Threat Modelling for Web Application Deployment

Ivan Ristic
ivanr@webkreator.com
(Thinking Stone)

Talk Overview

1. Introducing Threat Modelling
2. Real-world Example
3. Questions

Who Am I?

- Developer / architect / administrator, spent a great deal of time looking at web security issues from different points of view.
- Author of **ModSecurity**, an open source web application firewall/IDS.
- Author of **Apache Security** (O'Reilly)
- Founder of **Thinking Stone**, a web security company.

State of Web Security

- It is a difficult job – web deployments consist of many different systems.
- Most decisions are made ad-hoc.
- Assumptions under which defence is designed are rarely challenged.
- Consequently, many systems are not adequately protected.
- We are in need of methodology that will help us design secure systems.
- **Threat Modelling** can do this.

Threat Modelling

1. Introduction To Threat Modelling

Threat Modelling

- Threat modelling is a semi-formal technique that is used to understand threats against your system.
- It is a hot, fashionable, buzzword!
- But it is genuinely useful and does not have to be difficult.
- **Not rocket science**.

Key Questions

- Where does your system live?
- What do you have to protect?
- Who are your users?
- Who are your adversaries?
- What are your weak points?
- What can you do to mitigate the threats?

Threat Modelling Advantage

Thinking like the adversary!

(What is wrong in this system
and how can I exploit it?)

Who Should Practice It?

- **Everyone!**
  - Developers.
  - System Administrators.
  - System Architects.
  - Consultants.

What Is It Good For?

- Planning.
- Testing (especially penetration testing).
- Training – my favourite.
- Security improvement.
  - It is never too late to start using it. Do try to use it as early as possible – it is much safer and cheaper that way.

Scope

- Full title of this talk: **A lightweight threat modelling methodology for web application deployment**.
- Includes a mixture of the following: network security, host security, web security, application security.
- Practical: **20%** effort – **80%** gain. (The remaining 80% of effort is mostly in details of web application security.)
- At this level **we treat web applications as black boxes**.

Methodology Overview

1. Information Gathering
2. Analysis
3. Mitigation

Methodology Overview

1. Information Gathering
   - Look at existing documents.
   - Interview stakeholders.
   - Inspect system.
   - **Understand system.**
2. Analysis
3. Mitigation

Methodology Overview

1. Information Gathering
2. Analysis
   - User roles and usage scenarios.
   - Components and trust boundaries.
   - Assets and attacker motivation.
   - Entry points, exit points, data flow.
   - Weaknesses and threats.
3. Mitigation

Methodology Overview

1. Information Gathering
2. Analysis
3. Mitigation
   - Establish budget.
   - Rank treats (fuzzy) - use a model that works for you.
   - Decide what to do with the threats.

Analysis: Stepping Stones

- Algorithm:
  1. Pretend you are the adversary.
  2. Look at the exposed parts.
  3. Find ways to subvert them.
  4. Find ways to use the resources available to you to get to the inner layers.
  5. Repeat until you grab the asset!
- Also known as **Attack Trees**.

Mitigation: Choices

- Ignore risk. (**Popular choice!**)
- Mitigate risk.
  - Intrusions are expensive
  - Security is expensive
  - **What can you afford?**
- Accept risk.

Mitigation: Strategies

- Remove entry points (attack vectors).
- Reduce attack surface.
- Compartmentalise.
- Practice the principle of least privilege.
- Fail safely.

Tips & Tricks

- Do not attempt to do too much - you might get lost. Branch out sub-models to cope with complexity, or work in iterations.
- Most web applications are similar - develop a library of reusable threat models.
- Start from scratch and assume nothing; mitigating the most obvious threats will result in foolproof operational procedures.

Threat Modelling

2. Real-world Example

Overview

- E-commerce operation:
  - Web site (CMS-powered)
  - Online store
- Two servers:
  - Application server
  - Database server

