Web Application Firewalls: When Are They Useful?

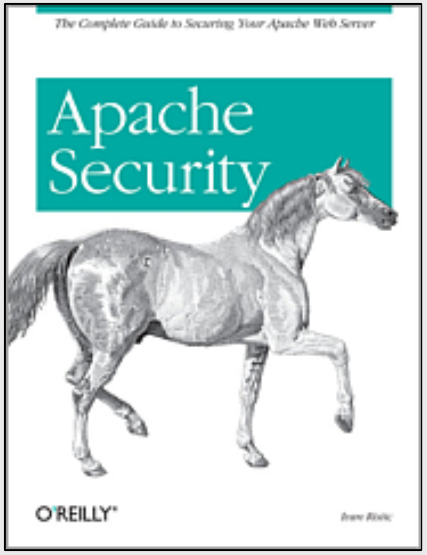
Ivan Ristic
Thinking Stone
ivanr@webkreator.com
+44 7766 508 210

OWASP AppSec Europe
May 2006

The OWASP Foundation
http://www.owasp.org/

Ivan Ristic

- Web Application Security specialist; Developer.
- Author of **Apache Security**.
- Author of **ModSecurity**.
- Founder of **Thinking Stone**.



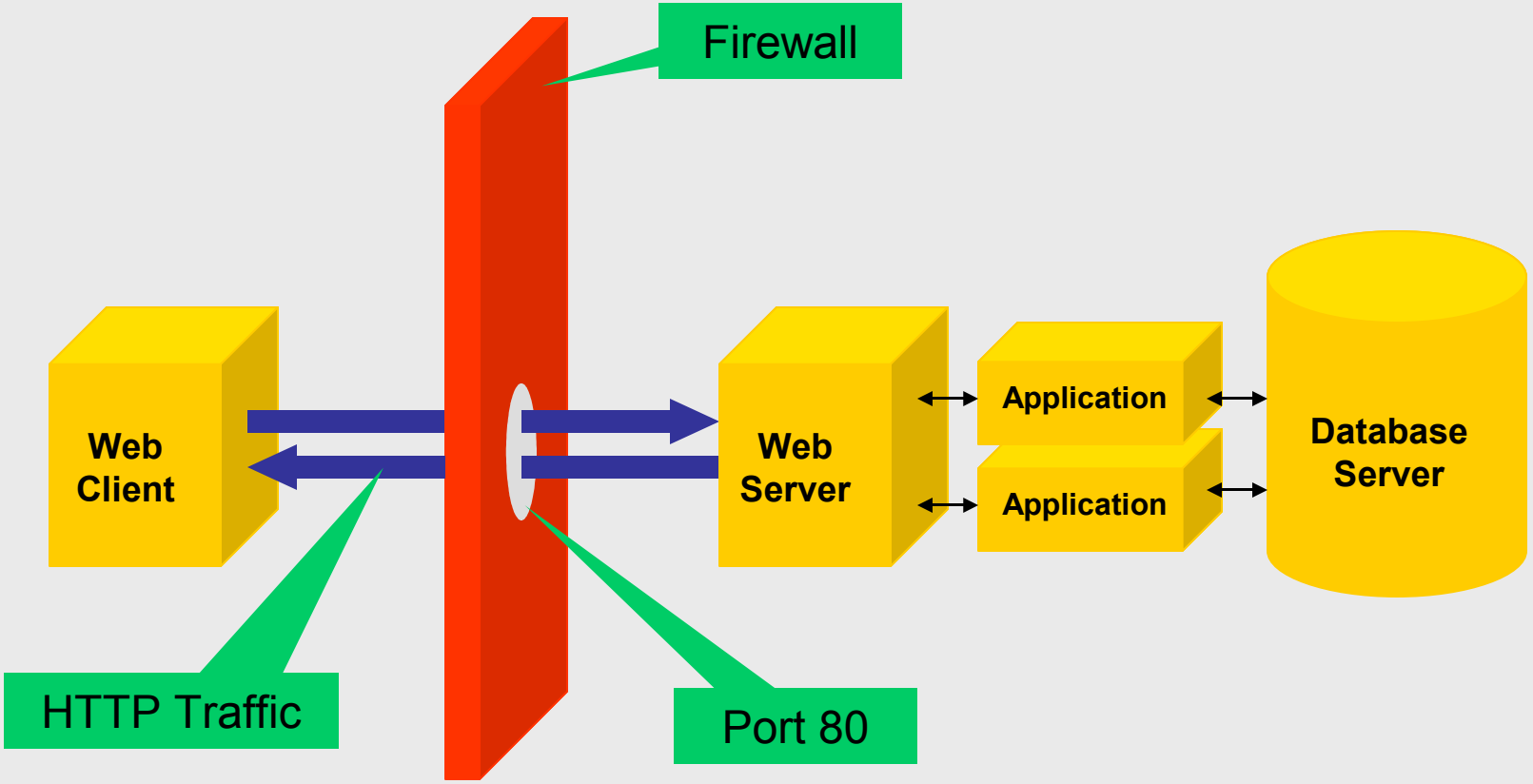
modsecurity

ThinkingStone
Securing Web Applications

Why Use Web Application Firewalls?

