Positive Security Model for Web Applications, Challenges and Promise

Ofer Shezaf

OWASP IL Chapter leader

CTO, Breach Security

Introduction

Breach Security, Inc.

Breach Security is the market leader in web application security with global headquarters in Carlsbad, California.

With web application security expertise for over six years and led by experienced security executives, Breach Security is trusted by large enterprise customers.

Breach Security provides next-generation web application security solutions for protecting business-critical web applications transmitting privileged information, resolving security challenges such as identity theft, information leakage, regulatory compliance, and insecurely coded applications.

BREACH™

Introduction

Ofer Shezaf

Community Participation:

- ModSecurity Core Rule Set Project Leader
- OWASP Israeli chapter leader
- Web Application Security Consortium (WASC) Board Member
- WASC Web Hacking Incidents Database Project Leader

Day Job:

- CTO, Breach Security
- In charge of security research, rules and signatures.

Introduction

Ofer Shezaf

Community Participation:

- ModSecurity Core Rule Set Project Leader
- OWASP Israeli chapter leader
- Web Application Security Consortium (WASC) Board Member
- WASC Web Hacking Incidents Database Project Leader

Day Job:

- CTO, Breach Security
- In charge of security research, rules and signatures.

BREACH

Product Portfolio

Web Application Protection

WebDefend™
A next-generation web application firewall featuring unique out-of-line blocking capabilities and continuous application profile learning.

ModSecurity™
An entry level, low-cost web application firewall. Based on the open source version , it the most widely deployed web application firewall in the world with over 10,000 deployments.

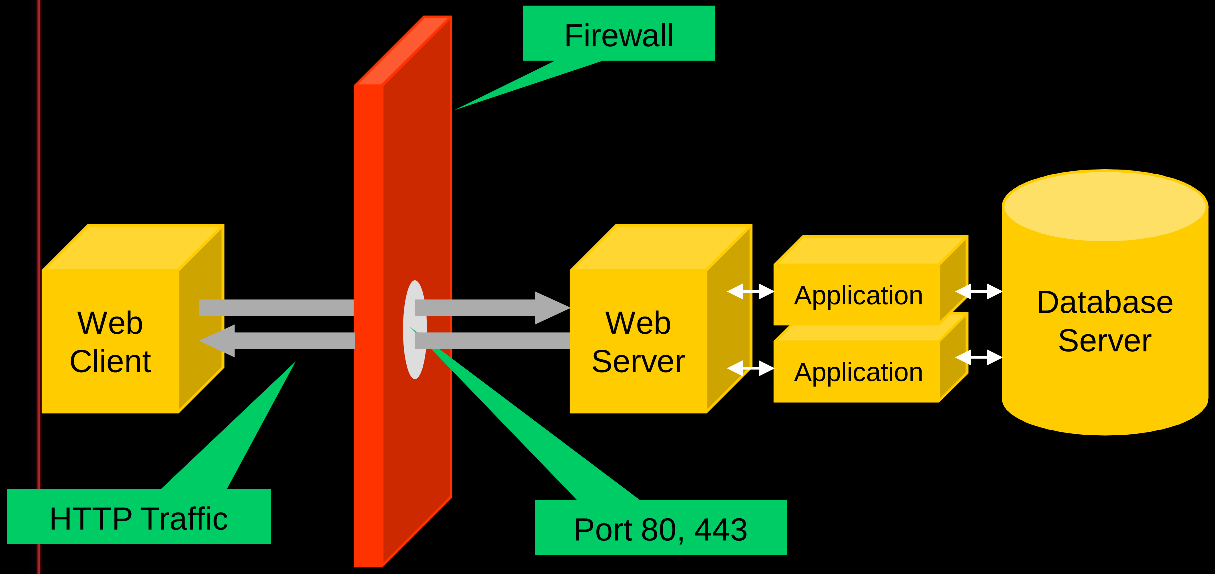
SSL Encrypted Traffic Viewer

BreachView SSL
A plug-in or security appliance which passively decrypts SSL traffic so that hidden threats can be detected by the IDS/IPS system.

BREACH

Web Application Security

Traditional Network Security



- **Firewalls block ALL inbound traffic to the web servers EXCEPT traffic over ports 80 & 443.**
- **The web application becomes the weak point, leaving responsibility for security in the hands of developers.**

BREACH

The Web Application Security Problem

Web applications:

- Are unique, each one exposing its own vulnerabilities.
- Changes frequently, requiring constant tuning of application security.
- Became complex and feature rich with the advent of AJAX, Web Services and Web 2.0, requiring developers to prioritize features and schedule before security.

