERESIS	00000000 11100100
kleines russisches "ѣ"	CYRILLIC SMALL LETTER EF	00000100 01000100
kleines griechisches "Delta"	GREEK SMALL LETTER DELTA	00000011 10110100
Eurozeichen	EURO SIGN	00100000 10101100

Repräsentationsformen von Unicode

Die mit ISO-10646 festgelegte Kodierung sagt noch nicht aus, wie sie in einem Computer oder auf einem Datenträger abgebildet wird. Die einfachste Form wäre sicherlich, jedes Zeichen in 4 Byte zu kodieren. Das ist aber sehr uneffektiv, da die gebräuchtesten Zeichen unter Weglassen führender "0"-Bits in 1 oder 2 Byte darstellbar sind.

Unicode-Kodierung UTF-8

Aus Effektivitätsgründen (viele Texte bestehen aus Zeichen des ACSII-Zeichensatzes) und der Historie (viele Programme verarbeiten Texte byteweise, bestehende Textdateien sind byteorientiert) wurde als Kodierung **Universal Character Set Transformation Format 8**, kurz **UTF-8**, entwickelt.

Kodierungsvorschrift von Unicode in UTF-8:

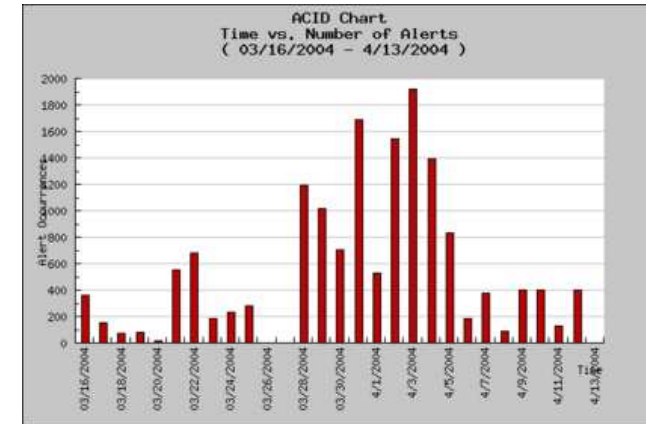
Um das Prinzip zu erkennen, werden der Übersichtlichkeit halber nur Zeichen mit 16 Bit Länge betrachtet. Das Verfahren ist natürlich auf 32 Bit erweiterbar.

Unicodezeichen von - bis	Bitfolge in UTF-8
00000000 00000000 - 00000000 01111111	0xxxxxxx
00000000 10000000 - 00000111 11111111	110xxxxx 10xxxxxx
00001000 00000000 - 11111111 11111111	1110xxxx 10xxxxxx 10xxxxxx

xxx sind die Bitpositionen des Unicodes

- Die Zeichen des ASCII-Zeichensatzes werden in einem Byte verschlüsselt. Das erste Bit des Bytes ist "0". Das entspricht genau der ASCII-Kodierung!
- Beginnt ein Byte mit mehr als einem "1"-Bit, so ist das der Beginn einer zusammengehörenden Bytefolge. Die Anzahl der "1"-Bits entspricht der Anzahl der notwendigen Bytes für die Verschlüsselung des Unicode-Zeichens. Die nachfolgend zugehörenden Bytes beginnen alle mit "10".
- Für maximal 16 Bit lange Unicodes werden maximal 3 Bytes benötigt
- Für mehr als 16 Bit lange Codes wird die Anzahl der benötigten Bytes erhöht: 1110xxxx 10xxxxxx 10xxxxxx 10xxxxxx usw.

Beispiele für die Verschlüsselung von Unicode-Zeichen in UTF-8:



In Abhängigkeit von den konfigurierten Regelsätzen können mehrere tausend Alerts pro Tag auftreten. Um die Snort-Datenbank schlank zu halten und damit ein performantes Nutzerinterface zu sichern, erfolgt mittels ACID auch eine Archivierung älterer Alerts in einer zweiten Datenbank.