- In the nutshell:
  1. Web applications are deployed terribly insecure.
  2. Developers should, of course, continue to strive to build better/more secure software.
  3. But in the meantime, sysadmins must do something about it. (Or, as I like to say: **We need very help we can get**.)
  4. **Insecure applications aside, WAFs are an important building block in every HTTP network.**

Network Firewalls Do Not Work For HTTP



WAFEC (1)

- **Web Application Firewall Evaluation Criteria**.
- Project of the Web Application Security Consortium (webappsec.org).
- It's an open project.
- Nine WAF vendors on board, but **I'd like to see more users on the list.**
- WAFEC v1.0 published in January.
- We are about to start work on v1.1.

WAFEC (2)

- Nine sections:
  1. **Deployment Architecture**
  2. **HTTP and HTML Support**
  3. **Detection Techniques**
  4. **Prevention Techniques**
  5. **Logging**
  6. Reporting
  7. Management
  8. Performance
  9. XML

WAFEC (3)

WAFEC is not for the vendors.

It's for the users.

(So please voice your opinions!)

http://www.webappsec.org/projects/wafec/

WAF Identity Problem (1)

- There is a long-standing WAF identity problem.
- With the **name**, first of all*:

Adaptive Firewall
Adaptive Proxy
Adaptive Gateway
Application Firewall
Application-level Firewall
Application-layer Firewall
Application-level Security Gateway
Application Level Gateway
Application Security Device
Application Security Gateway
Stateful Multilayer Inspection Firewall

Web Adaptive Firewall
Web Application Firewall
Web Application Security Device
Web Application Proxy
Web Application Shield
Web Shield
Web Security Firewall
Web Security Gateway
Web Security Proxy
Web Intrusion Detection System
Web Intrusion Prevention System

List compiled by Achim Hoffmann.

WAF Identity Problem (2)

- There are four aspects to consider:
  1. **Audit device**
  2. **Access control device**
  3. **Layer 7 router/switch**
  4. **Web Application Hardening tool**
- These are all valid requirements but the name **Web Application Firewall** is not suitable.
- On the lower network layers we have a different name for each function.

WAF Identity Problem (3)

- Appliance-oriented web application firewalls **clash** with the **Application Assurance market**.
- Problems solved long time ago:
  - Load balancing
  - Clustering
  - SSL termination and acceleration
  - Caching and transparent compression
  - URL rewriting
  - ...and so on

WAF Identity Problem (4)

- Key factors:
  1. Application Assurance vendors are very strong.
  2. Web Application Firewall vendors not as much.
- Result:
  - **Appliance-oriented WAFs are being assimilated by the Application Assurance market.**
- In the meantime:
  - **Embedded WAFs are left alone because they are not an all-or-nothing proposition.**

WAF Functionality Overview

The Essentials (1)

- **Full support for HTTP**:
  - Access to individual fields (field content, length, field count, etc).
  - Entire transaction (both request and response).
  - Uploaded files.
- **Anti-evasion features** (also known as normalisation/canonicalisation/transformation features).

The Essentials (2)

- **Blocking features**:
  - Transaction
  - Connection
  - IP Address
  - Session
  - User
  - Honeypot redirection
  - TCP/IP resets (connection)
  - Blocking via external device
- **What happens upon detection?**

Fancy Features

- **Stateful operation:**
  - IP Address data
  - Session data
  - User data
- **Event Correlation**
- **High availability:**
  - Failover
  - Load-balancing
  - Clustering
  - State replication

Hard-Coded Protection Techniques (1)

- **Cookie protection**
  - Sign/encrypt/virtualise
- **Hidden field protection**
  - Sign/encrypt/virtualise
- **Session management protection**
  - Enforce session duration timeout, inactivity timeout.
  - Prevent fixation.
  - Virtualise session management.
  - Prevent hijacking or at least warn about it.

Hard-Coded Protection Techniques (2)

- **Brute-force protection**
- **Link validation**
  - Signing
  - Virtualisation
- **Request flow enforcement**
  - Statically
  - Dynamically

Other Things To Consider (1)

- **Management**:
  - Is it possible to manage multiple sensors from one place?
  - Support for administrative accounts with different privileges (both horisontal and vertical).
- **Reporting** (giving Management what it wants):
  - On-demand and scheduled reports with support for cus
- **XML**:
  - WAFs are expected to provide basic support for XML parsing and validation.
  - Full XML support is usually available as an option, or as a completely separate product.

