BREACH

Evaluation Criteria for Web Application Firewalls

Ivan Ristić

VP Security Research

Breach Security

Introduction

Breach Security

- Global headquarters in Carlsbad, California
- Web application security provider for over six years
- Led by experienced security executives
- Trusted by large enterprise customers

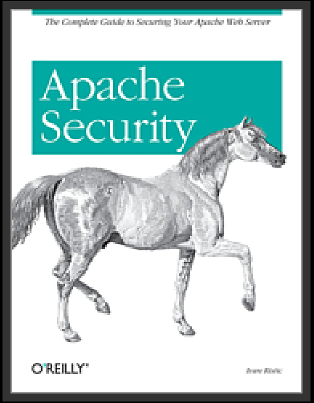
- Next-generation web application security solutions for protecting business-critical applications transmitting privileged information.
- Resolve security challenges such as identity theft, information leakage, regulatory compliance, and insecurely coded applications.
- Best threat detection in the industry and the most flexible deployment options available.

BREACH™

Introduction

Ivan Ristić

- **Web application security** and **web application firewall** specialist
- Author of **Apache Security**
- Author of **ModSecurity**
- **OWASP London Chapter** leader
- Officer of the **Web Application Security Consortium**
  - WAFEC project leader



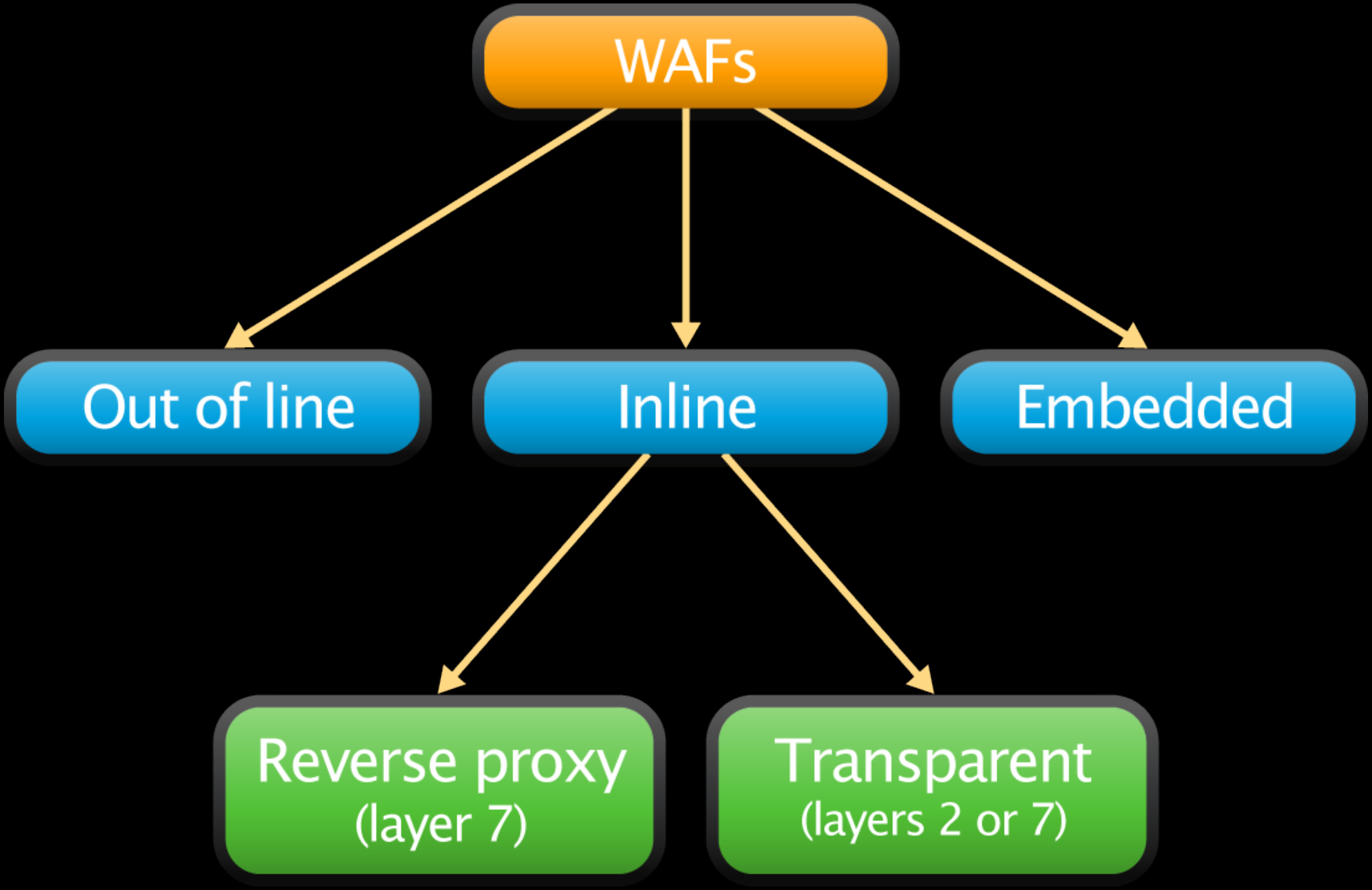
modsecurity

BREACH

BREACH™

1. Seeing

A variety of deployment options



BREACH

What you really care about

Key questions:

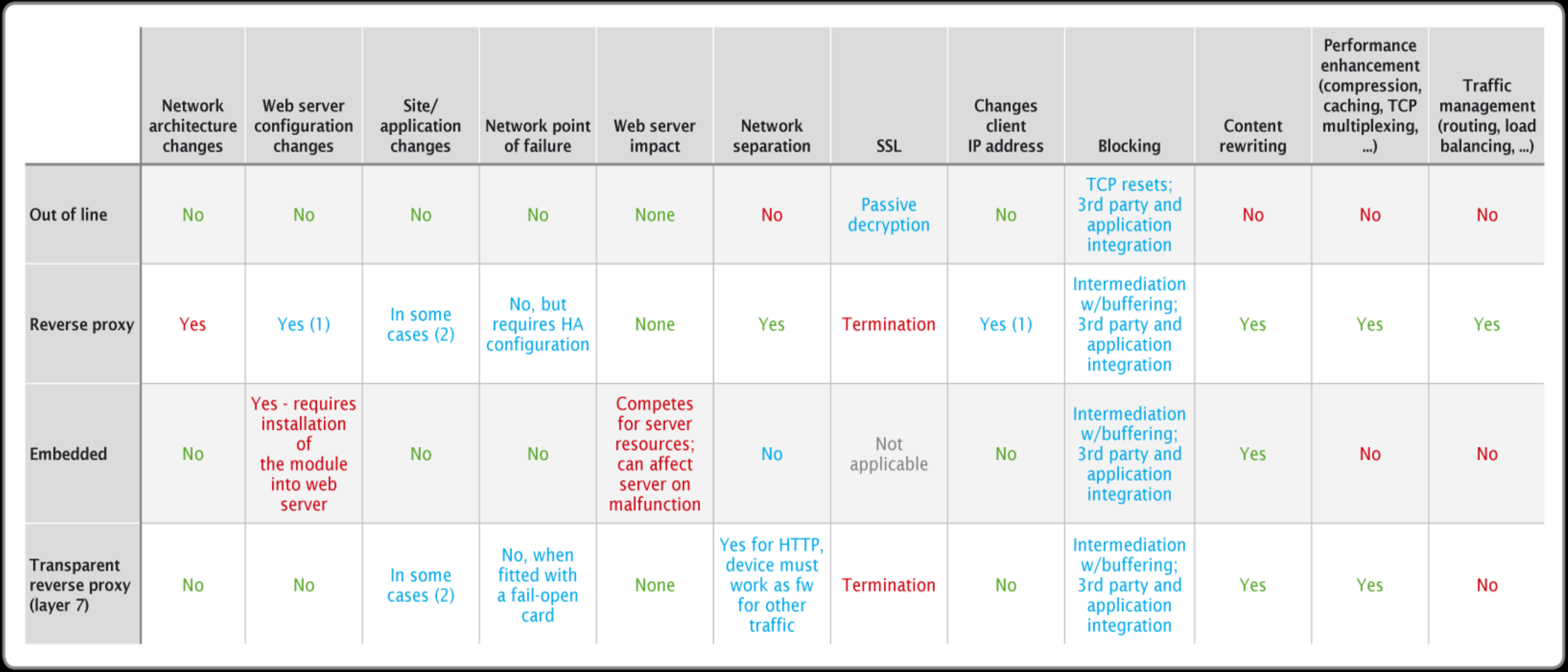
1. How easy is it to install?
2. What happens when it breaks?
3. How will it impact my stuff?
   - Performance
   - Traffic modification
   - SSL handling
4. How does it block?

