"The Core Rule Set":
Generic detection of application layer attacks

Ofer Shezaf
OWASP IL Chapter leader
CTO, Breach Security
About Breach Security, Inc.

- The market leader in web application security
- Headquarters in Carlsbad, CA, with R&D Center in Herzliya, Israel and London, UK.
- Sales offices in Boston, Austin, Chicago, London and Tel-Aviv
- Experience with Web security solutions since 1999
- Managed by an experienced group of security professionals
- 55 Employees
ModSecurity Technology

- An Open Source Application Firewall.
- The most popular WAF in the world with more than 10,000 installations.
- An Apache module. Supports either embedded or reverse proxy deployment.
- Advanced Rules Language. A Swiss Army knife for the experienced user.
- Also available for free:
  - Core Rule Set
  - An entry level console
- Professionally Supported by Breach Security.
ModSecurityPro™ M1000

- Hardened reverse-proxy Web application firewall appliance based on ModSecurity technology, and additionally:
  - Packaged tested and certified by Breach Security.
  - Web based management.
  - Enhanced Rule Set tailored for specific applications.
  - Support packaged rule sets such as PCI compliance.
- Plug-and-play Web application security for organizations of any size.
- Highly competitive pricing
Top Notch Web App Sec Expertise

- **Ivan Ristic, Chief Evangelist**
  - Creator of ModSecurity
  - Leads WASC’s Web Application Firewall Evaluation Criteria project
  - Written Apache Security for O'Reilly.

- **Ofer Shezaf, CTO**
  - WASC Board Member,
  - OWASP IL chapter leader
  - Leader of WASC Web Hacking Incidents Database Project
  - Israeli National Security Background

- **Ryan Barnett, Directory of Training:**
  - SANS and Foundstone instructor
  - Written “Preventing Web Attacks with Apache” for O’Reilly
  - Leads WASC’s Distributed Honeypot Project
Web Application Firewalls vs. Intrusion Prevention Systems
Multiple Deployment Modes

- **In-Line mode**
- **Embedded mode**
- **Out of line mode**
Three Protection Strategies for WAFs

1. **External patching**
   - Also known as "just-in-time patching" or "virtual patching".

2. **Positive security model**
   - An independent input validation envelope.
   - Rules must be adjusted to the application.
   - Automated and continuous learning (to adjust for changes) is the key.

3. **Negative security model**
   - Looking for bad stuff,
   - Mostly signatures based.
   - Generic but requires some tweaking for each application.

IPS?
Virtual Patching

- Testing reveals that the login field is vulnerable to SQL injection.
- Login names cannot include characters beside alphanumerical characters.
- The following rule will help:

```xml
<LocationMatch "^/app/login.asp$">
  SecRule ARGS:username "!\w+$" "deny,log"
</LocationMatch>
```
Positive security

- The same, but for every field in every application

```xml
<LocationMatch "^/exchweb/bin/auth/owaauth.dll$">
    SecDefaultAction "log,deny,t:lowercase"
    SecRule REQUEST_METHOD !POST
    SecRule ARGS:destination "URL" "t:urlDecode"
    SecRule ARGS:flags "[0-9]{1,2}" 
    SecRule ARGS:username "[0-9a-zA-Z]{256,}" 
    SecRule ARGS:password ".{256,}" 
    SecRule ARGS:SubmitCreds "!Log.On"
    SecRule ARGS:trusted "!(0|4)"
</LocationMatch>
```

- Very hard to create, requires learning by:
  - Monitoring outbound traffic (match input to web server request)
    - Caveats: JavaScript, Web Services
  - Monitoring inbound traffic (normal behavior):
    - Caveats: Statistics, attacks in learning period.
Positive Security
Negative Security

An IPS, but:

- Deep understanding of HTTP and HTML
  - Breaking up to individual fields: headers, parameters, uploaded files.
  - Validation of field attributes such as content, length or count
  - Correct breakup and matching of transactions and sessions.
  - Compensation for protocol caveats and anomalies, for example cookies.

- Robust parsing:
  - Unique parameters syntax
  - XML requests (SOAP, Web Services)

- Anti Evasion features:
  - Decoding
  - Path canonizations
  - Thorough understanding of application layer issues: Apache request line delimiters, PHP parameter names anomalies.

- Rules instead of signatures:
  - Sessions & state management, Logical operators, Control structures.
IDPS signatures vs. WAF Rules

**Signatures:**
- Simple text strings or regular expression patterns matched against input data.
- Usually detect attack vectors for known vulnerabilities, while web applications are usually custom made.
- Variations on attack vectors are very easy to create

**Rules:**
- Multiple operators and logical expressions: Is password field length > 8?
- Selectable anti-evasion transformation functions.
- Control structures such as IF:
  - Apply different rules based on transactions.
- Variables, Session & state management:
  - Aggregate events over a sessions.
  - Detect brute force & denial of service.
  - Audit user name for each transaction
The Core Rule Set

- modsecurity-core-rules_2.0-1.1.1 (blocking).zip
- modsecurity_crs_10_config.conf
- modsecurity_crs_20_protocol_violations.conf
- modsecurity_crs_30_http_policy.conf
- modsecurity_crs_35_bad_robots.conf
- modsecurity_crs_40_generic_attacks.conf
- modsecurity_crs_45_trojans.conf
- modsecurity_crs_50_outbound.conf
- modsecurity_crs_55_marketing.conf
Detection of generic app layer attacks

- Core Rule Set available for ModSecurity at:
  - Probably translatable to any App Firewall

- Benefits from ModSecurity features:
  - Anti Evasion
  - Granular Parsing

- Detection Mechanisms:
  - Protocol Validation
  - Generic Attack Signatures
  - Known Vulnerabilities Signatures
  - More…
Protocol Validation
Protocol Violations

- Protocol vulnerabilities such as Response Splitting, Request Smuggling, Premature URL ending:
  - Content length only for none GET/HEAD methods
  - Non ASCII characters or encoding in headers.
  - Valid use of headers (for example, content length is numerical)
  - Proxy Access

- Attack requests are different due to automation:
  - Missing headers such as Host, Accept, User-Agent.
  - Host is an IP address.
Protocol Policy

- Policy is usually application specific:
  - Some restrictions can usually be applied generically.
  - White lists can be build for specific environments.