Other Things To Consider (2)

- **Extensibility**:
  - Is it possible to add custom functionality to the firewall?
  - Is the source code available? (But not as a replacement for a proper API.)
- **Performance**:
  - New connections per second.
  - Maximum concurrent connections.
  - Transactions per second.
  - Throughput.
  - Latency.

Signatures and Rules

Signatures or Rules?

1. **Signatures**
   - Simple text strings or regular expression patterns matched against input data.
   - Not very flexible.
2. **Rules**
   1. Flexible.
   2. Multiple operators.
   3. Rule groups.
   4. Anti-evasion functions.
   5. Logical expressions.
   6. Custom variables.

Three Protection Strategies

1. **External patching**
   - Also known as "just-in-time patching" or "virtual patching").
2. **Negative security model**
   - Looking for bad stuff.
   - Typically used for Web Intrusion Detection.
   - Easy to start with but difficult to get right.
3. **Positive security model**
   - Verifying input is correct.
   - Usually automated, but very difficult to get right with applications that change.
   - It's very good but you need to set your expectations accordingly.

Auditing and HTTP Traffic Monitoring

Web Intrusion Detection

- Often forgotten because of marketing pressures:
  - **Detection** is so last year (decade).
  - **Prevention** sounds and sells much better!
- The problem with prevention is that it is **bound to fail** given sufficiently determined attacker (or inexperienced WAF operator).
- **Monitoring** (logging and detection) is actually more important as it allows you to independently audit traffic, and go back in time.

Monitoring Requirements

- Centralisation.
- Transaction data storage.
- Control over **which transactions are logged** and **which parts of each transaction** are logged, **dynamically** on the **per-transaction** basis.
  - Minimal information (session data).
  - Partial transaction data.
  - Full transaction data.
- Support for data sanitisation.
- Can implement your retention policy.

