ModProfiler: Defending Web Applications from 0-day Attacks

Signatures out. Traffic profiling in.

Ivan Ristić and Ofer Shezaf, Breach Security, BlackHat August 2008

About Us

Ivan Ristić and Ofer Shezaf, Breach Security

- Web application firewall experts:
  - Ivan created ModSecurity, the most popular WAF on earth, and wrote “Apache Security” by O’reilly.
  - Ofer created WebDefend, the first and most advanced behavioral based WAF.
- Web application security leaders:
  - Officers, the Web Application Security Consortium (WASC)
  - Lead OWASP chapters in London & Israel respectively.
- Open source & community projects:
  - Ivan leads the WASC Web Application Firewall Evaluation Criteria (WAFEC) project.
  - Ofer leads the WASC Web Hacking Incidents Database (WHID) project.

Breach Security

Technology Leaders

- Breach is a leading WAF vendor.
- Sole focus is web application security since 1999.
- Managed by an experienced group of security professionals.
- Best application security DNA in the industry. We wrote the books.
- Home to ModSecurity, the open source WAF.

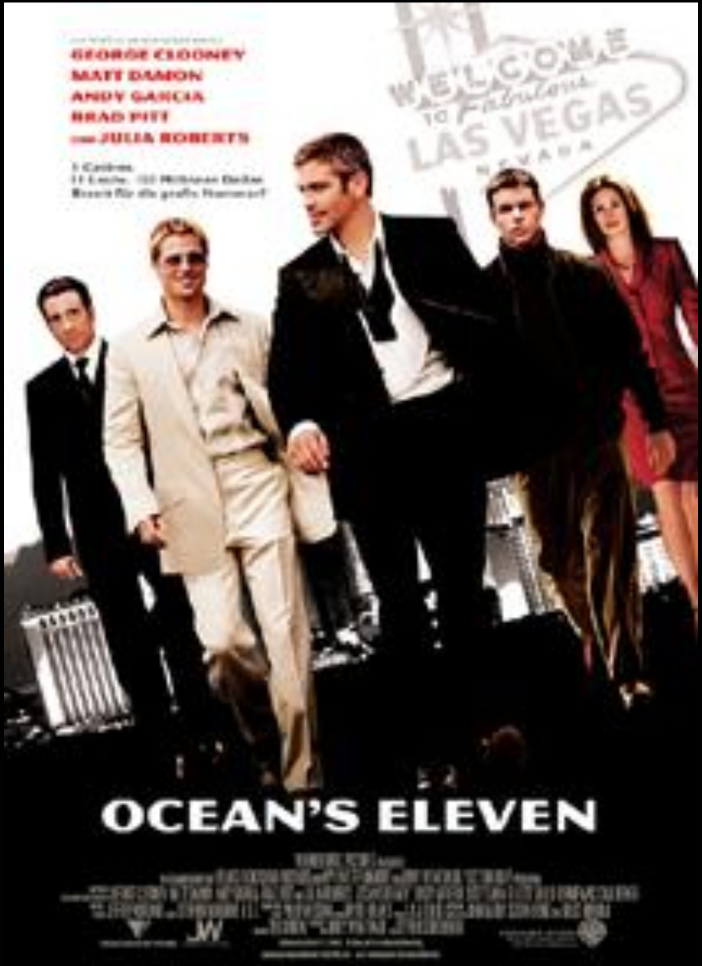
The Complete Guide to Securing Your Apache Web Server
Apache Security
Ivan Ristić
PREVENTING WEB ATTACKS WITH APACHE
RYAN C. BARNETT

http://www.modsecurity.org/projects/modprofiler

Part I: The Problem Domain

Why are Web Applications Inherently Insecure?