Die aktive Alarmierung wird mittels des von Mattias Sandström entwickelten Scripts **ext-snort** (<http://www.ffoff.org/~mattis/linux/ext-snort/>) realisiert. Damit ist eine Einbindung in die aktuelle Big-Brother-Installation des URZ möglich und somit eine Alarmierung per e-mail.

Erste Erfahrungen

Alle genannten Komponenten laufen zur Zeit auf einem System. Performancetests zeigen, dass beim gegenwärtigen Verkehrsaufkommen (300 Mbit/s im 5-Minuten-Mittel) an diesem Sensor keine Paketverluste, sowohl auf dem Gigabit-Interface als auch auf Snort-Ebene, auftreten. Perspektivisch ist der Einsatz mehrerer Sensoren und die Trennung von Sensor- und Analyse-Funktion denkbar. Mit Hilfe dieser Snort-Lösung konnten mittlerweile eine Vielzahl von Unregelmäßigkeiten im Campusnetz erkannt werden. Hilfreich ist das System z. Zt. vor allem, um die Verbreitung von Würmern wie W32.Blaster, Mydoom, Welchia usw. einzudämmen, indem befallene Systeme erkannt und administrativ vom Netz genommen werden können. Rund 300 Systeme wurden so in letzter Zeit identifiziert - sicher nur die Spitze eines Eisberges von unzureichend administrierten Endgeräten. Die Reaktion auf Snort-Alarme erfolgt zur Zeit manuell. Automatismen wären denkbar, sind aber hinsichtlich möglicher Fehlalarme nicht unkritisch.

auf der Berkeley-Paketfilterbibliothek libpcap und filtert den Netzwerkverkehr anhand von Regelsätzen. Man kann zur Zeit aus ca. 2300 vorgefertigten Regeln für den konkreten Einsatz auswählen und mit einer umfangreichen Beschreibungssprache Regeln ändern oder eigene Regeln kreieren. Die Regeln definieren die Eigenschaften der Pakete, die von Snort untersucht werden sollen. Sie erlauben die Inspektion der IP-Adressen, Ports und auch des Inhaltes der Pakete. So erzeugt beispielsweise folgende Regel einen Alarm und protokolliert das Paket, wenn innerhalb von 60 Sekunden 60 TCP-Pakete mit gesetztem SYN-Flag und Destination-Port 135 von einer Quell-IP-Adresse versendet werden. Ein recht sicheres Indiz dafür, dass auf dem Rechner mit dieser Quell-IP-Adresse W32.Blaster heimisch geworden und auf der Suche nach weiteren Opfern ist.

```
alert tcp any any -> any 135 (flags: S; msg: "RPC DCOM exploit / Blaster Worm Attack";
threshold:type both, track by_src, count 60, seconds 60;
sid:1000008; priority:2;)
```

Wird also über den Regelsatz-Vergleich verdächtiger Verkehr erkannt, protokolliert Snort diesen "Alert" inklusive Datenpaket in Dateien im binären "Unified"-Format. Snort kann u.a. auch direkt im Klartext oder in eine Datenbank protokollieren. "Unified" als schnellster Ausgabemodus ist jedoch in Gigabit-Ethernet-Umgebungen auf jeden Fall vorzuziehen.

Die Weiterverarbeitung dieser binären Daten übernimmt **Barnyard** (<http://www.snort.org/dl/barnyard/>), ein Werkzeug, welches das "Unified"-Format lesen und in unterschiedlichen Formaten wieder ausgeben kann. Im URZ erfolgt die Barnyard-Ausgabe in eine MySQL-Datenbank.

Zur Analyse der Snort-Daten und zur Verwaltung der Datenbank kommt **ACID** (Analysis Console for Intrusion Databases) zum Einsatz. ACID

(<http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html>) bietet eine grafische Oberfläche zur Untersuchung und Pflege der Snort-Ergebnisse. ACID basiert auf PHP und benötigt zur Ausführung einen Apache-Webserver mit PHP-Unterstützung. Der Zugriff auf ACID erfolgt per Web-Browser. Über eine Menüstruktur stehen verschiedene Funktionen zur Verfügung. Diese reichen von der Verwaltung der Meldungen über die Nutzung umfangreicher Suchfunktionen, den direkten Einblick in die Pakete bis zur Erzeugung von Statistiken und Grafiken.