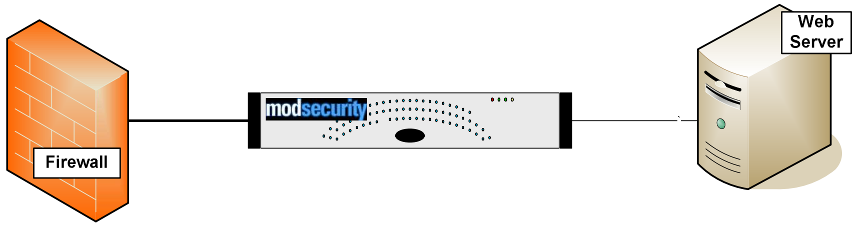
Consequently:

- Signature-based, deterministic, traditional “Network Security” can not protect custom & dynamic Web applications.
- Web applications reviews only provide visibility at the time it was performed.

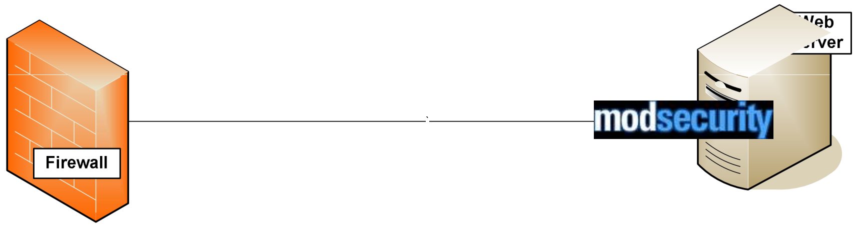
What Are Web Application Firewalls?

Multiple Deployment Modes

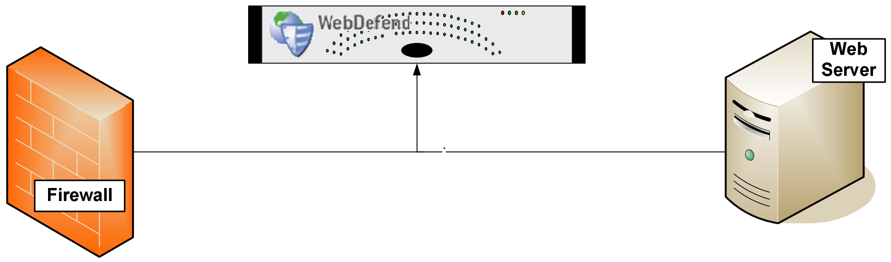
In-Line mode



Embedded mode



Out of line mode



Three Protection Strategies for WAFs

1. **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.
2. **Negative security model**
   - Looking for bad stuff,
   - Mostly signatures based.
   - Generic but requires some tweaking for each application.

IPS?

3. **Limited Positive Security: External patching**
   - Also known as "just-in-time patching" or "virtual patching".