- Items that can be allowed or restricted:
  - Methods - Allow or restrict WebDAV, block abused methods such as CONNECT, TRACE or DEBUG.
  - File extensions – backup files, database files, ini files.
  - Content-Types (and to some extent other headers)

- Limitations on sizes:
  - Request size, Upload size,
  - # of parameters, length of parameter.
Application Layer Signatures
Snort signature for Bugtraq vulnerability #21799

Exploit:

```
/cacti/cmd.php?1+1111)**/UNION/**/SELECT/**/2,0,1,1,127
.0.0.1,null,1,null,null,161,500, proc,null,1,300,0, ls -l
la > ./rra/suntzu.log,null,null/**/FROM/**/host/**/+11111
```

Snort Signature:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(
    msg:"BLEEDING-EDGE WEB Cacti cmd.php Remote Arbitrary SQL Command Execution Attempt";
    flow:to_server,established;
    uricontent:"/cmd.php?"; nocase;
    uricontent:"UNION"; nocase;
    uricontent:"SELECT"; nocase;
    reference:cve,CVE-2006-6799; reference:bugtraq,21799;
    reference: cve,CVE-2006-6799; reference: bugtraq,21799;
    type: web-application-attack; classtype: web-application-attack;
    sid:2003334; rev:1;
)
```

- **UNION and SELECT are common English words. So is SELECTION**
- **Does the application accepts POST requests?**
- **Signature built for specific exploit**
- **An SQL injection does not have to use SELECT or UNION**
Case study: 1=1

- Classic example of an SQL injection attacks. Often used as a signature.
- But, can be avoided easily using:
  - Encoding: 1%3D1
  - White Space: 1 =%091
  - Comments 1 /* This is a comment */ = 1
- Actually not required at all by attacker.
  - Any true expression would work: 2 > 1
  - In some cases, a constant would also work. In MS-Access all the following are true: 1, “1”, “a89”, 4-4.
- No simple generic detection
Generic application layer signatures

- Detect attack indicators and not attack vectors:
  - `xp_cmdshell`,
  - `<`, single quote - Single quote is very much needed to type O'Brien
  - `select`, `union` – which are English words

- **Aggregate indicators to determine an attack:**
  - Very strong indicators: `xp_cmdshell`, `varchar`,
  - Sequence: `union .... select, select ... top ... 1`
  - Amount: `script`, `cookie` and `document` appear in the same input field.
  - Sequence over multiple requests from the same source.
Back to Bugtraq vulnerability #21799

The Core Rule Set Generic Detection

Supports any type of parameters, POST, GET or any other

Common evasion techniques are mitigated

Every SQL injection related keyword is checked

SQL comments are compensated for

Se:POST_FILENAME|ARGS|ARGS_NAMES|
REQUEST_HEADERS|!REQUEST_HEADERS:Referer \n
"(?:\b(?::s(?::elect\b(?::[1,100]}?\b(?::length|count|top}\b{1,100 }?\bfrom|from\b{1,100}?\bwhere}).*?\b(?::d(?::ump}\b.*\bfrom|ata_type})(?:
to_(?:numbe|cha|inst)r)|p_(?:addextende|pre|sqlexe)c(?:oacreat|prepa|execute(?::sql)?|makewebtask)|ql_(?:.... .... \n
Back to Bugtraq vulnerability #21799

Virtual Patching

```
<LocationMatch :"/cmd.php$">
  SecRule QUERY_STRING "^[\d\s]*$" "deny,log"
</LocationMatch>
```

Or

```
SecRule REQUEST_FILENAMES :"/cmd.php$" "deny,log"
```

Parameters Must Be Numeric

Actually script should not be run remotely

Simpler, isn’t it?
Odds and Ends
Malicious Robots

- Detection of malicious robots:
  - Unique request attributes: User-Agent header, URL, Headers
  - Black list of IP addresses
- Not aimed against targeted attacks, but against general malicious internet activity:
  - Offloads a lot of cyberspace junk & noise
  - Effective against comment spam.
  - Reduce event count.
- In addition:
  - Detection of security scanners
  - Detection of non malicious robots (such as search engines).
  - Confusing security testing software (HTTPrint)
Trojans and Viruses

- Major problem at hosting environments
  - Uploading is allowed.
  - Some sites may be secure while others not.

- Generic detection:
  - Check upload of Viruses.
  - Check upload of Trojans – AV software is not very good at that.
  - Check for access to Trojans:
    - Known signatures (x_key header)
    - Generic file management output (gid, uid, drwx, c:\)
Error conditions

- Last line of defense if all else fails
- Provide feedback to application developers
- Important for customer experience
- Makes life for the hacker harder
Future Plans
- **Session bases protection:**
  - Brute force detection.
  - Scanner and automation detection based on rate and result code.
  - Anomaly scoring.

- **XML protection:**
  - Schema validation for known XML payloads, such as SOAP.
  - Context based signature check in XML using XPath.
Thank You!

Ofer Shezaf
ofers@breach.com