Zeichen	Bitkombination nach ISO-10646	Darstellung in UTF-8-Kodierung
kleines "a"	00000000 01100001	01100001
kleines "ä"	00000000 11100100	11000010 10000100
kleines russisches "f"	00000100 01000100	11010001 10110100
kleines griechisches "Delta"	00000011 10110100	11001110 10110100
Eurozeichen	00100000 10101100	11100010 10000010 10101100

Kodierung mittels UTF-8 wird in Linux-/Unix-Systemen eingesetzt.

Unicode-Kodierung UTF-16

UTF-16 benutzt als kleinste Einheit 16 Bit. Damit können viele Zeichen aus dem Unicodestandard direkt abgebildet werden. Beanspruchen Textzeichen mehr als 16 Bits, werden dafür Kombinationen aus 2x2 Bytes genutzt.

"Die meisten Windows-Oberflächen verwenden UTF-16" (lt. Hilfe- und Supportcenter von Microsoft). UTF-16 wird auch in Java genutzt.

Unicode-Kodierung UTF-32

Die Repräsentation von Unicode in 32 Bit ist definiert, aber ein Anwendungsfall ist zumindest dem Autor noch nicht bekannt.

Weitere Informationsquellen:

- "UTF-8 and Unicode FAQ for Unix/Linux" von Markus Kuhn
- "The Unicode HOWTO" von Bruno Haible
- Unicode Homepage

Gerd Heide, April 2004

Sicheres Programmieren mit PHP (Teil 2)

In den "Mitteilungen des URZ" 1/2004 haben wir bereits Hinweise zum sicheren Programmieren mit PHP gegeben. In diesem Artikel sollen nun weitere Problemfälle und Lösungsmöglichkeiten diskutiert werden.

Cross Site Scripting (XSS)

Unter "Cross Site Scripting" versteht man das "Einschleusen" von fremden HTML-Kode auf eine WWW-Seite. Damit kann ein Angreifer folgende Ziele verfolgen:

- Falsche Anzeige der WWW-Seite: Verunstaltung (z.B. unleserlich), Anzeige fremder Inhalte (Texte, Bilder)
- Ausspähen von Daten, z.B. Zugangsdaten (insb. Cookies), durch Einbringen von JavaScript-Kode
- Ausführen von fremden Programm-Kode (insb. JavaScript)

Betroffen von solchen XSS-Problemen sind prinzipiell alle PHP- und CGI-Programme, die externe Parameter (wie Formulareingaben) verarbeiten, z.B. Gästebücher, Such- oder Anmeldeformulare.

Betrachten wir zunächst ein schlechtes Beispiel:

```
<form action="..."><input name="name" ... > ... </form>
<?php # Gib den eingetippten Namen aus:
    print "Hallo " . $_REQUEST['name'];
?>
```

Unerwünschte Effekte treten ein, wenn der URL so aussieht:

```
.../script.php?name=<h1>Gross-und-fett!
```

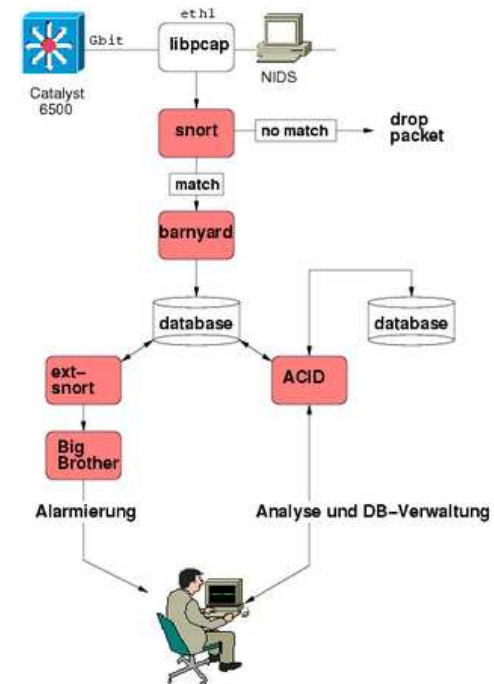
Dann würde durch die PHP-Anweisung ausgegeben:

```
Hallo <h1>Gross-und-fett
```

Das HTML-Tag <h1> würde die Ausgabe ganz empfindlich stören. Natürlich könnten auch noch "wüstere" HTML-Tags (z.B. Laden externer Bilder) oder gar JavaScript-Kode abgearbeitet werden.