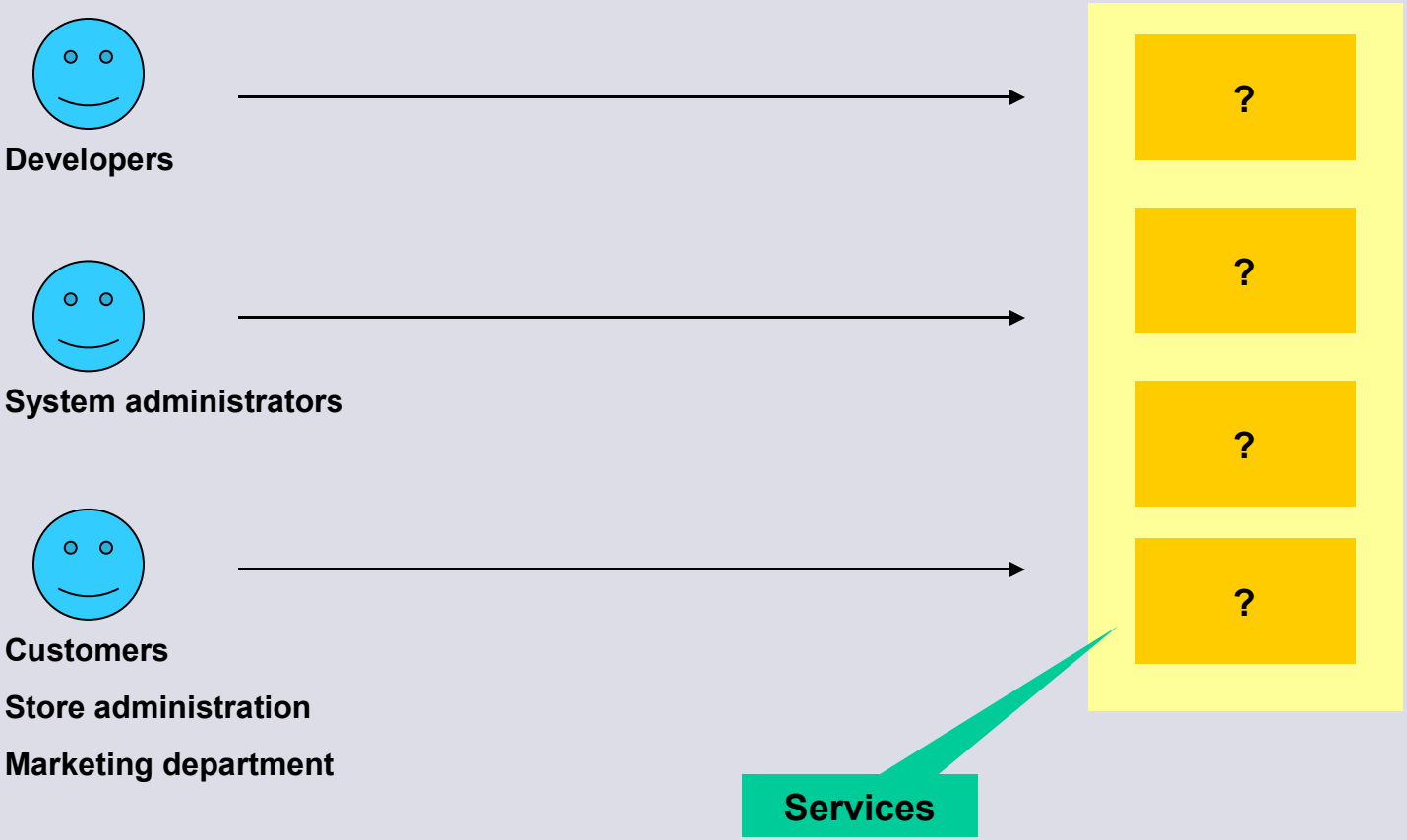
Physical Security

- Servers are collocated with a hosting company:
  - Restricted physical access
  - Biometrics at the entrance
  - Equipment in own locked cage
- Good, physical security is out of the scope of this talk anyway.

System Users (1)

- Customers (public).
- Store administration staff.
- Marketing department.
- Developers.
- System administrators.

System Users (2)



System Users (3)

SSH (22), FTP (20, 21), MYSQL (3306)

Developers (manage application, maintain and backup database): 2

SSH (22)

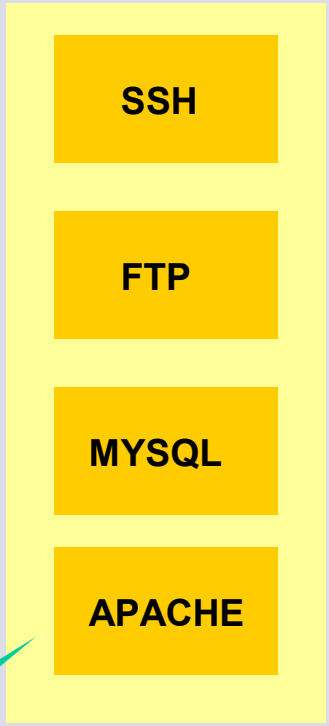
System administrators (manage servers): 2