Do not forget to look within

Internal influencing factors are often stronger than the external ones:

- Legacy
- Internal and external deadlines
- Resources (installation and maintenance)
- Architecture
- Growth (scaling)
- Will
- Organisational boundaries

Capability matrix



	Network architecture changes	Web server configuration changes	Site/ application changes	Network point of failure	Web server impact	Network separation	SSL	Changes client IP address	Blocking	Content rewriting	Performance enhancement (compression, caching, TCP multiplexing, ...)	Traffic management (routing, load balancing, ...)
Out of line	No	No	No	No	None	No	Passive decryption	No	TCP resets; 3rd party and application integration	No	No	No
Reverse proxy	Yes	Yes (1)	In some cases (2)	No, but requires HA configuration	None	Yes	Termination	Yes (1)	Intermediation w/buffering; 3rd party and application integration	Yes	Yes	Yes
Embedded	No	Yes - requires installation of the module into web server	No	No	Competes for server resources; can affect server on malfunction	No	Not applicable	No	Intermediation w/buffering; 3rd party and application integration	Yes	No	No
Transparent reverse proxy (layer 7)	No	No	In some cases (2)	No, when fitted with a fail-open card	None	Yes for HTTP, device must work as fw for other traffic	Termination	No	Intermediation w/buffering; 3rd party and application integration	Yes	Yes	No

Impact

Internal influencing factors:

1. Performance
2. Traffic modification
3. SSL handling, which can impact applications that are using private certificates

Impact: Performance

Performance:

1. Out-of-line – no impact
2. Inline devices can go either way:
   - Added latency
   - Performance decrease under load
   - Reverse proxy performance improvements (compression, caching, etc...)
3. Embedded solutions compete for web server resources

Impact: Traffic

Traffic modification:

1. Change of IP address by reverse proxies
   - Can be mitigated with a web server module
   - Or by using a transparent reverse proxy
2. Changes to dynamics due to use of buffering for blocking
   1. Problems with very large requests
   2. Problems with applications that expect instant data delivery

Impact: SSL

SSL:

1. Products that passively decrypt SSL will not cause any impact
2. SSL termination might:
    1. If you are using private SSL certificates you will have to reconfigure the web server or the application

BREACH™

Blocking

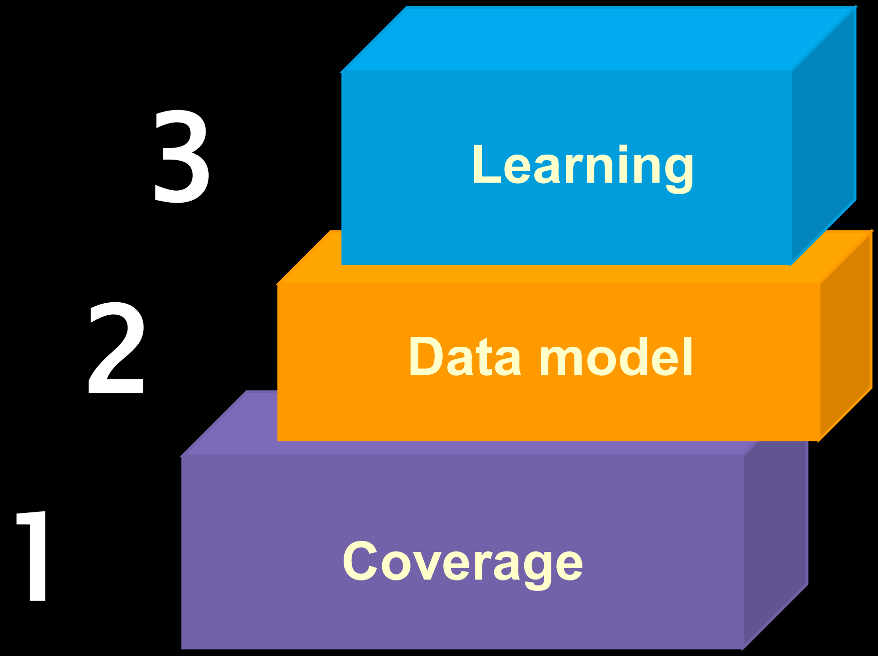
Blocking options:

1. TCP resets (out of line)
2. Packet blocking (inline layer 2)
3. Intermediation w/buffering (inline layer 7 and embedded)
   - Can be mitigated with a web server module
   - Or by using a transparent one

- Orchestration of external blocking architecture (e.g. firewalls, apps)

2. Understanding

Building blocks of understanding



BREACH™

Coverage

1

Ability to peel through layers of data:

1. Access to the entire data stream (in & out)
2. Complete HTTP parsing, including:
   - Chunked encoding
   - Compression
3. Various request body formats
   - application/x-www-form-urlencoded
   - multipart/form-data
   - XML

- Complete character encoding support
- Ability to handle non-standard traffic
- Strong counter-evasion facilities

BREACH

Data model

Data model requires **stateful operation**, and consists of the following elements:

1. Location
   - GeoIP lookups
   - IP address blocks
2. Application
3. Session
4. User
   1. Sign in
   2. Sign out

Learning

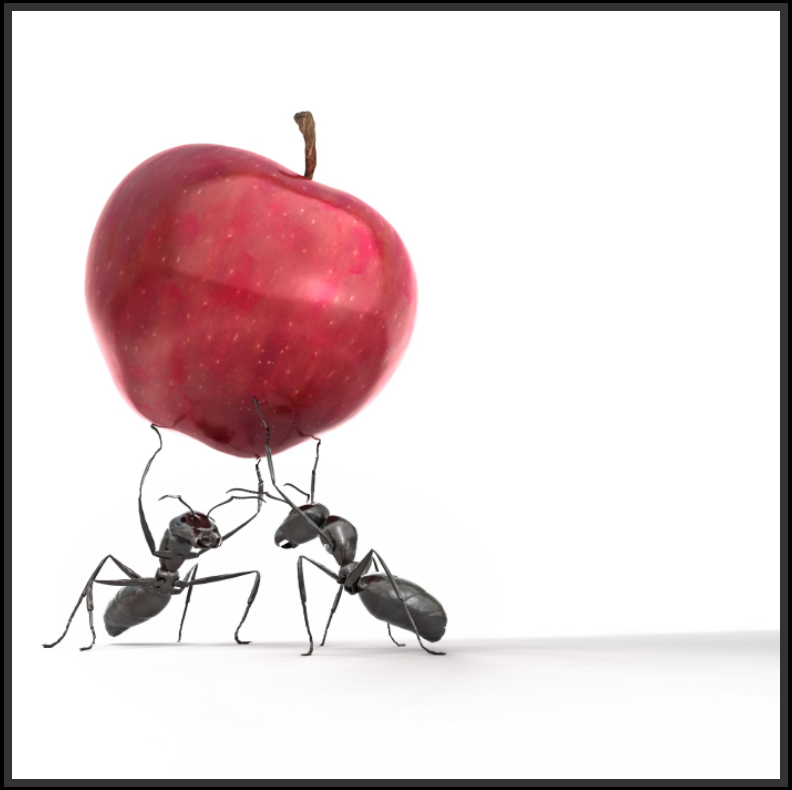
3

What **you** really want is for someone else to do the hard work:

