HÄRTUNG VON WEB-APPLIKATIONEN MIT OPEN-SOURCE-SOFTWARE

Münchener Open-Source-Treffen,
Florian Maier, 23.05.2014

ÜBER MICH

- 34 Jahre, verheiratet
- Open Source Enthusiast seit 1997
- Beruflich seit 2001
- Sicherheit, Softwarearchitektur, Automatisierung

BEKÄMPFUNG DER AUSWIRKUNGEN

- Warum reicht Ursachenbekämpfung alleine nicht aus?
  Fremdcode!

- Patchen ist manchmal einfach zu kostenintensiv oder
  überhaupt nicht möglich

- Menschliches Fehlerpotential
  Minimierung des „Windows of Exploitation“ Security
  Principal: „Defense in depth“

DEFENSE IN DEPTH

- Implementiere Sicherheit in allen möglichen Schichten (Netzwerk, Anwendung, Webserver, Datenbank, Betriebssysteme, ...).

- Falls die Sicherheit in einer Schicht versagt, greifen die Mechanismen der anderen Schichten

- Netzwerkbasierte Firewall => gehärtetes DMZ System => Web Applikation Firewall => sichere Web Applikation => Monitoring

- Proaktives Testing

DIE OWASP TOP 10 (2013)

- Hitiste der 10 kritischsten Sicherheitsrisiken in Web Applikationen
- Für jedes Risiko werden anschauliche Beispiele und Gegenmaßnahmen aufgeführt
- Creative Commons Attribution-Share Alike 3.0 Lizenz

DIE OWASP TOP 10 (2013)

A1 - Injection

A2 - Broken Authentication and Session Management

A3 - Cross-Site Scripting (XSS)

A4 - Insecure Direct Object References

A5 - Security Misconfiguration

A6 - Sensitive Data Exposure

A7 - Missing Function Level Access Control

A8 - Cross-Site Request Forgery (CSRF)

A9 - Using Components with Known Vulnerabilities

A10 - Unvalidated Redirects and Forwards

TOOLS OF THE TRADE

modsecurity
Open Source Web Application Firewall

OSSEC

tripwire

ICINGA

arachni
web application security scanner framework

w3af

MODSECURITY

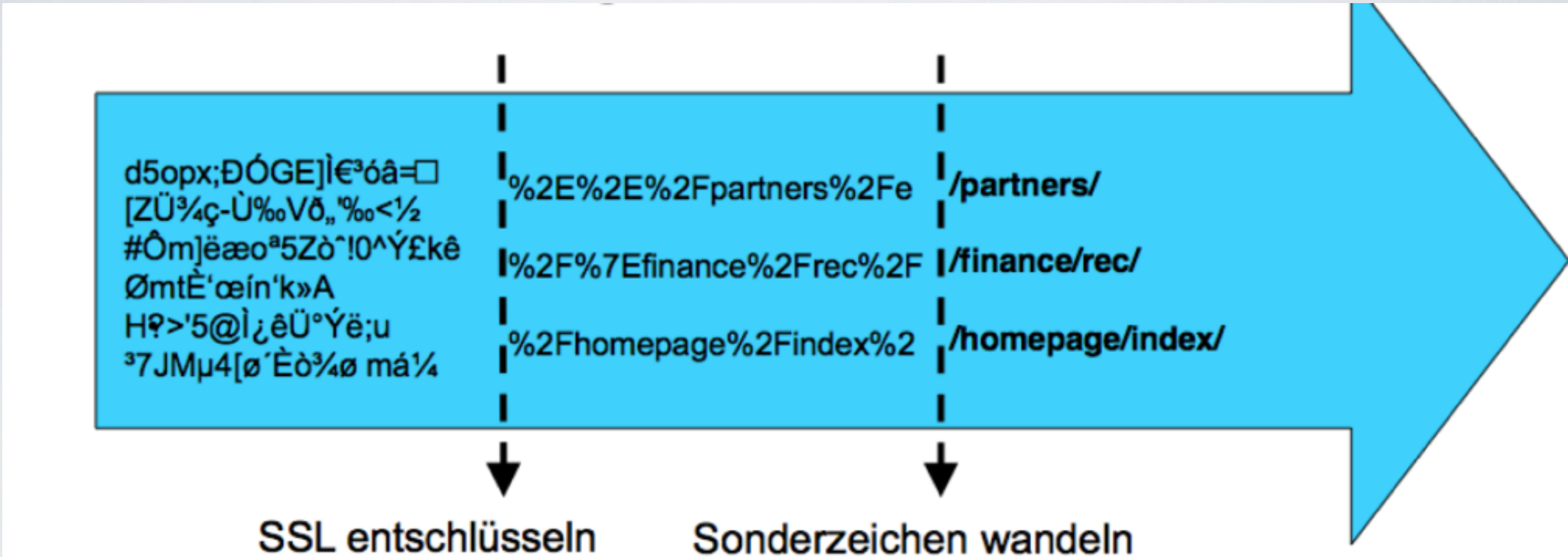
modsecurity
Open Source Web Application Firewall

- Erste Version 2002 veröffentlicht
- Open Source und kommerzielle Version
- Verfügbar für Apache HTTP Server, IIS und NGINX
- Aho-Corasick-Algorithmus (fgrep, snort)

MODSECURITY

- Schutz vor bekannten Angriffen über Signaturen PHP-Würmer,Exploits, usw.
- Updates via “Core Ruleset”
- Vor unbekannten Angriffen über generische Mustererkennung (Zero Day Protection)
- Effektiv v.a. gegen SQL Injection, XSS, ...