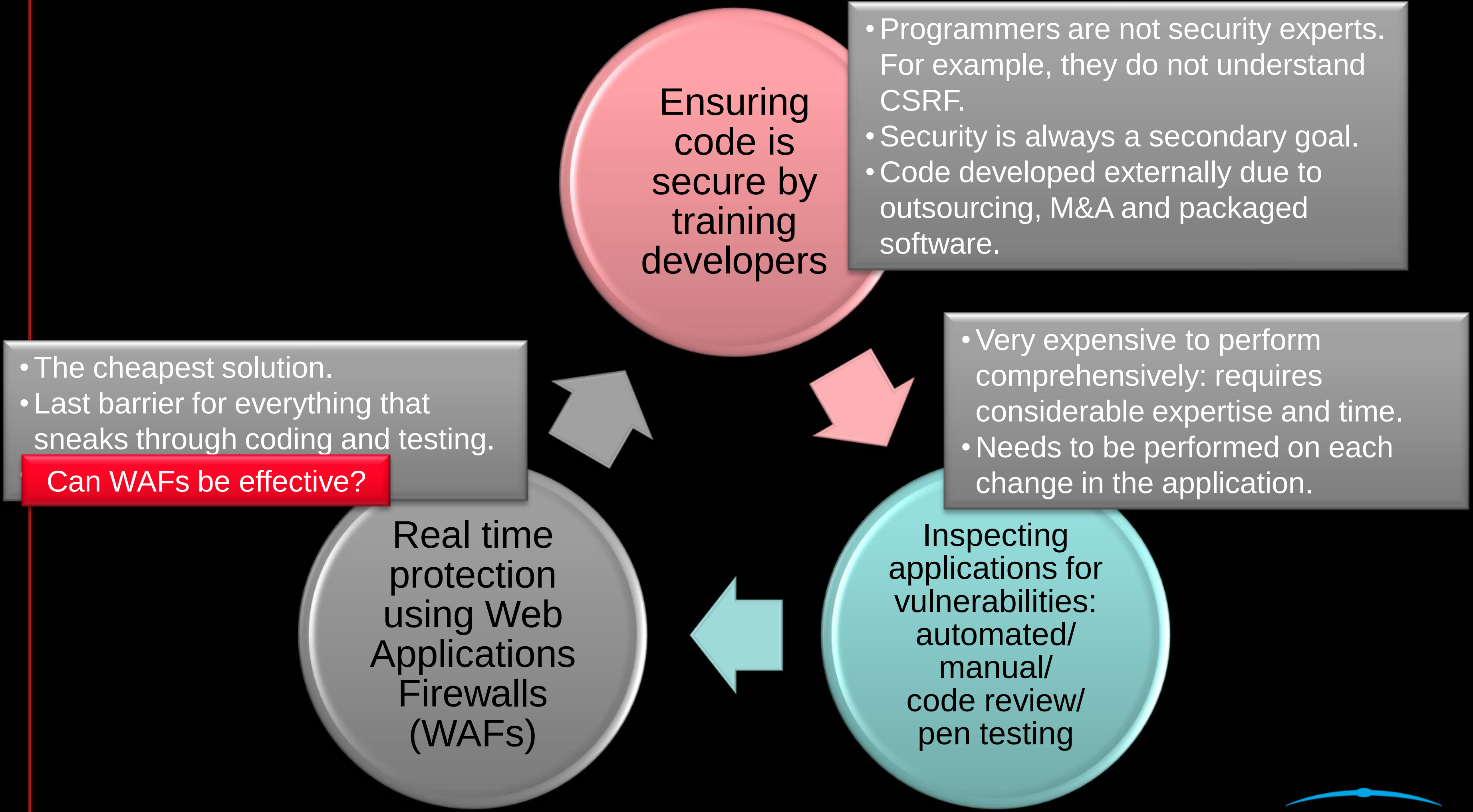
- Applications are **vulnerable**:
  - Unique, each one exposing its own vulnerabilities.
  - Change frequently, requiring constant tuning of application security.
  - Complex and feature rich with the advent of AJAX, Web Services and Web 2.0.
- Applications are **threatened**:
  - New business models drive “for profit” hacking.
  - Performed by professionals enabling complex attacks.
- Potential **impact** may be severe:
  - Web applications are used for sensitive information and important transactions.
  - Attack may target site customers.



http://www.modsecurity.org/projects/modprofiler

What are we doing about it?

Web Application Security through the application lifecycle



Ensuring code is secure by training developers

- Programmers are not security experts. For example, they do not understand CSRF.
- Security is always a secondary goal.
- Code developed externally due to outsourcing, M&A and packaged software.

Inspecting applications for vulnerabilities: automated/ manual/ code review/ pen testing

- Very expensive to perform comprehensively: requires considerable expertise and time.
- Needs to be performed on each change in the application.

Real time protection using Web Applications Firewalls (WAFs)

- The cheapest solution.
- Last barrier for everything that sneaks through coding and testing.

