



Carsten Eilers | [ceilers-it.de](http://ceilers-it.de)

# Schwachstellen-Scans im Web 2.0



## Vorstellung

- » Berater für IT-Sicherheit
- » Autor
  - » About Security
  - » Standpunkt Sicherheit
  - » und anderes ...
- » Schwachstellen-Datenbank
  - » „Security Aktuell“ auf [entwickler.de](http://entwickler.de)



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » Scans planen und durchführen



# Einführung



## Agenda

- » Einführung
- » **Suche nach Schwachstellen**
  - » Manuelle Suche
  - » Automatischer Scan
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » ...



## Manuelle Suche: 1. Informationen sammeln (1)

- » Ermitteln aller Ressourcen und Parameter
  - » Webcrawler
  - » Manuelle Suche
    - » Proxy zeigt POST- und Cookie-Parameter
    - » Rollen von unten nach oben durchgehen
- » Suche nach Informationen in den Seiten



## Manuelle Suche: 1. Informationen sammeln (2)

- » Suche nach nicht offensichtlichen Infos
  - » nicht verlinktes
  - » Informationen im Client
    - » Hidden-Felder
    - » Eingabenbeschränkungen und -prüfungen



## Manuelle Suche: 2. Parameter testen

- » Reaktion auf falsche Eingaben?
- » Fehlermeldungen auswerten
- » Auf XSS, SQL-Injection, Directory Traversal, Remote File Inclusion in PHP-Scripts, ... mit typischen und ggf. untypischen Mustern testen



## Manuelle Suche: 3. Programmlogik prüfen

- » Negative Werte im Webshop?
- » Durch Setzen bestimmter Parameter Überspringen von Schritten im vorgegebenen Ablauf möglich?
- » ...



## Automatischer Scan: 1. Informationen sammeln

- » Ermitteln aller Ressourcen und Parameter
  - » Webcrawler
  - » Aufzeichnung einer Sitzung über einen Proxy



## Automatischer Scan: 2. Parameter testen

- » Datenbank mit Signaturen bekannter Angriffe
- » Fuzzing
- » Test auf bekannte Schwachstellen
  - » Nessus



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » **Was findet ein Scanner?**
- » Was findet ein Scanner nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Was findet ein Scanner? (1)

- » Alle Schwachstellen, die auf bestimmten Testwert mit bestimmten Ausgaben antworten, deren Muster bekannt ist
  - » Reflektiertes Cross-Site-Scripting  
Testwert erscheint in der Ausgabe
  - » SQL-Injection-Schwachstellen  
die bekannte Ausgabe zurückliefern:  
Fehlermeldung, Wartezeit bei wait



## Was findet ein Scanner? (2)

- » Directory Traversal  
Inhalt bekannter Datei in der Ausgabe
- » Command Injection  
Spezifische Ausgabe eines eingeschleusten Befehls wird erkannt
- » Verzeichnis-Listen  
anhand ihrer Struktur



## Was findet ein Scanner? (3)

- » Nicht verlinkte Dateien mit bekannten oder erratenem Namen
- » Allgemein Schwachstellen, bei denen das Testmuster ganz oder teilweise in der Ausgabe erscheint



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner?
- » **Was findet ein Scanner nicht?**
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Was findet ein Scanner nicht? (1)

- » Nichts, was nicht mit Standardtests erkannt werden kann
  - » XSS/SQL-Injection abseits der Standardfälle
  - » SQL-Injection mit Nicht-Standard-Ausgaben
  - » persistentes Cross-Site-Scripting
  - » Directory Traversal ohne direkte Ausgabe
- » Nichts, was keine einheitliche Ausgabe liefert (fehlende Signatur)



## Was findet ein Scanner nicht? (2)

- » Fehler in der Zugriffskontrolle
- » Angriffe im Kontext der Anwendung
  - » Preisänderung, negative Mengen, ...
- » Angriffe auf die Programmlogik
  - » Überspringen von Schritten im Ablauf



## Was findet ein Scanner nicht? (3)

- » Designfehler:
  - » Schwache Passwortregeln,
  - » Preisgabe gültiger Benutzernamen,
  - » ...