Erinnern wir uns an den "Merksatz" aus dem Artikel in 1/2004 - der kann nicht oft genug genannt werden:

Aufbau und Funktionsweise



Verschiedene Softwarekomponenten realisieren die notwendigen IDS-Funktionen wie Paketmitschnitt, Signaturvergleich, Protokollierung/Archivierung, Analyse- und Verwaltungs-Schnittstelle sowie die Alarmierung. Sogenannte Sensoren sind zum Mitschneiden der Datenpakete notwendig. Die Platzierung dieser Sensoren im Netzwerk spielt dabei eine wichtige Rolle. Welcher Netzwerkverkehr ist interessant, vor oder nach dem Eintritt ins Campusnetz? Da wegen der anfänglich genannten Virus-/Wurm-Problematik neben dem Traffic aus dem Internet auch der lokale Verkehr interessiert, wurde ein erster Sensor am zentralen Backbone-Switch über Gigabit-Ethernet angeschlossen. Damit der angeschlossene Sensor sämtlichen über diesen Switch laufenden Verkehr empfangen kann, wird die vorhandene Switchfunktionalität des Port Mirroring ausgenutzt. Dabei werden per Kommando alle Datenpakete, welche den Switch "betreten", zusätzlich auf den Switchport gespiegelt, an dem der Sensor angeschlossen ist.

Sensor ist ein Doppelprozessor-System mit AMD MP2800+, 4 GByte RAM und 120 GByte Festplatte welches unter RedHat 7.3 betrieben wird.

Zentrale Komponente auf diesem System ist das von Martin Roesch entwickelte Snort (<http://www.snort.org/>). Snort ist in der Lage, als Sniffer, als Netzwerkprotokollant oder als Network Intrusion Detection System eingesetzt zu werden. Snort basiert

NIDS im Campusnetz

Seit Sommer vergangenen Jahres treten verstärkte Sicherheitsprobleme im Zusammenhang mit Viren und Würmern im Campusnetz auf. Um Anomalien und Angriffe im Netz erkennen und geeignet reagieren zu können, wurde ein erstes Network Intrusion Detection System in Betrieb genommen. Dieser Artikel beschreibt Aufbau, Funktionsweise und erste Erfahrungen mit diesem System.

Rückblick

+++ **16. Juli 2003:** Microsoft veröffentlicht das Security Bulletin MS03-026. Darin wird über eine Schwachstelle im RPC/DCOM-Dienst unter Windows NT/2000/XP informiert. Der Patch zum Schließen der Sicherheitslücke steht zur Verfügung.

+++ **12. August 2003, 9:30 Uhr:** Heise-Meldung: Der Wurm W32.Blaster, der die RPC/DCOM-Schwachstelle ausnutzt, verbreitet sich rasant im Internet.

+++ **12. August 2003, 14:20 Uhr:** Das URZ der TU Chemnitz sperrt vorsorglich alle UDP- und TCP-Ports, worüber sich W32.Blaster verbreiten soll, am GWiN-Router.

+++ **13. August 2003, 6:55 Uhr:** Der erste von W32.Blaster befallene Rechner der TU Chemnitz wird im Router-Log sichtbar.

Rasant verbreitet sich W32.Blaster im Campusnetz. In den folgenden Wochen und Monaten laufen immer wieder "Wurmwellen" durch das Netz.

+++ **13. April 2004:** W32.Blaster-Varianten finden reichlich Nährboden im Campusnetz, da zahlreiche Systeme weiterhin die RPC/DCOM- und weitere Schwachstellen aufweisen.