Can WAFs be effective?

http://www.modsecurity.org/projects/modprofiler

BREACH SECURITY LABS

To Be Effective, WAFs need to:

- Provide protection against all attacks, both known and unknown.
- Be easy to use:
  - Work automatically, with little or no involvement from the user.
  - Allow for manual updates as needed.
- Have a low rate of false positives.
- Be production grade.

http://www.modsecurity.org/projects/modprofiler

WAF Protection Strategies

- Negative security model: allow all, deny what's wrong
  - Web specific IPS:
    - Simple concept, generic to all applications and provides instant security.
    - Based on rules instead of signatures: full parsing, complex logic, anti-evasion.
  - Difficult to guard against every attack variant and evasion attempts.
- Positive security model: deny all, allow what's right
  - An independent input validation envelope for web applications.
  - Provides the best protection.
  - Hard to implement:
    - Rules must be written specifically for each page in the application.
    - Rules needs to be maintained as the application changes.
  - Easy to write for specific vulnerabilities (virtual patching)
- Learning is needed to effectively use the positive model.

http://www.modsecurity.org/projects/modprofiler

Case study: The ‘1=1’ Signature

- Classic example of an SQL injection attack
  - Many IPS solutions include a signature to detect this attack.
  - The tautology ensures that the injected query returns ‘true’ .
- A WAF would easily overcome these evasions:
  - Encoding: 1%3D1,
  - Including white space characters: 1   =%091
  - Adding SQL inline comments: 1 /* comment */ = 1
- But it is impossible to create a signature for every tautology:
  - 1+1=2, 2 > 1 and for some databases just 1 or Ivan.
- A positive security rule will provide the best security:

```
<LocationMatch :"/login.php$">
            SecRule ARGS:username “!^\w+$″ ‶deny,log"
</LocationMatch>
```

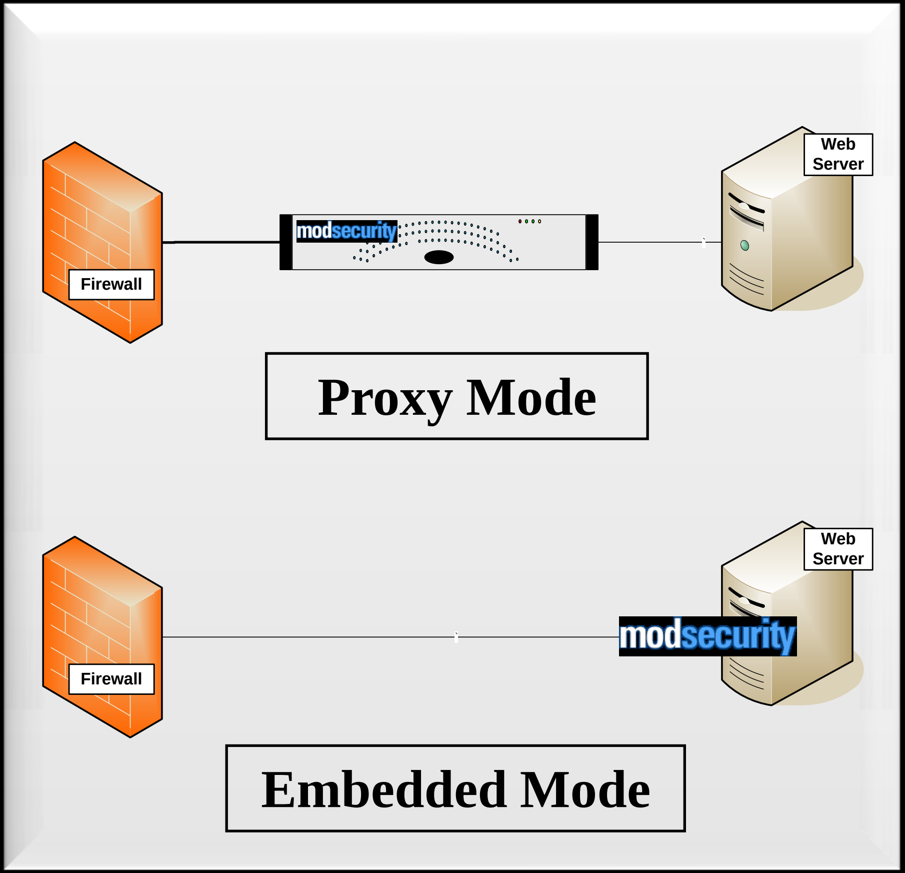
http://www.modsecurity.org/projects/modprofiler

BREACH SECURITY LABS

PART II - MODSECURITY

What is ModSecurity?