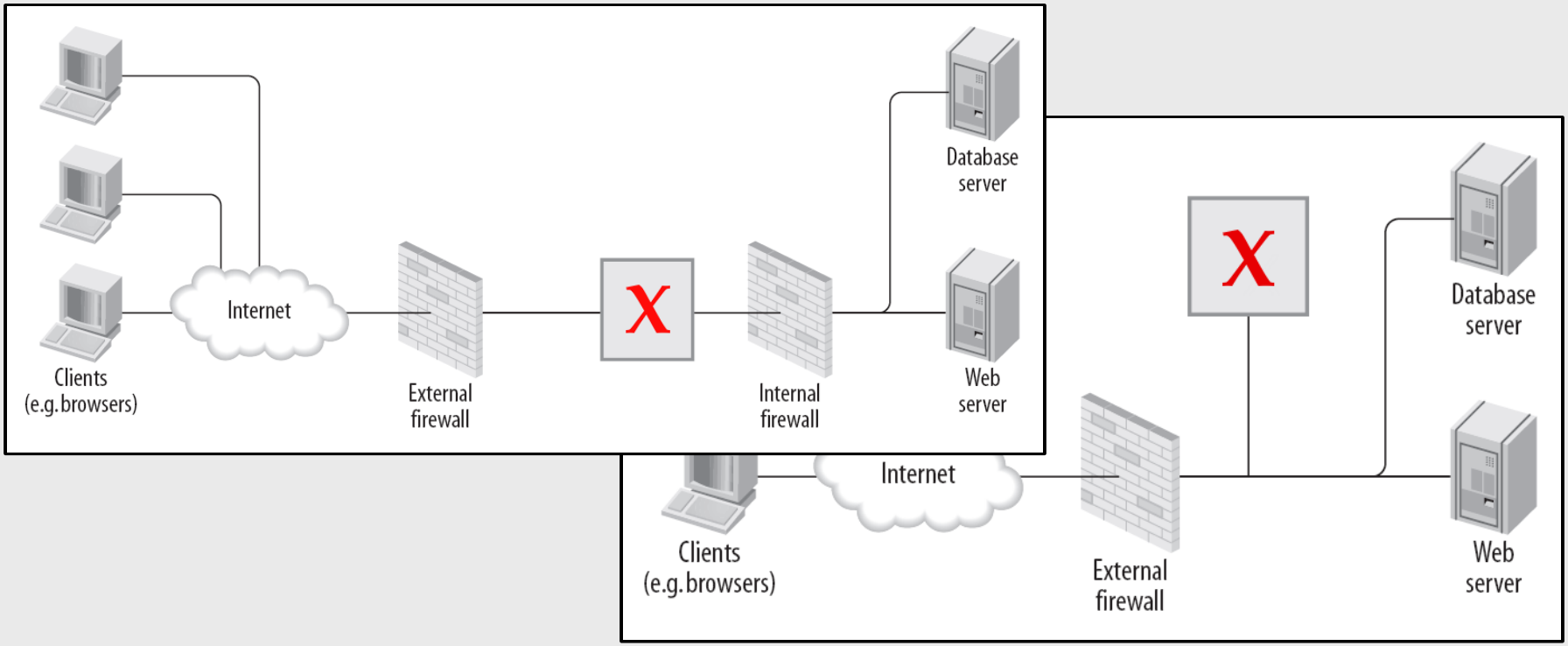
Deployment

Deployment

- Three choices when it comes to deployment:
  1. **Network-level device.**
  2. **Reverse proxy.**
  3. **Embedded in web server.**

Deployment (2)

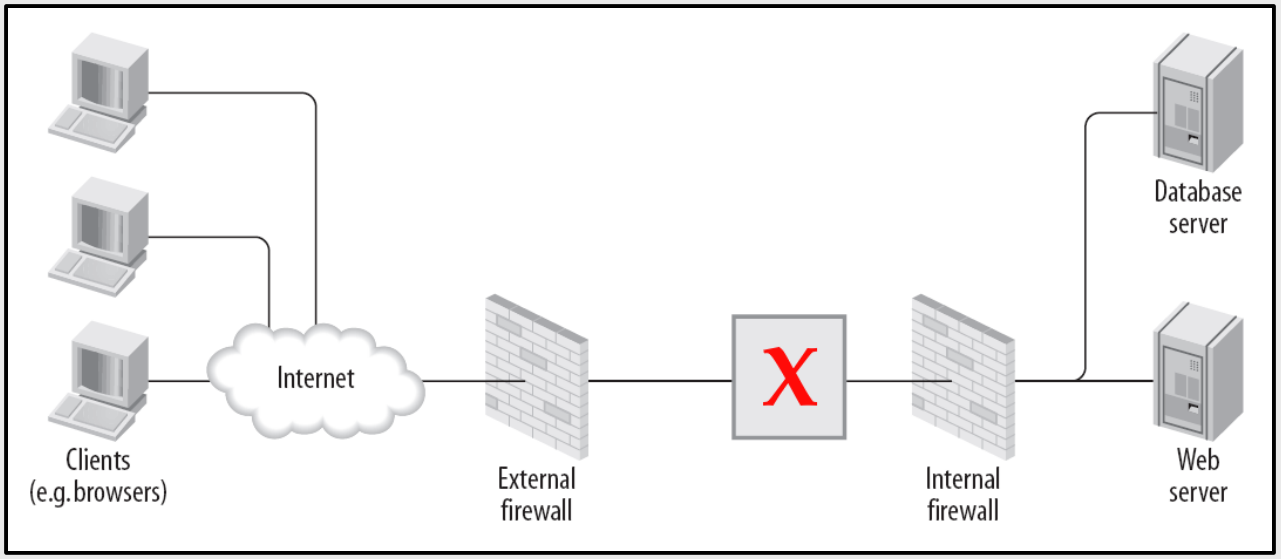
1. Network-level device



Does not require network re-configuration.

Deployment (3)