FUNKTIONSWEISE EINER WAF



FUNKTIONSWEISE EINER WAF

- Eine WAF arbeitet auf Ebene HTTP/HTML (Kontext der Applikation), Wichtige Abgrenzung zu IDS/IPS!

- Extrahieren von URLs, Parametern für GET und POST-Requests

- White- und Blacklisting

- Ausgabefilterung

KONTROLLE VON PARAMETERN

- Whitelisting
    - Beschreibung gültiger Wertebereiche
- Blacklisting
- Anwendung von Signaturen und Mustererkennung
- Schutz von Session Daten und kontextabhängigen Daten
- Individuell je Parameter konfigurierbar
- Ggf. Ausnahmen für einzelne Prüfungen bei bestimmten Parametern
- False Positives

WHITELIST VS. BLACKLIST

- Whitelisting bietet höchstmöglichen Schutz, ist aber sehr aufwändig zu konfigurieren

- In der Praxis hat sich ein solider Basisschutz aus Blacklisting und generischem Whitelisting bewährt

- Ggf. Ergänzung durch spezifisches Whitelisting über dynamische Statusverfolgung oder “Flows” in kritischen Bereichen

ERSTE INBETRIEBNAHME

- Passiver Modus ermöglicht Anwendung der Security Policy und Logging von Sicherheitsvorfällen => Kein aktives Blockieren
- Ideal für die ersten Stunden/Tage der Inbetriebnahme
- Eventuelle False Positives sind kein Problem
- Besonders wichtig, wenn produktive Anwendung nachträglich mit WAF-Schutz versehen wird
- Deployment Prozesse müssen angepasst werden

GRENZEN EINER WAF

- Erkennung von Logikfehlern
  z.B. falsch implementiertes Berechtigungsmodell

- Fehlerhafte Implementierung von Sicherheit auf Client Seite
  z.B. Berechtigungsmodell wird in JavaScript geprüft

- Ausnutzung von Browserschwachstellen z.B. “Clickjacking”

- Kombination mit weiteren Tools notwendig

OSSEC

- Host-based Intrusion Detection System mit Fokus auf Loganalyse, Integritätsprüfung, Windows Registry Monitoring, Rootkit Erkennung, Echtzeit Benachrichtigungen

- Erste Version um 2006 veröffentlicht

- Besonderheit “Active” response

- Verfügbar u.a. für Linux, OpenBSD, FreeBSD, MacOS, Solaris und Windows

- Lizenz GPL-2.0+

OPEN SOURCE TRIPWIRE

- Fokus auf Integrität der Daten
- Erste Version um 1992, veröffentlicht
- Seit 2000 als Open Source und kommerzielle Version
- GNU General Public License version 2.0 (GPLv2)

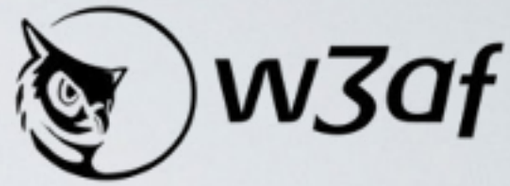
SNORT

- Network Intrusion Prevention / Detection System
- Regelbasierte Konfiguration (Signatur, Protokoll und Normabweichung)
- De facto Standard
- Erste Version ca. 2002 veröffentlicht
- Open Source und kommerzielle Lizenzmodelle
- GNU GPL v.2

ICINGA

- Monitoring System basierend auf Nagios
- Viele Plugins, leicht erweiterbar
- Erste Version 2002 veröffentlicht (2009)
- GPL-2.0+

W3AF



- Framework und Scanner zugleich
- Prüft mehr als 200 Schwachstellen wie SQL Injection, Cross-Site Scripting, Einfache Zugangsdaten, Unzureichendes Fehlerbehandlung, usw.
- Erste Version 2006 veröffentlicht
- GPLv2.0

ARACHNI

arachni
web application security scanner framework

- Open Source Ruby Framework
- Berichte und Scheduled Scans
- Verteilte Architektur (Dispatcher Instanzen)
- Erste Version 2010 veröffentlicht
- Apache License Version 2.0

REFERENZEN

- https://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013
- http://w3af.org/
- http://www.arachni-scanner.com/
- https://www.modsecurity.org/
- http://www.snort.org/
- http://www.ossec.net/
- http://sourceforge.net/projects/tripwire/

DANKE! FRAGEN?