## Was findet ein Scanner nicht? (4)

- » Preisgabe sensibler Informationen, z.B. Session-IDs in Logfiles
- » Session-Hijacking  
Erkennt evtl. vorhersagbare IDs, kennt aber die Bedeutung nicht



## Was findet ein Scanner nicht? (5)

- » Scanner arbeiten Syntax-basiert:
  - » Ausgabe wird analysiert
  - » bekannte Fehlermeldungen, HTTP-Status-Codes, Teile der Eingabe,... werden erkannt
- » Scanner kennen keine Semantik:  
Bedeutung von Parametern ist unbekannt  
Bsp: Warenkorb
  - » Preis darf nicht verändert werden
  - » Menge darf verändert werden, aber nicht  $< 0$



## Was findet ein Scanner nicht? (6a)

- » Scanner kennen keine Semantik
  - » `http://www.server.example/do.cgi?from=123&to=456&what=789`
  - » Was soll, was darf passieren?



## Was findet ein Scanner nicht? (6b)

- » Scanner kennen keine Semantik
  - » `http://www.server.example/do.cgi?from=123&to=456&what=789`
  - » Was soll, was darf passieren?

Bank:

*what* Euro vom Konto *from* auf Konto *to*



## Was findet ein Scanner nicht? (6c)

- » Scanner kennen keine Semantik
  - » `http://www.server.example/do.cgi?from=123&to=456&what=789`
  - » Was soll, was darf passieren?

Fahrplan:

*from*, *to* PLZ von Start und Ziel,  
*what* Beförderungsmittel



## Was findet ein Scanner nicht? (6d)

- » Scanner kennen keine Semantik
  - » `http://www.server.example/do.cgi?from=123&to=456&what=789`
  - » Was soll, was darf passieren?

Ganz was anderes?



## Was findet ein Scanner nicht? (7)

- » Scanner machen Dienst nach Vorschrift
  - » Nicht-Standard-Fehlermeldungen, eine ungewöhnliche Session-Verwaltung, Navigation oder Datenübertragung - auf ungewöhnliches kann nur schwer reagiert werden
  - » Keine Improvisation:  
Kein Umgehen teilweise wirksamer Filter, keine Kombination mehrerer Schwachstellen  
Bsp.: Blind-SQL-Injection mit Ausgabe in Datei, auf die anderswo zugegriffen wird



## Was findet ein Scanner nicht? (8)

- » Scanner handeln nicht intuitiv
  - » Jede verfügbare Funktion wird mit jedem Angriffsmuster getestet
  - » Erkennen nicht:
    - » Schwachstellen in mehrstufigen Prozessen
    - » Angriffe mit Änderung von Prozessschritten
    - » Angriffe mit Kombination verschiedener Parameter
  - » Ein Mensch erkennt, wo sich weitere Tests lohnen und wo nicht – dem Rechner bleibt nur 'Brute Force'



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » **Anforderungen an Scanner**
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Anforderungen an Scanner (1)

- » Authentifizierung und Session-Handling
  - » „Reinkommen und drin bleiben“
  - » „Bin ich noch drin?“
- » Keine Nebenwirkungen!
  - » Kein Löschen der Daten über den Admin-Bereich
  - » Kein Überschreiben von DB-Daten über SQL-Injection



## Anforderungen an Scanner (2)

- » Erkennen von Funktionen
  - » Viele Seiten, aber nur wenige Funktionen, z.B.
    - » Webshop: Viele Beschreibungen, aber nur wenige Funktionen
    - » Kalender, der eine Seite pro Tag erzeugt - bis in alle Ewigkeit
  - » Aufruf einer Seite über mehrere URLs
  - » Vom Scanner erzeugte neue Seiten
- » Viele gefundene Schwachstellen - davon viele Dubletten