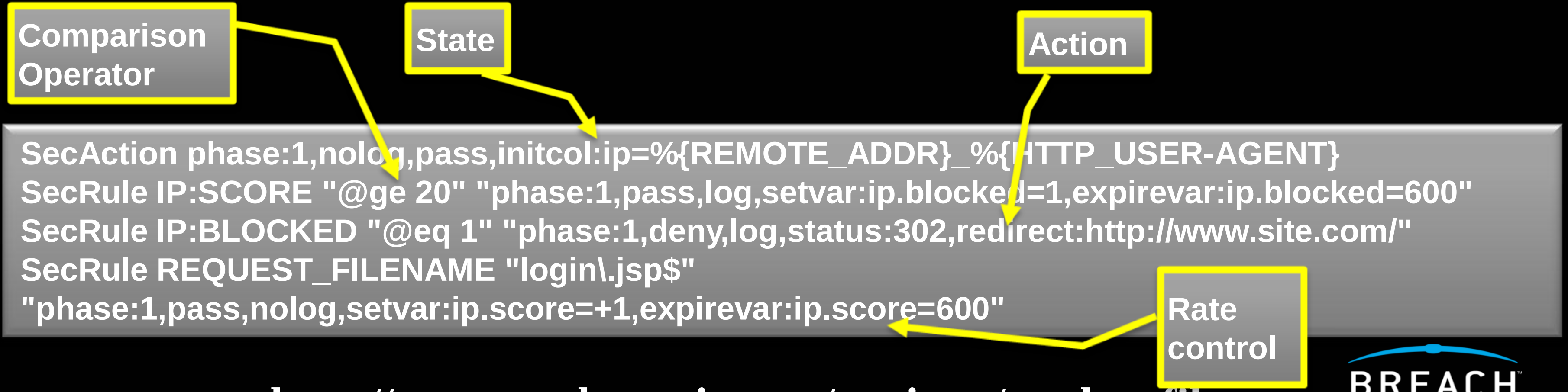
- The most popular WAF in the world with (a lot) more than 10,000 installations.
- An open source production grade project started in 2002.
- An Apache module which supports both embedded and reverse proxy deployments.
- Support and training by Breach Security.



http://www.modsecurity.org/projects/modprofiler

Technical overview

- Rules language is not a simple custom signatures engine, but rather an event-based scripting language targeted at inspecting HTTP transactions.
- Supports variables, state, control structure and even full blown scripting using LUA.
- Simple things are easy to do; complex things are possible, for example:
  - A signature for detecting a known attack vector.
  - A state based rule for detecting a brute force attack (see example below)



```
SecAction phase:1,nolog,pass,initcol:ip=%{REMOTE_ADDR}_%{HTTP_USER-AGENT}
SecRule IP:SCORE "@ge 20" "phase:1,pass,log,setvar:ip.blocked=1,expirevar:ip.blocked=600"
SecRule IP:BLOCKED "@eq 1" "phase:1,deny,log,status:302,redirect:http://www.site.com/"
SecRule REQUEST_FILENAME "login\.jsp$"
"phase:1,pass,nolog,setvar:ip.score=+1,expirevar:ip.score=600"
```

http://www.modsecurity.org/projects/modprofiler

Components

- ModSecurity 2.5:
  - The core rules processing engine.
- ModSecurity Core Rules:
  - An open source rule set providing a generic negative security application layer protection.
- ModSecurity Community Console:
  - A free tool for aggregating events from up to 3 ModSecurity sensors.

http://www.modsecurity.org/projects/modprofiler

Part III – Positive Security Using Learning

Alternative Learning Methods

- Outbound based dynamic policy
  - The original application firewalls technology.
  - WAF analyzes output pages to generate rules for input pages:
    - Input fields, hidden fields, links etc.
  - Defunct due to Web 2.0, AJAX & Web Services.
- Crawler based learning
  - Same process as dynamic policy, but built in advance.
  - Somewhat better than dynamic policy as crawler can interpret JavaScript.
  - Still a problem to adjust to changes and to achieve full coverage.
- Behavioral based learning:
  - Analyze inbound traffic to determine normal behavior.
  - The leading method today; Used by ModProfiler.