Apache (80)

Public (browse web site, buy stuff): 1 BILLION

Store administration (manage store): 4

Marketing department (manage web site): 1



System Users (4)

- Threat: possible password and data compromise through the use of plain-text communication protocols.
- Mitigation: Disallow plain-text protocols:
  - Shopping in the store
  - Administrative interfaces (Store, CMS)
  - Database access

System Users (5)

SSH (22), SFTP (115), MYSQL+SSL (3306)

Developers (manage application, maintain and backup database): 2

SSH (22)

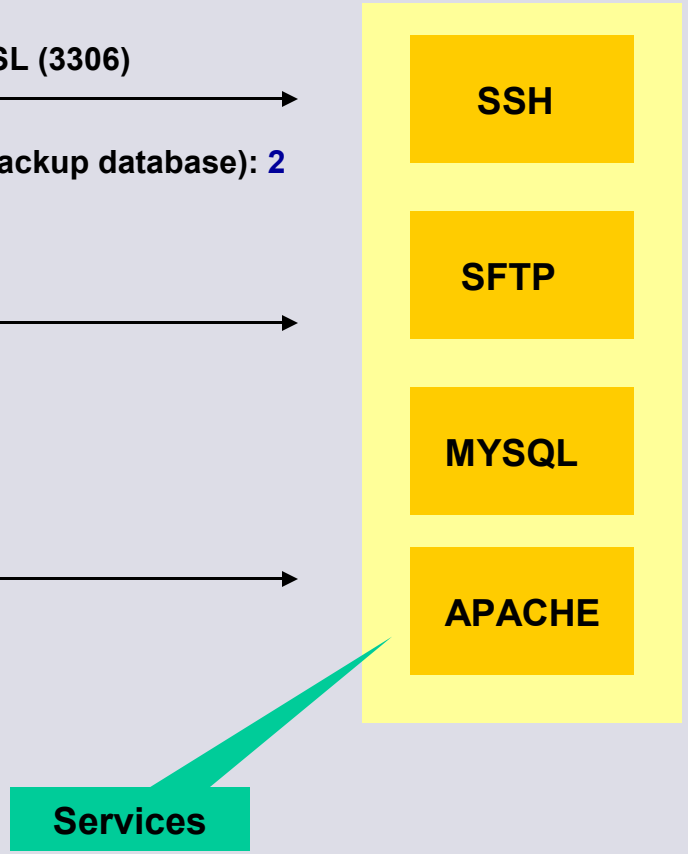
System administrators (manage servers): 2

Apache (80), Apache+SSL (443)