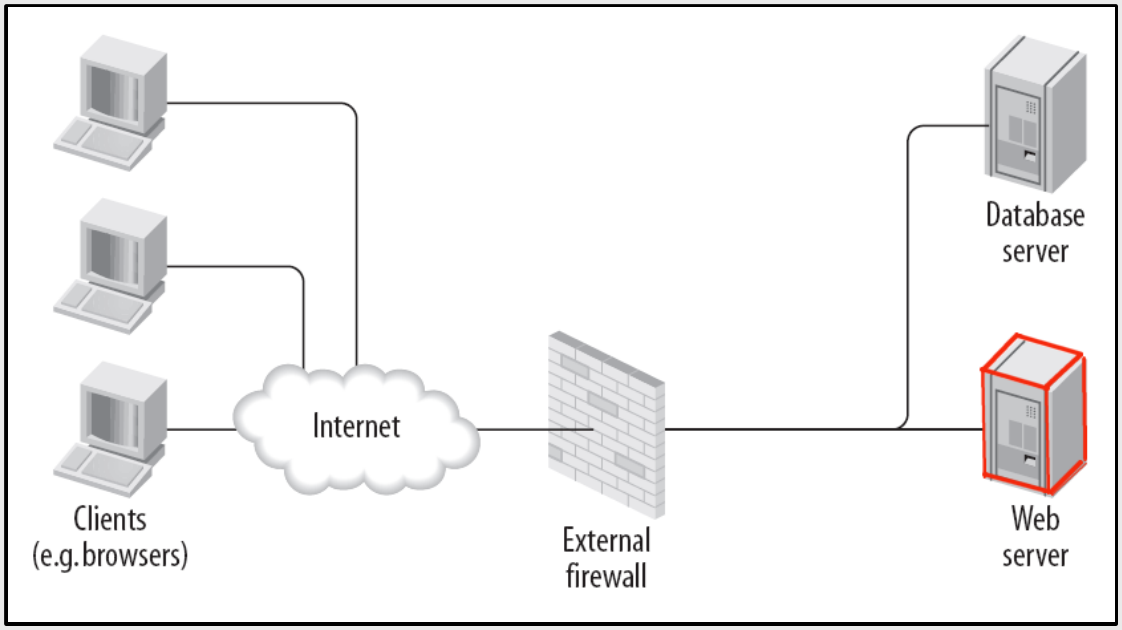
2. Reverse proxy



Typically requires network re-configuration.

Deployment (4)

3. Embedded



Does not require network re-configuration.

Deployment (5)

1. Network passive
   - Does not affect performance.
   - Easy to add.
   - Not a bottleneck or a point of failure.
   - Limited prevention options.
   - Must have copies of SSL keys.
2. Network in-line
   - A potential bottleneck.
   - Point of failure.
   - Must have copies of SSL keys.
   - Easy to add.

Deployment (6)

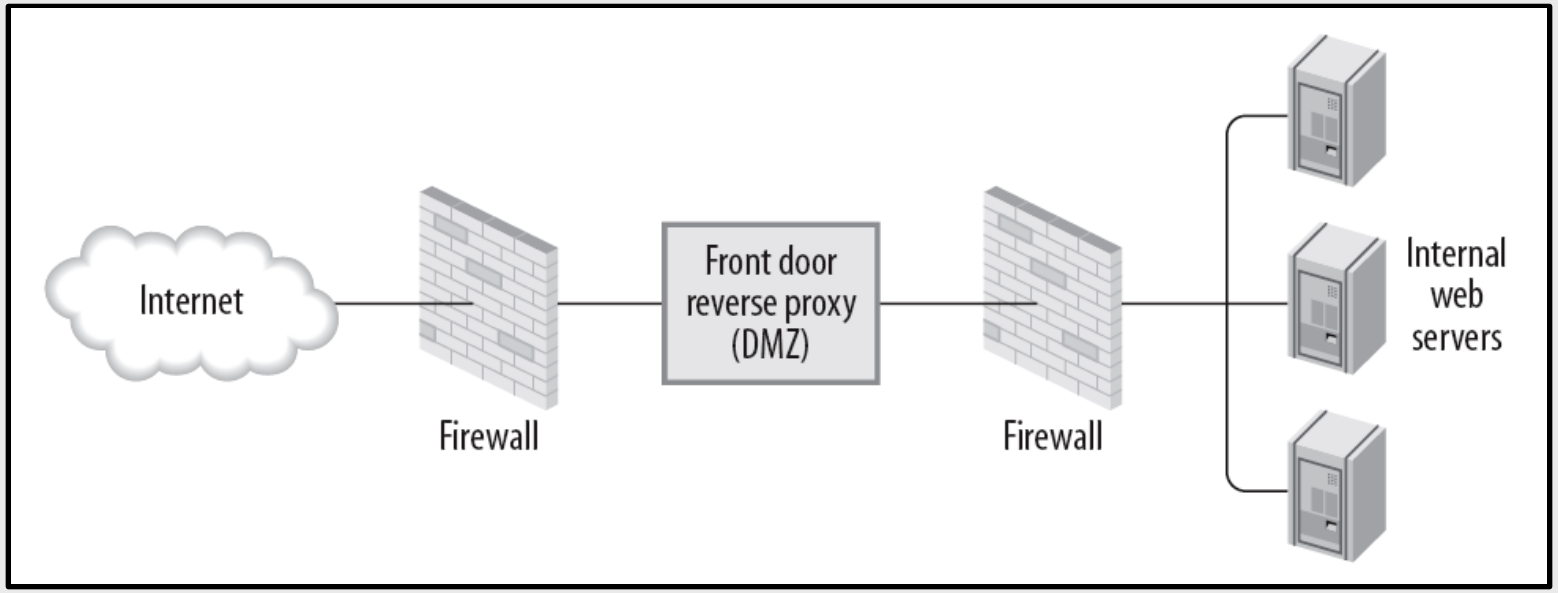
3. Reverse proxy
   - A potential bottleneck.
   - Point of failure.
   - Requires changes to network (unless it's a transparent reverse proxy).
   - Must terminate SSL (can be a problem if application needs to access client certificate data).
   - **It's a separate architecture/security layer.**
4. Embedded
   - Easy to add (and usually much cheaper).
   - Not a point of failure.
   - Uses web server resources.