## Anforderungen an Scanner (3)

- » Umgehen der Schutzfunktionen vor automatischer Nutzung
  - » Captchas sperren auch den Scanner aus



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » **Probleme im Web 2.0**
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Probleme im Web 2.0 (1)

» Ajax-Anwendungen haben interne Zustände:

- » Reihenfolge der Benutzerinteraktionen hat Auswirkung auf Ablauf des JavaScript-Codes
- » Automatische Requests zur Aktualisierung von z.B. Aktienkursen, Wetterdaten, Nachrichten ...
- » Änderungen während der Laufzeit:  
Vom Server gelieferter JavaScript-Code ändert das Verhalten der Seite



## Probleme im Web 2.0 (2)

» Welche Code-Teile sind wann aktiv?



## Probleme im Web 2.0 (3)

- » neue Datenstrukturen:
  - » XML, JSON, JavaScript-Arrays, proprietäre Strukturen, ...
- » neue Protokolle:
  - » SOAP, REST, ...



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » **Neue Anforderungen an Scanner**
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Neue Anforderungen an den Scanner (1)

- » Ermitteln aller verwendeten Ressourcen
  - » Simple Absuchen reicht nicht mehr, um alle Ressourcen zu finden
  - » Von JavaScript erzeugte Links müssen erkannt werden
  - » Nachgeladener JavaScript-Code kann Aufrufe an neue Ressourcen enthalten, die wieder Aufrufe an bisher unbekannte Ressourcen enthalten, ...
- » JavaScript-Weiche sperrt Scanner u.U. aus



## Neue Anforderungen an den Scanner (2)

### » Scan des Clients

- » Jeden möglichen Zustand ermitteln
- » Aufrufe in JavaScript erkennen
- » DOM-basiertes XSS und CSRF erkennen

» Web 1.0: Angriffe auf den Server

» Web 2.0: Angriffe auf Server und Client

» Auswerten von HTTP-Response reicht nicht mehr aus



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
  - » **Scan des Clients**
- » Wann/wieso scannen?
- » Scans planen und durchführen



## Scan des Clients (1)

### » Beispiel-Client

- » Webanwendung auf *beispiel.example*
  - » Anwendungs-Ressourcen, z.B. HTML-Seiten, Scripts usw.
  - » Proxy für XMLHttpRequests an andere Domains



## Scan des Clients (2)

### »4 Schritte:

» 1. Ermitteln der Technologien und Bibliotheken  
Haben die Schwachstellen?

» 2. Ermitteln der eingebundenen Daten und Skripte

Vertrauenswürdig: *beispiel.example*

Nicht vertrauenswürdig: Alle anderen

Nicht Vertrauenswürdiges muss geprüft werden



## Scan des Clients (3)

- » 3. Ermitteln der DOM-Zugriffe („DOM-Access-Points“)  
Jede Manipulation am DOM kann missbraucht werden
- » 4. Ermitteln von Ausführungslogik und Datenfluss  
Wo werden eingebundene Daten und Skripte während des Programmablaufs verwendet?



## Scan des Clients (4)

- » Beispiel:  
Nachrichtenseite, die verschiedene RSS-Feeds auswertet
  
- » 1. Ermittle Technologien und Bibliotheken
  - » JavaScript-Code aus HTML-Seite extrahieren und auswerten
  - » Welches Ajax-Toolkit?
    - » Fkt. für XMLHttpRequests, RSS-Feeds



## Scan des Clients (5)

» 2. Ermitteln eingebundener Daten und Skripte

- » Stelle, an der RSS-Feeds angefordert werden
  - » Benutzer gibt RSS-Feed *feed* an,
  - » Funktion *GetRSSFeed(feed)* ruft den über Proxy auf