Public (browse web site, buy stuff): 1 BILLION

Store administration (manage store): 4

Marketing department (manage web site): 1



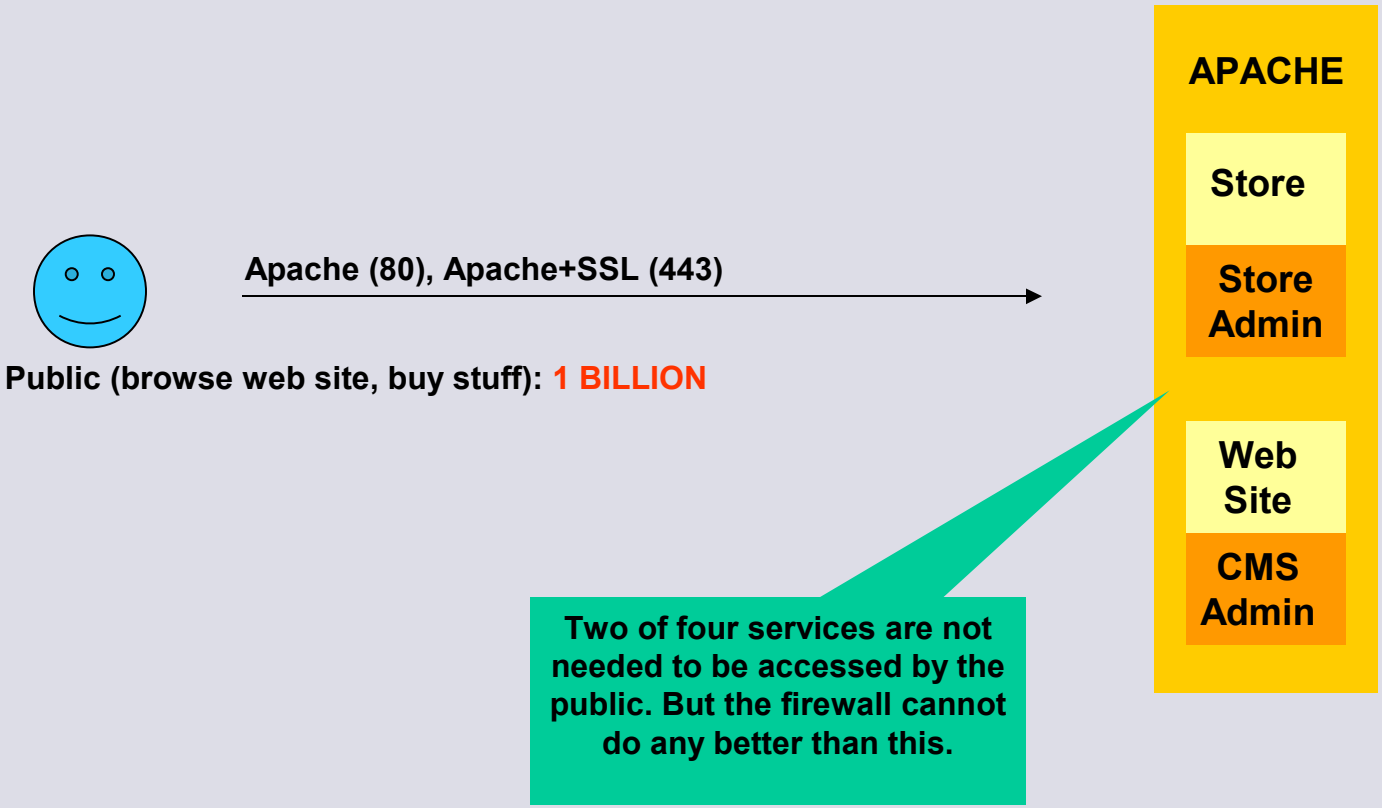
Entry Points (1)

- On the network level, each service represents one entry point.
- Implement firewall restrictions to allow only what is absolutely necessary:
  - External firewall (hosting company).
  - Host firewalls (iptables on Linux).
  - Leave no trust between two internal servers either, only let port 3306 through.

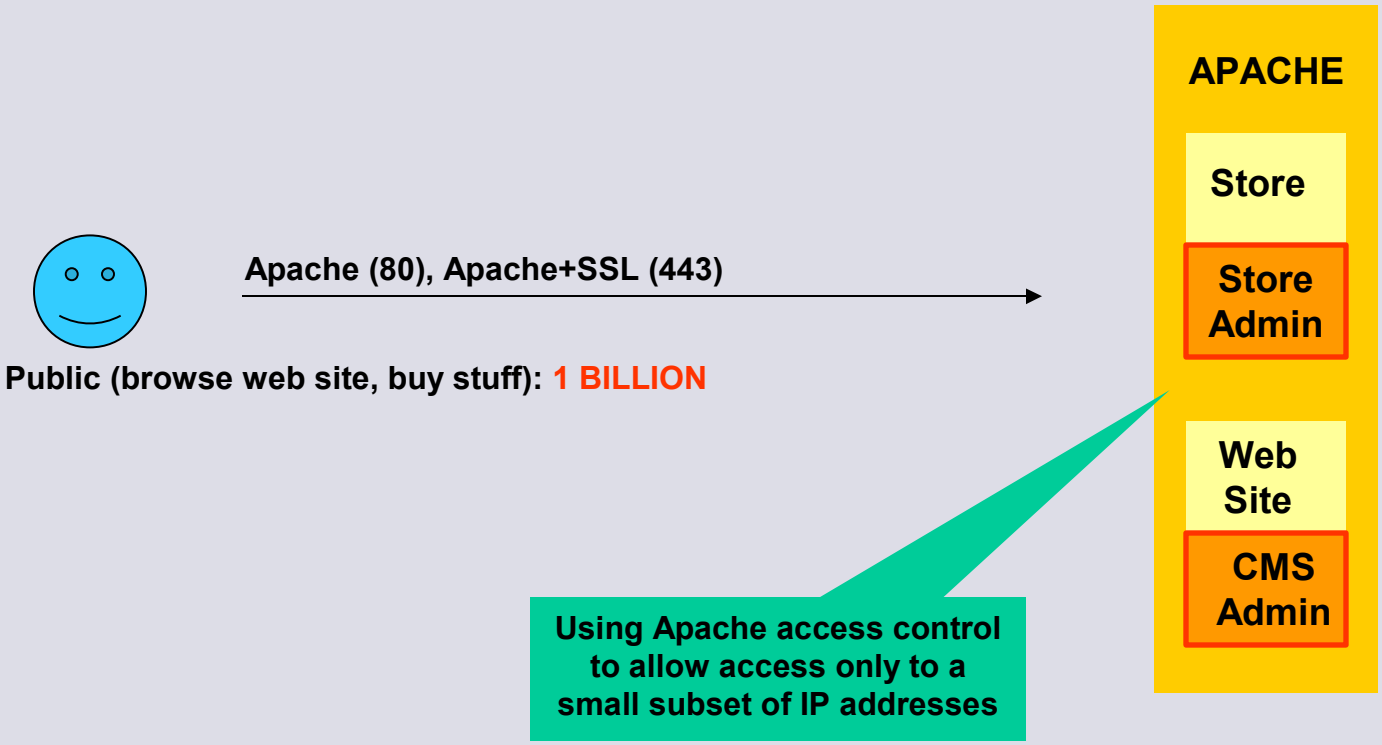
Entry Points (2)