Reverse Proxy As a Building Block

- Reverse proxy patterns:
  1. Front door
  2. Integration reverse proxy
  3. Protection reverse proxy
  4. Performance reverse proxy
  5. Scalability reverse proxy
- Logical patterns, orthogonal to each other.
- Often deployed as a single physical reverse proxy.

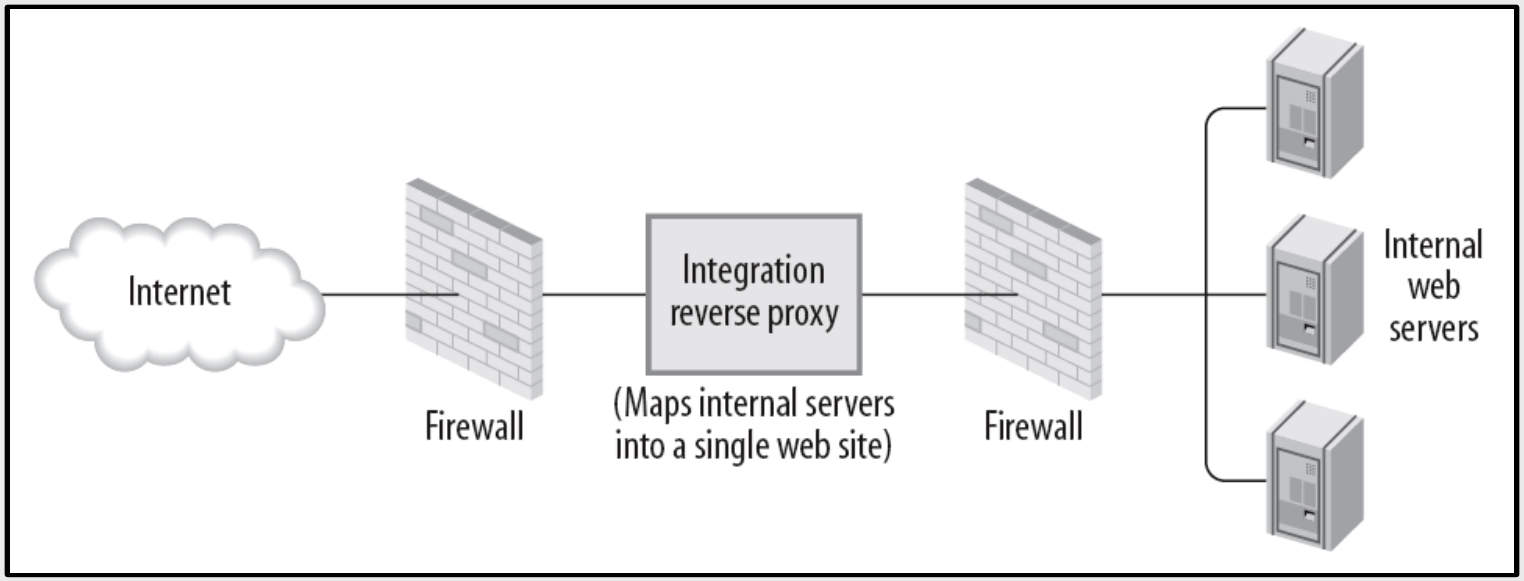
Front Door (1/5)

- Make all HTTP traffic go through the proxy
- Centralisation makes access control, logging, and monitoring easier