Dieser kleine Rückblick ist ein Beispiel für aktuelle Sicherheitsprobleme im Campusnetz und verdeutlicht, dass IP-Paketfilter oder auch komplexe Firewall-Systeme nur ein Baustein einer Sicherheitsstrategie sein können. Gegen das Einschleppen von Würmern und Viren per Laptop sind diese Mechanismen machtlos. Die versuchte Verbreitung von Schadroutinen über Standard-Dienste wie z.B. E-Mail nimmt immer größere Ausmaße an. Eine Vielzahl von Endgeräten im Campusnetz ist mangels Systempflege angreifbar.

Neben vorbeugenden Sicherheitsmaßnahmen wird es deshalb immer wichtiger, Mechanismen zum Erkennen von Angriffen und Anomalien im Netzwerk zu nutzen um geeignet reagieren zu können. An dieser Stelle setzen sogenannte NIDSs an. **NIDS** steht für **Network Intrusion Detection System**. Diese Systeme suchen im Netzwerkverkehr nach bestimmten festgelegten Mustern, die verdächtige oder bösartige Absichten erkennen lassen. Ihre Aufgabe besteht darin, bei erkannten Angriffsversuchen oder Anomalien Alarm zu schlagen und eventuell Gegenmaßnahmen einzuleiten. Neben einer Vielzahl kostenintensiver kommerzieller IDS-Produkte existieren mittlerweile auch Open-Source-Lösungen, die erstaunlichen Funktionsumfang bieten. Hier sticht insbesondere das Open-Source-NIDS **Snort** hervor. Auf Basis von Snort wurde im URZ ein erstes NIDS in Betrieb genommen.



Vertrauen Sie keinen Werten, die über Browsereingaben, den URL oder Cookies in das PHP-Skript gelangen. Alle externen Parameter, selbst wenn sie aus versteckten Feldern oder Auswahlmenüs kommen, müssen einer Plausibilitätsprüfung unterworfen werden, bevor sie im Programm verwendet werden.

Also betrachten wir für unser Beispiel auch den Wert für name kritisch:

```
<form action="..."><input name="name" ... > ... </form>
<?php # Gib den eingetippten Namen aus
      # evtl. HTML-Kode kodieren wir mit der Funktion htmlspecialchars:
      print "Hallo " . htmlspecialchars($_REQUEST['name']);
?>
```

Wenn in diesem Fall der eingetippte Name HTML-Kode enthält, wird er "unschädlich" gemacht. In obigem Beispiel wird ausgegeben:

```
Hallo &lt;h1&gt;Gross-und-fett
```

Und das bringt einen WWW-Browser nicht durcheinander - auch JavaScript wird so unschädlich gemacht. Alternativ können Sie die Funktion `strip_tags` einsetzen, um alle HTML- und PHP-Tags aus einer Zeichenkette zu entfernen.

SQL Injection

PHP eignet sich hervorragend, um mit WWW-Seiten Daten aus Datenbanken anzuzeigen oder zu ändern. Aber auch hier gilt es, Sicherheitsaspekte zu beachten, will man nicht böse Überraschungen erleben.

SQL Injection ist eine Technik, mit der böswillige Angreifer über WWW SQL-Kommandos erstellen oder existierende verändern, um versteckte Daten sichtbar zu machen, zu verändern oder zu löschen.

Betrachten wir wieder ein schlechtes Beispiel und ein Angriffs-Szenario:

```
<?php
#Datenbankabfrage an Hand einer ID aus dem URL
$id = $_REQUEST['id'];
$result = mysql_query("SELECT * from tabelle WHERE id=$id");
# Anzeige des Ergebnisse ...
?>
```

"Erwartet" wird ein URL der Form `.../script.php?id=42`, wodurch aus der Datenbank der Tabelleneintrag mit der `id=42` gelesen wird. Was aber, wenn der URL so geschrieben wird: `.../script.php?id=42;SHOW+TABLES` Die Pluszeichen wandelt der WWW-Server in Leerzeichen um, und der arglose `mysql_query`-Aufruf führt plötzlich noch eine zweite Aktion aus :-)

Auch in diesen Fall erinnern wir uns an den o.g. Merksatz: Wir müssen **alle Eingabewerte überprüfen**, auch wenn das etwas Mühe macht. PHP hält dafür aber eine Menge an Möglichkeiten bereit.