- Threat: possible compromise through vulnerabilities in Apache, SFTPD, SSHD, and MySQL.
- Mitigation: Prevent access to non-essential services (SFTPD, SSHD, MySQL).
- Option: buy an expensive hardware firewall.

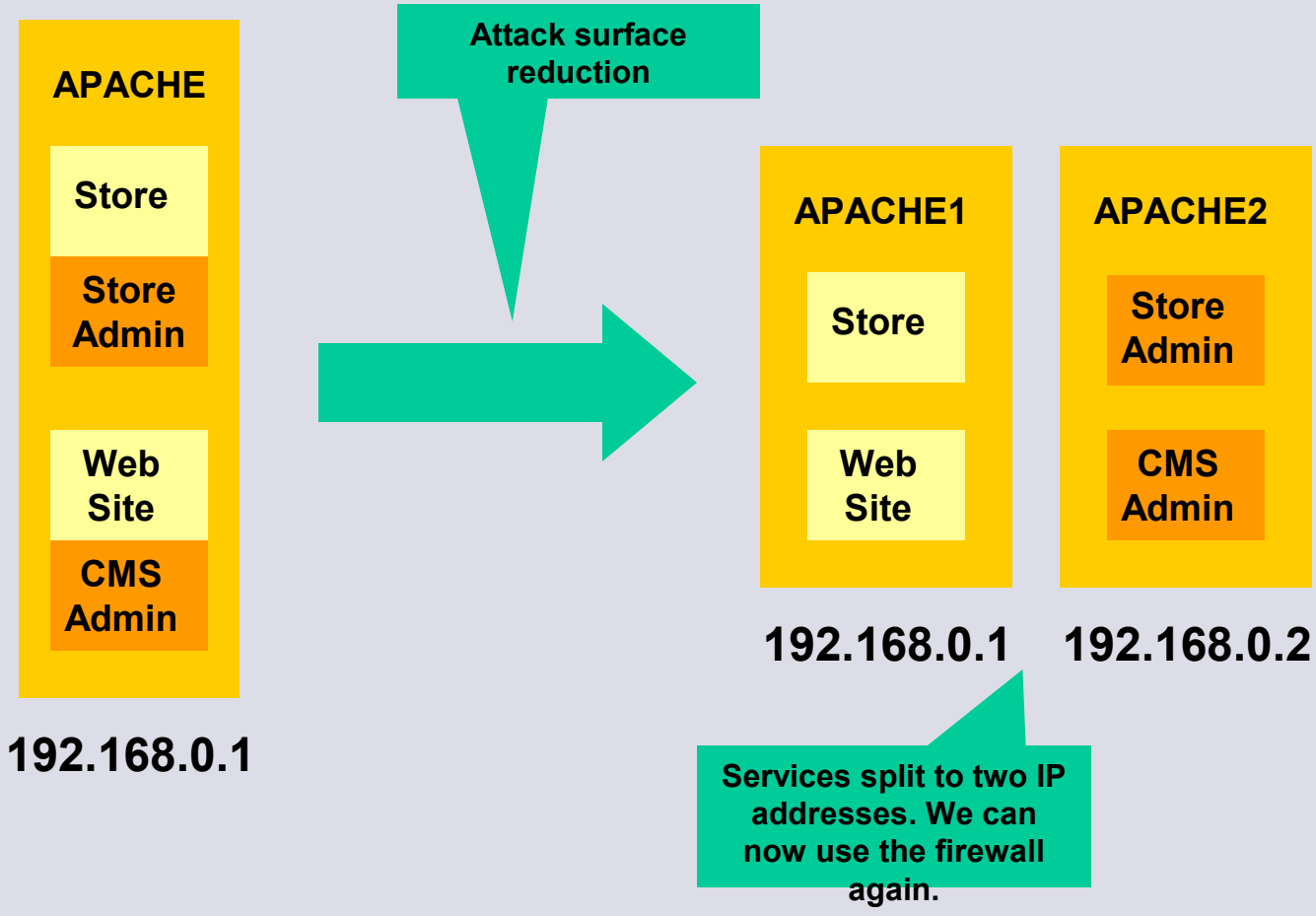
What the Public Now Sees (1)