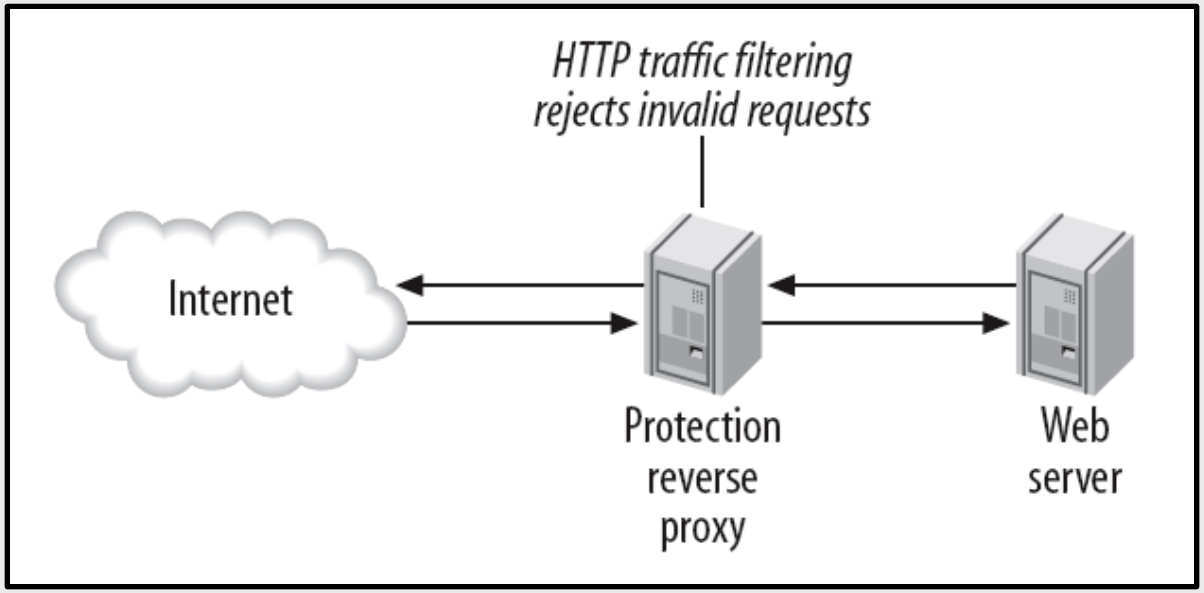
Integration Reverse Proxy (2/5)

- Combine multiple web servers into one
- Hide the internals
- Decouple interface from implementation



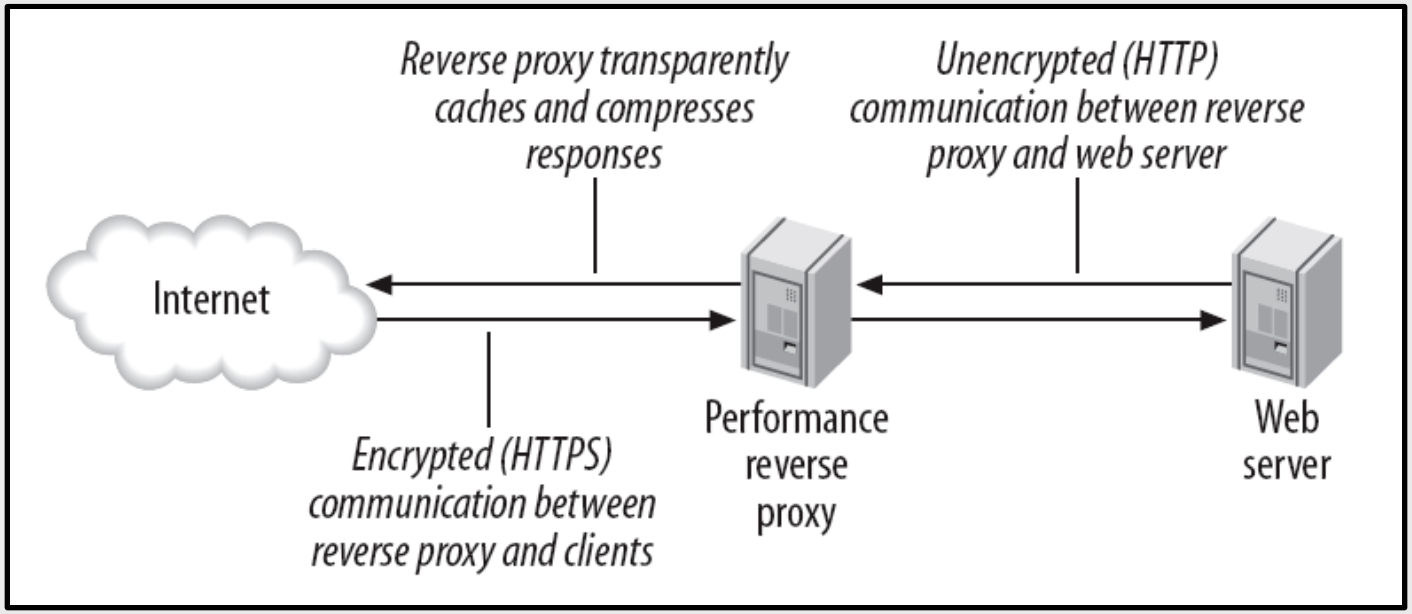
Protection Reverse Proxy (3/5)

- Observes traffic in and out
- Blocks invalid requests and attacks
- Prevents information disclosure