Wird als Wert eine Integer-Zahl erwartet, so können Sie dies prüfen mit der Funktion `is_numeric()` oder Sie setzen den Typ mittels `settype()` explizit auf `integer`:

```
<?php
#Datenbankabfrage an Hand einer ID aus dem URL
$id = $_REQUEST['id'];
settype($id, 'integer'); # Zwangsumwandlung in Zahl
$result = mysql_query("SELECT * from tabelle WHERE id=$id");
# Anzeige des Ergebnisse ...
}
?>
```

Werden als externe Parameter Zeichenketten erwartet, gerät die Prüfung meist etwas aufwändiger. Hier ist eine Prüfung auf plausible Werte angebracht (siehe Beispiel mit Dateinamen aus vorgegebener Menge in Artikel aus 1/2004). Zumindest sollten Sie Zeichen kodieren, die für SQL eine Sonderbedeutung haben: Semikolon, einfacher und doppelter Anführungsstrich usw.

```
<?php
#Datenbankabfrage an Hand eines eingetippten Suchwortes
$suchwort = $_REQUEST['wort'];

# Entferne rigoros alle Semikolons
$suchwort = ereg_replace(';','',$suchwort);
if (get_magic_quotes_gpc() == 0) {
    # Falls die System-PHP-Einstellungen es noch nicht automatisch tun,
    # kodiert die Funktion addslashes weitere SQL-Sonderzeichen:
    $suchwort = addslashes($suchwort);
}
$result = mysql_query("SELECT * from tabelle WHERE name like \"\$suchwort\"");
# Anzeige des Ergebnisse ...
}
?>
```

Am besten ist es, Sie lehnen dubios erscheinende Abfragen von vornherein ab:

```
<?php
#Datenbankabfrage an Hand eines eingetippten Suchwortes
$suchwort = $_REQUEST['wort'];

$laenge = strlen($suchwort);
# Lehne zu kurze/lange oder ein Semikolon enthaltende Suchwörter ab:
if ($laenge <= 0 || $laenge > 100 || ereg(';',$suchwort)) {
    print "Fehler im Suchwort ...";
    exit;
}
?>
```

Vor einer Abfrage von Daten muss zunächst die Verbindung zur Datenbank hergestellt werden, wozu neben Servernamen auch der Datenbank-Nutzer und das Passwort anzugeben ist, z.B. für MySQL:

```
<?php
#Datenbank-Verbindung herstellen:
# das @ vorm Funktionsaufruf verhindert die eingebaute Fehlermeldung
$db = @mysql_connect('mysql.hrz.tu-chemnitz.de', 'dbnutzer', 'geheim')
    or die ("Keine Verbindung zur Datenbank möglich");
?>
```

Hier haben wir das Problem, dass das Passwort des Datenbank-Benutzers im PHP-Skript stehen muss. Das birgt natürlich eine Gefahr, wenn es in falsche Hände gelangt. Als Grundregel beachten Sie: Verwenden Sie einen Datenbank-Benutzer, dessen Rechte entsprechend Ihrer Anwendung limitiert sind. Wenn Sie via PHP-Skript nur SELECT-Anweisungen ausführen, verwenden Sie den DB-Nutzer, der nur leseberechtigt ist.

Auf unseren zentralen WWW-Servern haben wir keine sichere Methode, ein böswilliges Auspähen durch andere PHP-Skripts auf dem gleichen Server zu verhindern. Wir setzen voraus, dass alle WWW-Autoren verantwortungsbewusst mit ihrer Aufgabe umgehen.

Für Anwendungen und Projekte mit höheren Sicherheitsanforderungen sind dedizierte WWW-Server vorzusehen. Wir erproben momentan ein solches System und planen, dies demnächst als Dienstleistung anzubieten.

Weitere Hinweise: <http://www.tu-chemnitz.de/docs/php/security.database.html>

Frank Richter, April 2004