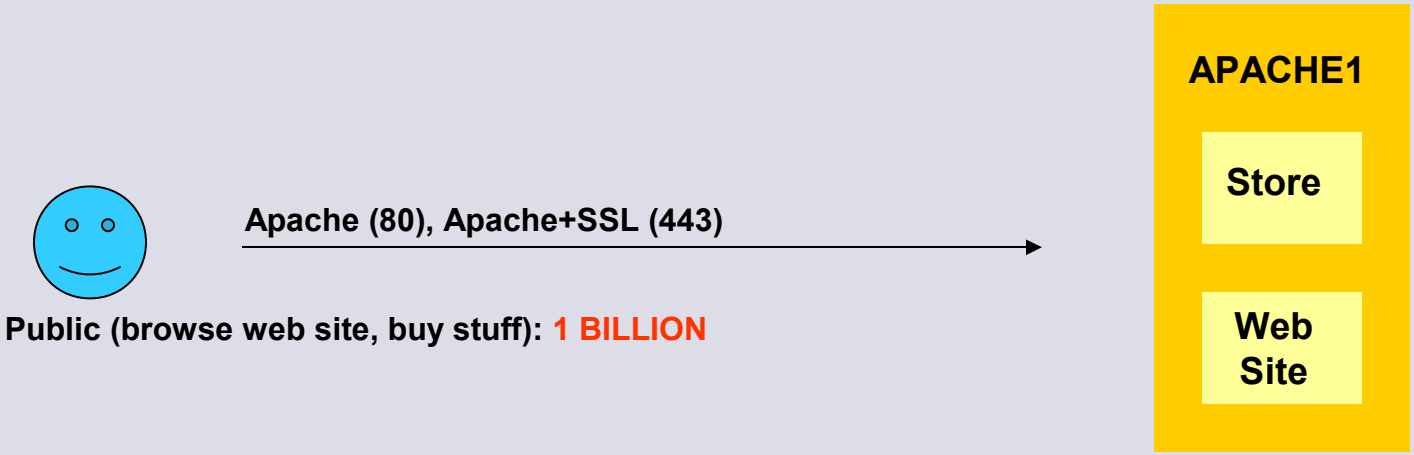
Attack Surface Reduction (1)



Attack Surface Reduction (2)



What the Public Now Sees (2)



Entry Points (4)

- Threat: possible compromise through vulnerabilities in Apache.
- Mitigation: Keep Apache up-to-date:
  - Automated patching (use binary Apache)
  - Manual patching (build Apache from source)

Entry Points (5)

- Threat: possible compromise through Apache misconfiguration.
- Mitigation: Configuration management.
- Mitigation: Regular independent configuration assessments.

Entry Points (6)

- Threat: possible compromise through unmitigated Apache problems.
- Mitigation: Put Apache in jail.
- Mitigation: Implement integrity validation.
- Mitigation: Implement kernel patches (e.g. grsecurity).

Remaining Assets (1)

- Threat: Adversary accesses the credit card database.
- Mitigation: Do not store credit cards online, or store them using public-key encryption. (**We lowered the value of the asset**.)
- Mitigation: Document policy on the web site.