Negative Security for Web Applications

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

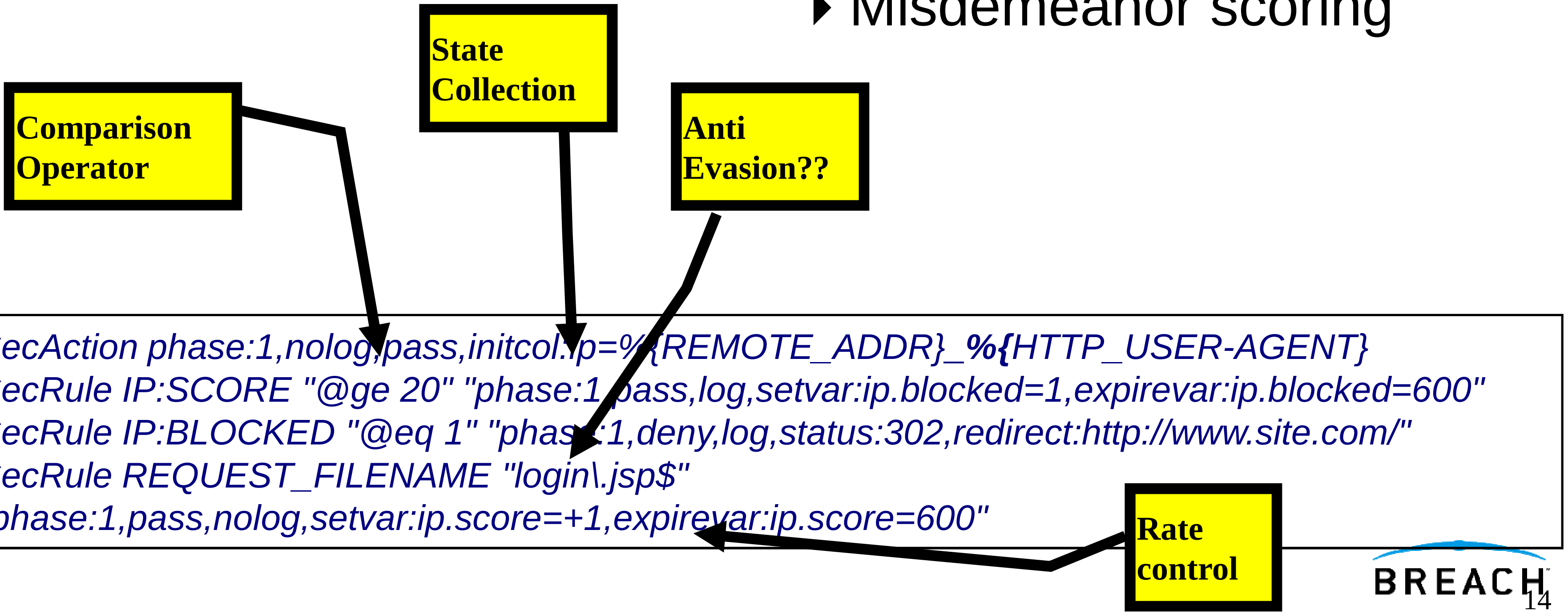
Some Complex Rules:

Monitoring:

- Capturing the user name
- Login failures

Protection

- Brute force detection
- Scanners and automation detection
- Misdemeanor scoring



```
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"
```

BREACH

Positive Security Model

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:

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

Positive Security Model

- The same but for every field in the application.
- Can also validate:
  - Links, Cookies, headers.
  - Output - sign each page to ensure correct page is sent

```
<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>
```

Learning

- Great security, but requires practically re-writing the application.
- Some auto policy generation is required:
  - Monitoring outbound traffic (dynamic policy)
  - Crawling
  - Monitoring inbound traffic (normal behavior):

Outbound Based Dynamic Policy

- The Original Application Firewalls Technology.
- How Does it Work:
  - WAF Analyzes output pages for: Input fields, hidden fields, combo boxes, links.
  - Build rules to validate field lengths, list of values, valid links
  - Validate incoming page according to rules
- Inherent problems:
  - No type validation information
  - Entry pages issue
  - JavaScript
- Does not fit modern, asymmetric, Web technologies:
  - Web 2.0 and AJAX
  - Web Services/SOAP