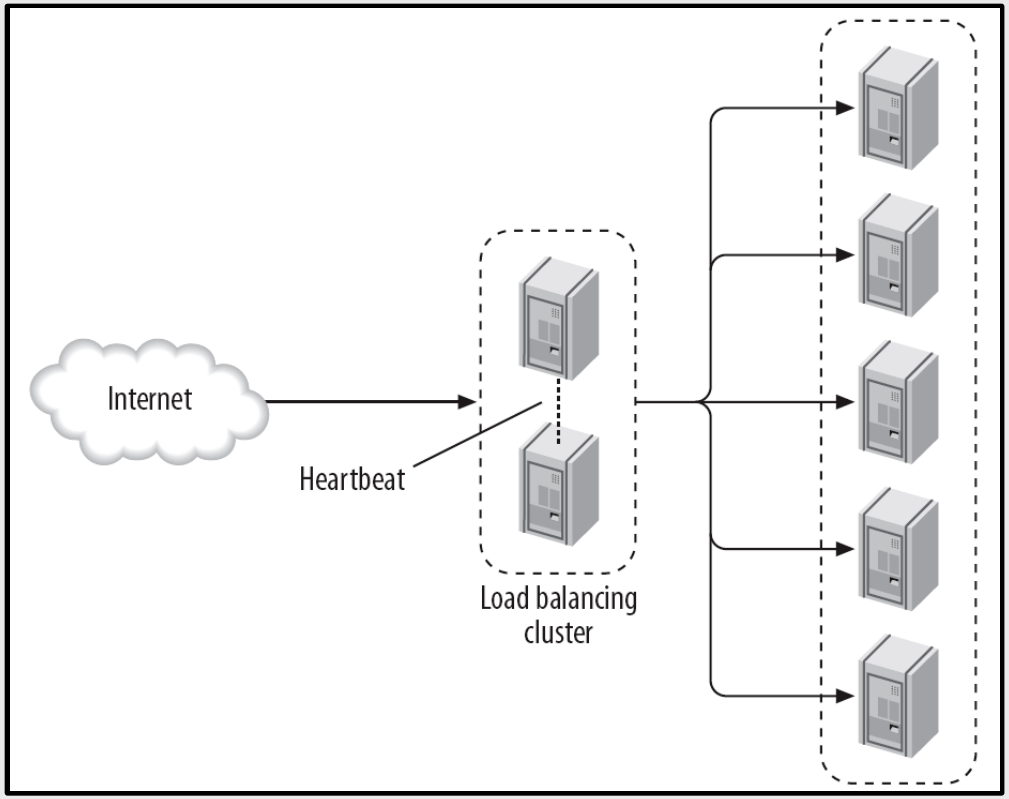
Performance Reverse Proxy (4/5)

- Transparent caching
- Transparent response compression
- SSL termination



Scalability Reverse Proxy (5/5)

- Load balancing
- Fault tolerance
- Clustering



Open Source Approach: Apache + ModSecurity

Apache

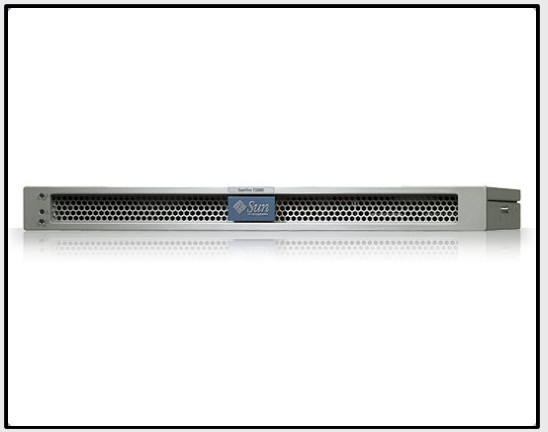
- One of the most used open source products.
- Available on many platforms.
- Free, fast, stable and reliable.
- Expertise widely available.
- Apache 2.2.x (finally!) released with many improvements:
  - Improved authentication.
  - Improved support for caching.
  - Significant improvements to the mod_proxy code (and load balancing support).
- **Ideal reverse proxy.**

ModSecurity

- Adds WAF functionality to Apache.
- In the 4th year of development.
- Free, open source, commercially supported.
- Implements most WAF features (and the remaining ones are coming soon).
- Popular and very widely used.
- Fast, reliable and predictable.

Apache + ModSecurity

- Deploy as **reverse proxy**:
  - Pick a nice server (I am quite fond of Sun's hardware offerings myself).
  - Install Apache 2.2.x.
  - Add ModSecurity.
  - Add SSL acceleration card (optional).
- Or simply run ModSecurity in **embedded mode**.



ModSecurity

- Strong areas:
  - **Auditing/logging support.**
  - **Real-time traffic monitoring.**
  - **Just-in-time patching.**
  - **Prevention.**
  - **Very configurable/programmable.**
- Weak areas:
  - **No automation of the positive security model approach yet.**

Questions?

Thank you!

Download this presentation from
http://www.thinkingstone.com/talks/