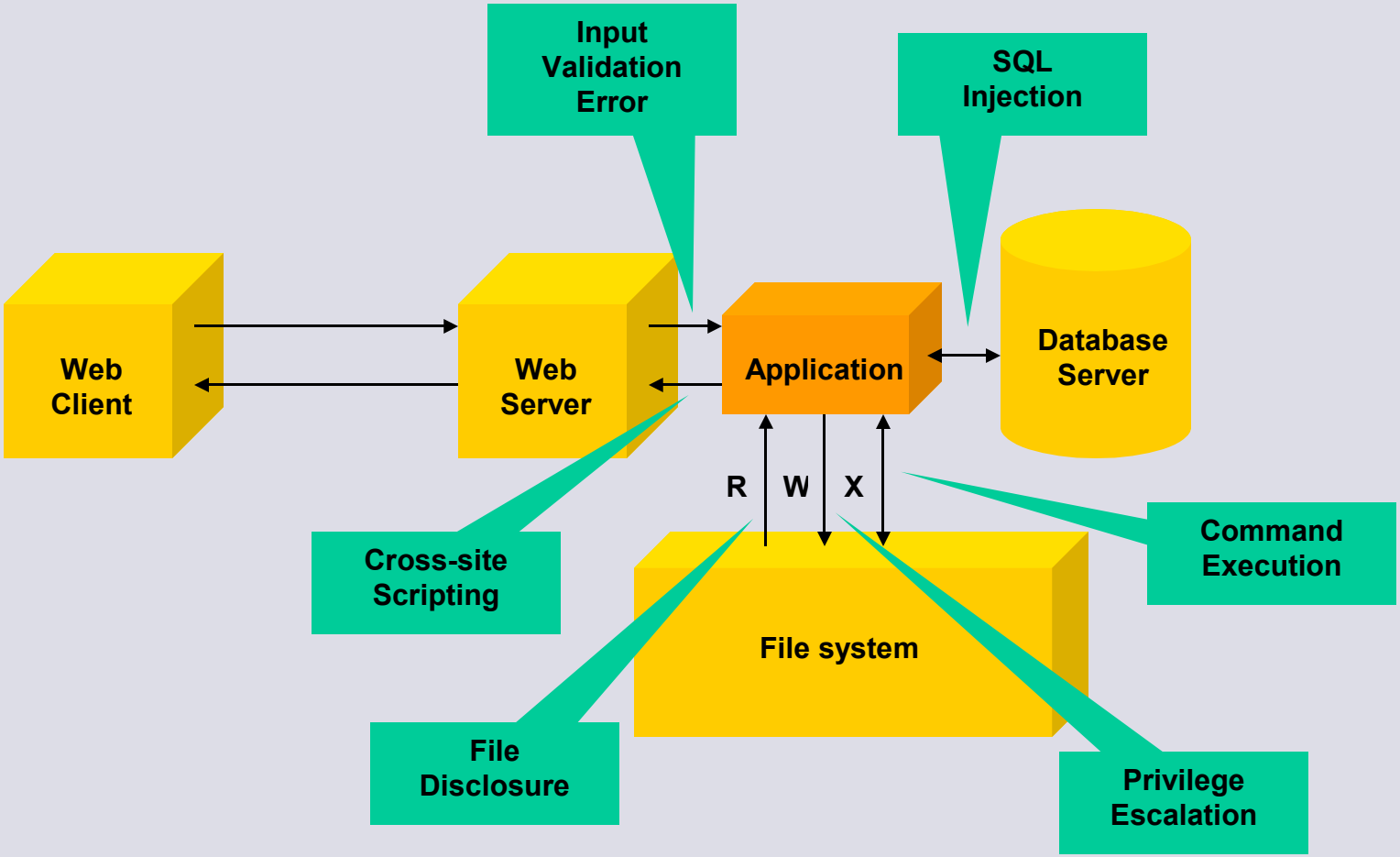
Remaining Assets (2)

- Threat: Source code stolen.
- Mitigation: Do not keep the source code online, compile PHP pages before uploading. (Again, we lowered the value of the asset.)

Threat That Remains…

- Threat: Compromise through a vulnerability of the application.
- This opens a door to a new threat modelling sub-model: **web application security**.
- Mitigate from the outside. Treat the application as a black box, and look where action gets out.