BREACH™

Crawler based learning

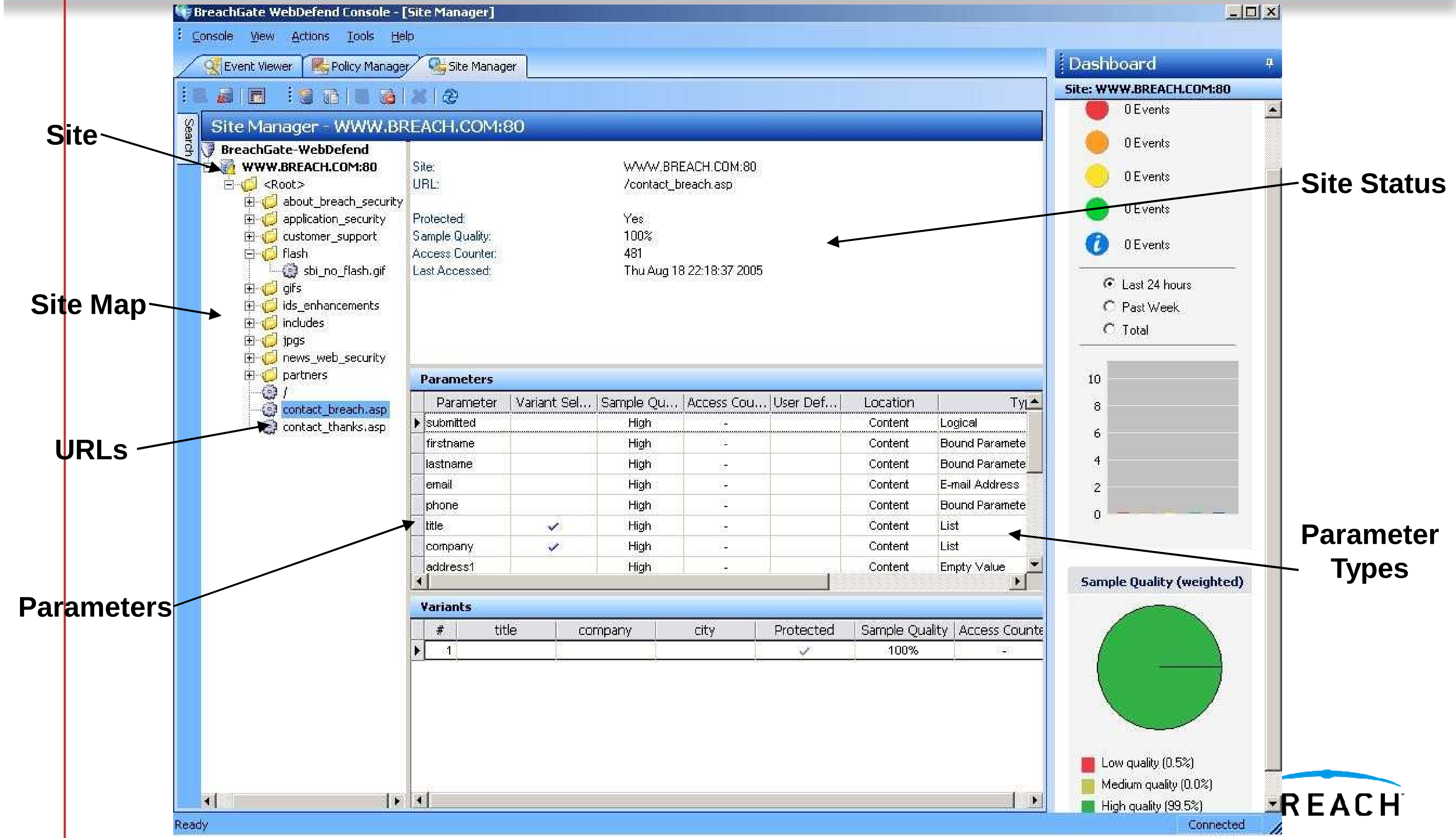
- How Does it Work:
  - Crawler crawls the site and builds the same rules as the dynamic policy, just before hand.
  - Client side can emulate JavaScript to overcome some of the limitations presented above.
- But:
  - A crawler cannot fully cover a site, especially a form based one
  - Type information and entry pages still an issue.
  - Changes are a problem.
- Scanner based learning
  - Using application security scanners to generate virtual patching rules.
  - A minimal approach utilizing the best in this approach.

BREACH™

Behavioral Based Learning

- How does it work:
  - Monitor inbound traffic and generate a normal behavior profile.
  - Profile includes different models over any aspect of the request or reply.
  - Validate requests (and replies) according to profile.
- Overcomes other learning methods shortcomings, but:
  - Learning period.
  - Filtering noise and attacks.
  - Change management.
  - Is what abnormal also an attack?
  - Seldom used pages.

A sample profile



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%	-

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

Behavioral Models

<u>For each attribute, the following models can apply:</u>

- Length
- Character set:
  - Probability of a character or group in field
  - Anonymous character distribution
- Token finder
  - Heuristic: learn specific lists.
  - Fixed maximum number of elements
  - By rate of new values
- Type:
  - Heuristic: learn specific types
  - Structural Inference
- Existence of attributes, order of attributes.

Based in part on work by Christopher Kruegel, Giovanni Vigna's et al.

Anomalies vs. Attacks

- Signatures and other detection engines help prevent malicious traffic from being learned.
- Model should eliminate highly abnormal values and predict unseen values:
  - Easier for length, harder for token finder
- Use of anomaly scoring for intermediate results.

Anomaly scoring

- Each test produce continuous result and not a binary one.
- Result of tests are compound to detect attacks:
  - In the same request/reply
  - Over time
- None behavioral tests are also quantified:
  - Parser and RFC compliance issues
  - Signature matching

O'Brien is Irish, O'Select is not

BREACH

Advanced Behavioral Learning

- Learning if outbound based dynamic policy is valid:
  - Are hidden fields changes by JavaScript?
  - Is a combo box limited to the values in the HTML form?
- Continuous learning:
  - Detect change by monitoring level of alerts
  - Continuously modify the profile using time windows.
  - Detect change by comparing learning results for different time windows.
- Per user learning:
  - Enables fraud detection.

BREACH

Thank You!

Ofer Shezaf

ofers@breach.com