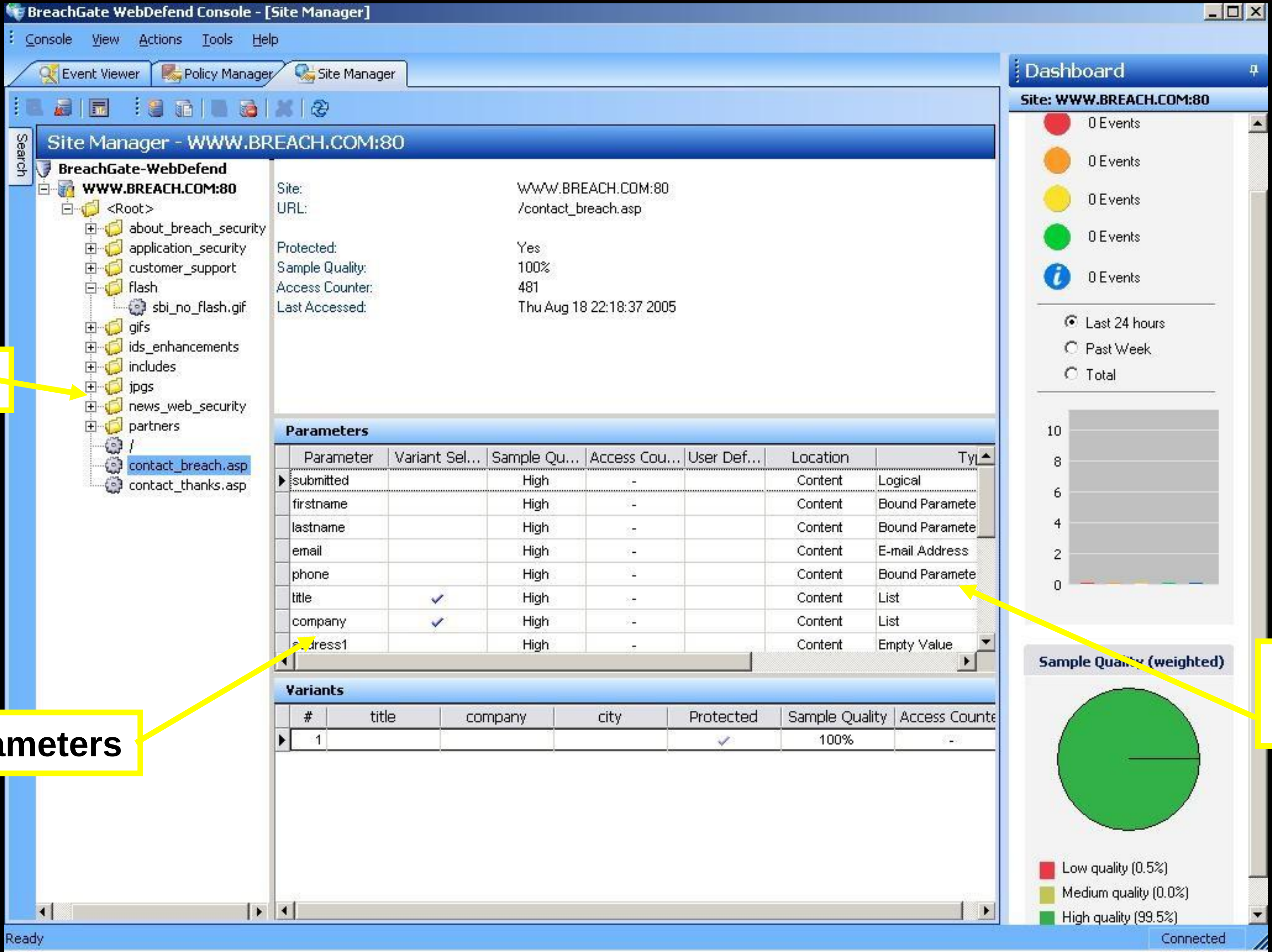
http://www.modsecurity.org/projects/modprofiler

Behavioral Based Learning

- Monitor inbound traffic and generate a normal behavior profile.
- Profile includes a statistical model for normal values of the properties of the request:
  - Field length, character set, expected value or type.
  - Existence, order, cardinality and location of fields.
  - Properties not limited to fields: can include for example also properties of headers or uploaded files.
- Validate request according to profile:
  - Each model separately.
  - Anomaly scoring: aggregating multiple tests.

http://www.modsecurity.org/projects/modprofiler

Sample Profile



Site Manager - WWW.BREACH.COM:80

- BreachGate-WebDefend
  - WWW.BREACH.COM:80
    - <Root>
      - about_breach_security
      - application_security
      - customer_support
      - flash
        - sbi_no_flash.gif
      - gifs
      - ids_enhancements
      - includes
      - jpgs
      - news_web_security
      - partners
      - /
      - contact_breach.asp
      - contact_thanks.asp