Web Application Model



Security Categories

- Data validation and transformation
- Authentication and authorisation
- Sensitive data transport and storage
- Session management
- Fault management
- Auditing and logging

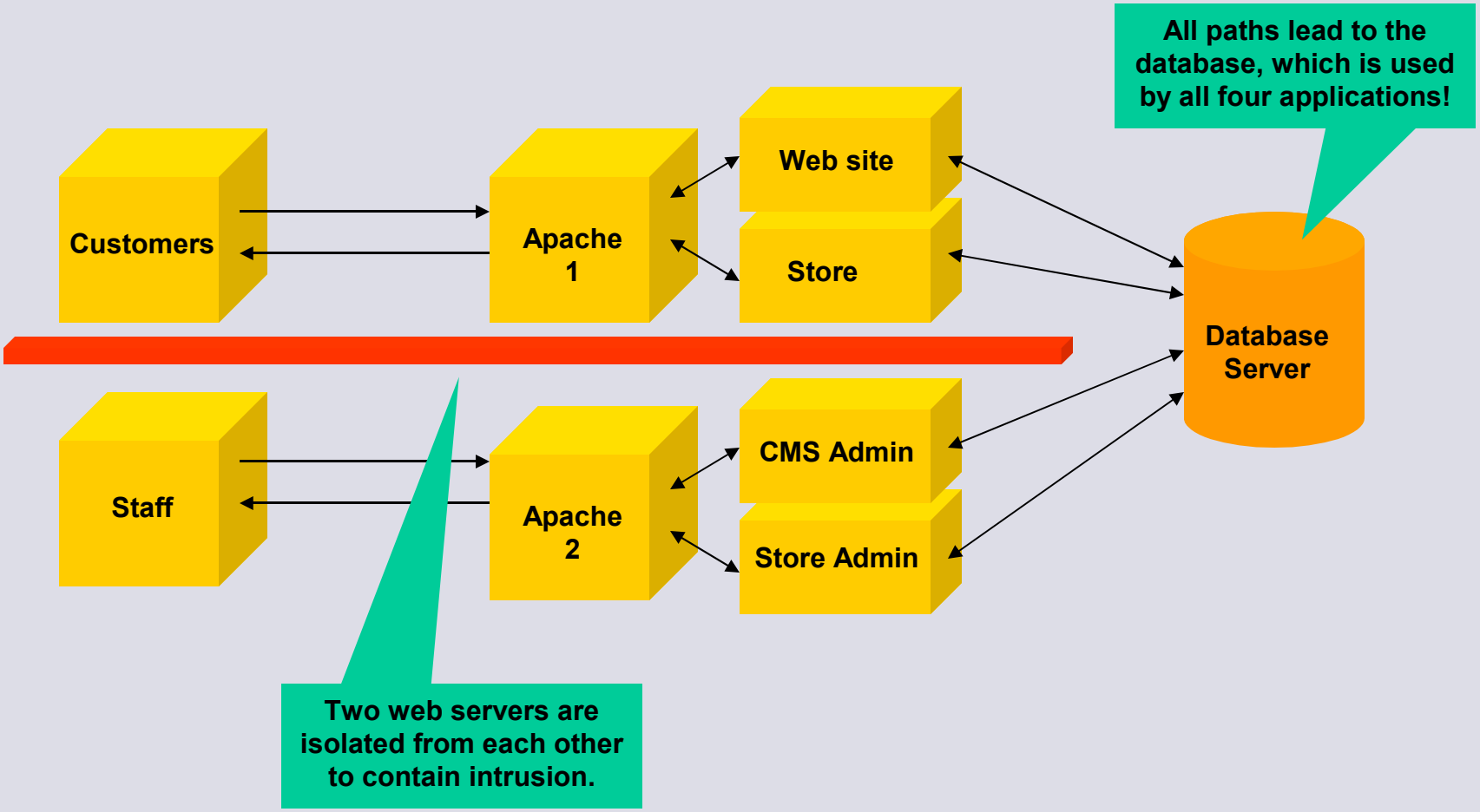
Fix Filesystem Permissions

- Do not allow read access – prevents file disclosure.
- Do not allow write access – prevents privilege escalation.
- Do not have binaries or a compiler around – prevents command execution.

Screen System Boundaries

- If you can’t fix the application focus on libraries that interact with external systems.
- Screen database queries.
- Screen external command execution.
- Screen file system operations.

Examine Attack Paths



Central Database Mitigation

- Have **four database accounts**, each with different access level.
- Or, deploy a **separate database engine** for the CMS application (web site).

Final Mitigation Activities…

- Know when you are compromised
  - Activity monitoring
  - Integrity Validation
  - Intrusion Detection
- Have off-site backups and disaster recovery procedures!

Where To Go From Here

- Chapter 3 of “Improving Web Application Security: Threats and Countermeasures” (free download).
- Chapter 4 of “Writing Secure Code”.
- If you are a programmer, read “Threat Modeling”.
- Use the free threat modelling tool from Microsoft.
- More here: http://www.modsecurity.org/blog/archives/2006/01/threat_modellin.html

Questions?

Thank you!

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