## Scan des Clients (6)

### » 3. Ermitteln der DOM-Zugriffe

#### » Gesucht: z.B.

» *document.getElementById(.).innerHTML*

» *document.write()*

#### » Schreiben der Daten aus den RSS-Feeds in das DOM



## Scan des Clients (7)

»4. Ermitteln von Ausführungslogik und Datenfluss



## Scan des Clients (8)

Nicht vertrauenswürdiger  
Bereich

Vertrauenswürdiger  
Bereich

Browser

RSS-Feeds von  
verschiedenen  
fremden Servern



Proxy  
(mit JavaScript-Filter)



GetRSSFeed()



ExtractRSSFeed()



DOM

Ausführungslogik und Datenfluss



## Scan des Clients (9)

### »4. Ermitteln von Ausführungslogik und Datenfluss

- » Gefahr: Über RSS-Feeds eingeschleuster Script-Code
- » Schutz: JavaScript-Filter im Proxy
  
- » Mögliche Gefahr: Eingefügte Links
- » Schutz: Bisher nicht geprüft



## Scan des Clients (10)

- » Scan des Clients ist aufwendig
- » Mehrere Quellen in einem Mashup verschärfen das Problem



## Scan des Clients (11)

» Nachrichtenseite in Mashup auf einer anderen Seite:

- » JavaScript in RSS-Feeds kann ausgefiltert werden
- » JavaScript aus der Newsseite darf nicht ausgefiltert werden



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » **Wann/wieso scannen?**
- » Scans planen und durchführen



## Wann/wieso scannen (1)

- » Ergebnisse des Scans stark vom Testobjekt abhängig
- » Scanner erkennen Standardfälle - und stoppen damit Standard-Angreifer
- » Scanner finden Standardfälle schneller als ein Mensch



## Wann/wieso scannen (2)

» Scanner liefern einen Überblick über die Anwendung

» Scanner zeigen verdächtige Bereiche, die dann manuell näher untersucht werden



## Agenda

- » Einführung
- » Suche nach Schwachstellen
- » Was findet ein Scanner, was nicht?
- » Anforderungen an Scanner
- » Probleme im Web 2.0
- » Neue Anforderungen an Scanner
- » Wann/wieso scannen?
- » **Scans planen und durchführen**



## Scans planen und durchführen (1)

- » Was soll gescannt werden?
  - » Welcher Scanner ist geeignet?
  
- » Scan von innen oder von außen?
  - » Interner Scan findet mehr Schwachstellen als externer Scan, da die Firewall einiges ausfiltert
  - » Anwendung muss ohne Firewall sicher sein, also muss die durchtunnelt werden



## Scans planen und durchführen (2)

- » Authentifizierung oder HTTPS erforderlich?
  - » Scanner muss Authentifizierung unterstützen
  - » HTTPS kann ggf. getunnelt werden (z.B. Stunnel)



## Scans planen und durchführen (3)

### » Hinweise zum Scan

- » Default-Einstellung reicht für erste Tests
- » Anpassung an die eigene Anwendung verbessert die Ergebnisse
- » Risiken und Nebenwirkungen berücksichtigen  
Ggf. riskante Bereiche vom Scan ausnehmen



## Scans planen und durchführen (4)

### » Hinweise zum Scan

- » Protokollierung: Je mehr, desto besser  
Bei vielen False Positives einschränken
- » Beachten, welche Schwachstellen Scanner finden können und welche nicht
- » Jede gefundene Schwachstelle manuell überprüfen



## Scans planen und durchführen (5)

### » Hinweise zum Scan

- » Scanner sind keine Diplomaten, sondern Barbaren - sie treiben IDS in den Wahnsinn!



## Fazit

- » Scanner erkennen nicht alles
- » Im Web 2.0 noch unzuverlässiger
- » Manuelle Nacharbeit nötig
- » Trotzdem Arbeitserleichterung
- » Ein Scan ist besser als gar kein Test  
ABER:  
Vorsicht vor falschem Sicherheitsgefühl



## Fragen?



**Vielen Dank...**

... für Ihre Aufmerksamkeit

Material und Links auf  
[www.ceilers-it.de/konferenzen/](http://www.ceilers-it.de/konferenzen/)