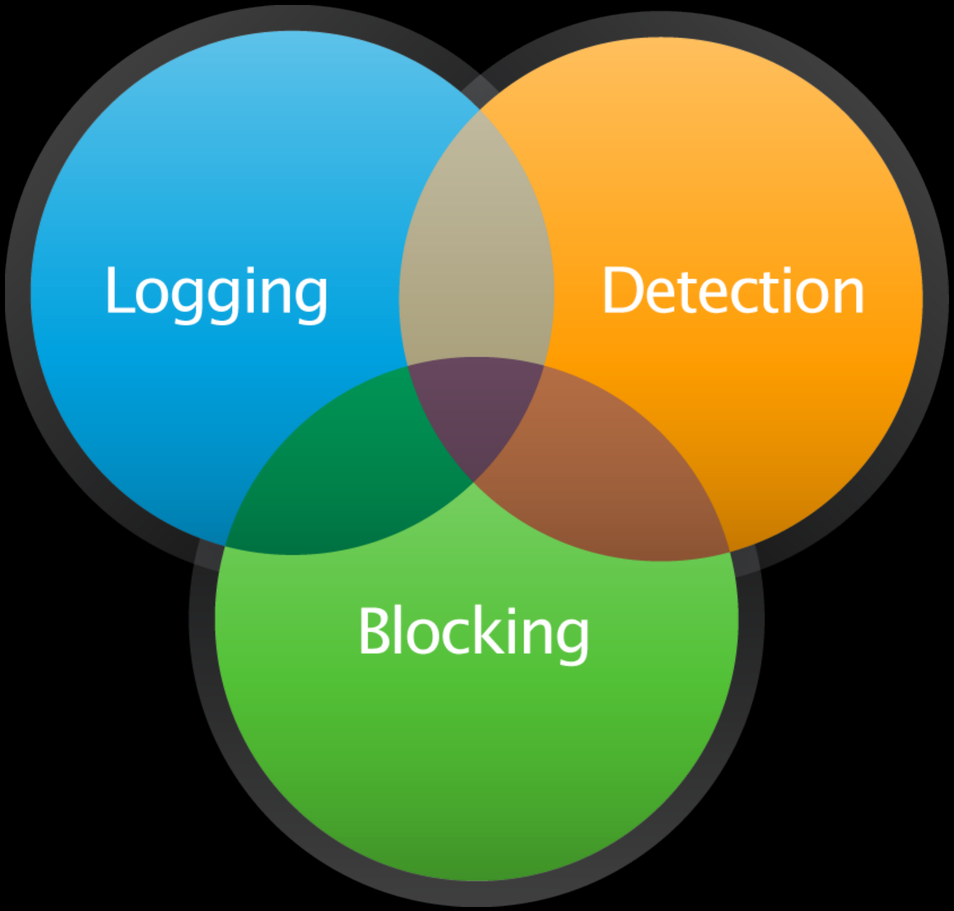
1. WAFs can create a (positive-security) model of your application by monitoring traffic
   - Not all products support learning
   - A few claim they do but don't work well
2. Try it out before you commit
   - It's the single best time saver

- And make sure it's continuous

3. Doing



Doing: Overview


Logging
Detection
Blocking

Logging

Full transaction logging is key:

- Very resource intensive:
  - Classify information
  - Remove sensitive data, then index and store data
  - Keep transaction around, and report on it
  - Delete transaction

2. Logging choices:
   1. Logging everything is rarely an option

      Ability to choose **exactly** what to log is very handy!
   - Log only relevant transactions
   - Log all transactions except static ones
   - Smart logging?

BREACH

Detection

1. Detection ability comes from several places:
    - Signature database
    - Custom-rule writing
    - Learning
    - Hard-coded functionality
2. The two most important aspects:
    1. False positives
    2. False negatives

BREACH™

Signature database

Negative security model is the foundation:

1. Scope
   - Generic web application attacks
   - Platform issues (e.g. Apache, IIS)
   - Product issues (e.g. WordPress)
2. Update frequency
3. Quality
   1. Exploit or vulnerability based
   2. Low rate of false positives
   3. Low rate of false negatives

BREACH

Custom rule writing

Just-in-time patching is an important WAF use case:

1. Very narrow scope: just one problem
2. Approaches:
   - Negative model
   - Positive model (preferred)
3. May require complex logic
4. May require custom normalisation
5. Ideally, a programming language

Learning

Contributing factors:

- Speed
- Accuracy
- Granularity
  - Partial learning/re-learning support
- Adaptation to changes
  - Manual
  - Change detection
  - Continuous learning
- Support for manual tweaking
- Adaptability

Packaged functionality

Some things just need to be hard-coded. For example:

1. Brute force attack detection
2. Cookie signing & encryption
3. PDF Universal XSS defence

Blocking

1. Block requests
2. Block responses
3. Persistent blocking
   - IP address/block
   - Session
   - User
4. Custom error page
   1. With unique transaction ID embedded

Decision time

Parting thoughts

Things we didn't mention are equally important:

1. Configuration granularity
2. Robust policy management
3. Usability
4. Management features
5. Reporting
6. Enterprise features

Resources

For further information:

- WAFEC – http://www.wafec.com
  - WAFEC v1 published in 2006
  - WAFEC v2 will be published soon
  - Join the group!
- ICSA Labs – http://www.icsalabs.com
  - WAF Certification Criteria
- PCI Security Council – https://www.pcisecuritystandards.org
  1. Requirement 6.6 Supplement

BREACH

Questions?

Thank you!

Ivan Ristić

ivan.ristic@breach.com