Site: WWW.BREACH.COM:80
URL: /contact_breach.asp

Protected: Yes
Sample Quality: 100%
Access Counter: 481
Last Accessed: Thu Aug 18 22:18:37 2005

Parameters

Parameter	Variant Sel...	Sample Qu...	Access Cou...	User Def...	Location	Ty
submitted		High	-		Content	Logical
firstname		High	-		Content	Bound Paramete
lastname		High	-		Content	Bound Paramete
email		High	-		Content	E-mail Address
phone		High	-		Content	Bound Paramete
title	✓	High	-		Content	List
company	✓	High	-		Content	List
address1		High	-		Content	Empty Value

Variants

#	title	company	city	Protected	Sample Quality	Access Counte
1				✓	100%	-

Dashboard

Site: WWW.BREACH.COM:80

- 0 Events
- 0 Events
- 0 Events
- 0 Events
- 0 Events

● Last 24 hours
○ Past Week
○ Total

Sample Quality (weighted)

- 🟥 Low quality (0.5%)
- 🟨 Medium quality (0.0%)
- 🟩 High quality (99.5%)

Site Map

Parameters

Parameter Types

http://www.modsecurity.org/projects/modprofiler

Behavioral Analysis Challenges

- Learning period:
  - Fixed length or determined by quality of sample?
  - Different for each element or global?
  - Protecting seldom used pages.
  - Avoiding learning attacks.
- Complex applications:
  - Identifying parameter: Custom separator, PATH_INFO, SOAP, JSON or non standard.
  - Dynamic URLs: Parameters as part of the URL.
  - A parameter specifying the action instead of the URL.
- Anomalies vs. attacks
  - O'Brien is Irish, O'Select is not.
- Change management.

http://www.modsecurity.org/projects/modprofiler

Part IV - ModProfiler

http://www.modsecurity.org/projects/modprofiler

Collecting Data

- Uses ModSecurity audit logs, which contain complete HTTP transaction data, as source of traffic.
- Filter out invalid traffic.
  - Ignore requests singled out by signatures.
  - Remove "noise" (e.g. non-200 transactions).
- Extract properties:
  - User defined mapping (Dynamic URLs, custom separators)

http://www.modsecurity.org/projects/modprofiler

Generation the Model

- Simple fixed size sample of requests used for elements and all models.
- Generates tests for each model (length, char set, type) for each parameters
  - This matches well ModSecurity rules capabilities.
- Exported as ModSecurity rules:
  - Blocking strategy set by user: Warn only, Block or Mixed mode: block for well-learned resources, warn for all others.
  - Recommended to use detection only mode initially to test rules and apply exceptions.

http://www.modsecurity.org/projects/modprofiler

Real Wold Issues

- Handling of partial learning:
  - Rules generated for URLs for which sample was too low can be set to alert even if other rules block.
  - Rules generated to alert/block on URLs and parameters not seen during learning.
- No handling of application changes: a change may result in a flood of events.
- Negative security should still be used:
  - Filter attacks for learning.
  - Provide protection during learning period and for partially and not learned resources.
  - Protection for free form text fields.

Part V - Conclusion

False Positives and False Negatives

- False positives (FPs):
  - How many times the rule set alerted when there was no attack?
  - As attack count is low, false positives are measured by counting total alerts.
- False negatives (FNs):
  - How many attacks did the rule set miss?
  - Nearly impossible to measure for a 0-day detection system. The best way to estimate is to measure level of protection against known exploits by running a scanner.
- FPs and FNs are a function of sample size, protected application and sample quality.

Future directions

- User profiling:
  - Learn the behavior of each user.
  - Can be used to detect fraud.
  - Requires handling a huge amount of information and compensating for a small sample per user.
- Session profiling:
  - Learn the normal flow of usage in the application.
- Handle additional data formats:
  - XML, JSON, URL Mapping.
- Real-time & continues operation:
  - Detect change by monitoring event flood or comparing profiles over time.
- Learning responses:
  - Detecting defacement, leakage and errors.

http://www.modsecurity.org/projects/modprofiler

Questions?

Ivan Ristic, ivanr@breach.com

Ofer Shezaf, ofers@breach.com

Further information:

http://www.modsecurity.org/projects/modprofiler