Introduction to the OWASP Top Ten

...

Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

Recordings:
https://goo.gl/a2VSG2

OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz

What is OWASP?

Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software.

- A website: owasp.org
- A bunch of cool tools: Zed Attack Proxy, Juice Shop, Proactive Controls, Software Assurance Maturity Model (SAMM), Application Security Verification Standard (ASVS)
- A global community of like-minded people, meetups and conferences

Who is the OWASP® Foundation?

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

- Tools and Resources
- Community and Networking
- Education & Training

For nearly two decades corporations, foundations, developers, and volunteers have supported the OWASP Foundation and its work. Donate, Join, or become a Corporate Member today.

Project Spotlight: OWASP Top 10

We are back again with yet another OWASP Spotlight series and this time we have a project which needs no introduction and I got the chance to interact with Andrew van der Stock, OWASP Foundation Executive Director and

OWASP 2022 Global AppSec European Virtual Event

OWASP Top 10:2021

Home

Notice

Introduction

How to use the OWASP Top 10 as a standard

How to start an AppSec program with the OWASP Top 10

About OWASP

Top 10:2021 List

A01 Broken Access Control

A02 Cryptographic Failures

A03 Injection

A04 Insecure Design

A05 Security Misconfiguration

A06 Vulnerable and Outdated Components

A07 Identification and Authentication Failures

A08 Software and Data Integrity Failures

A09 Security Logging and Monitoring Failures

A10 Server Side Request Forgery (SSRF)

Next Steps

Introduction

Welcome to the OWASP Top 10 - 2021

Welcome to the latest installment of the OWASP Top 10! The OWASP Top 10 2021 is all-new, with a new graphic design and an available one-page infographic you can print or obtain from our home page.

A huge thank you to everyone that contributed their time and data for this iteration. Without you, this installment would not happen. **THANK YOU!**

Table of contents

Welcome to the OWASP Top 10 - 2021

What's changed in the Top 10 for 2021

Methodology

How the categories are structured

How the data is used for selecting categories

Why not just pure statistical data?

Why incidence rate instead of frequency?

What is your data collection and analysis process?

Data Factors

Thank you to our data contributors

Thank you to our sponsors

https://owasp.org/Top10/

OWASP Top Ten

Globally recognized by developers as the first step towards more secure coding.

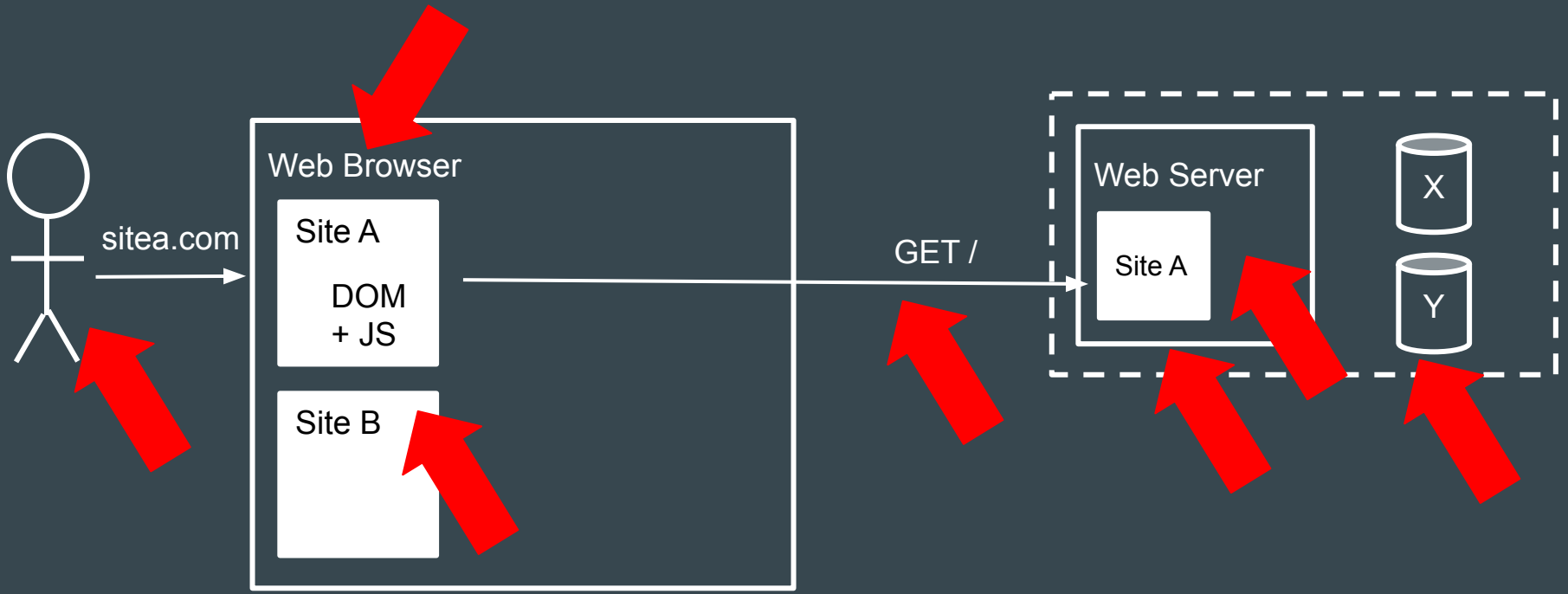
The *most critical* security risks to web applications.

Updated every 2-3 years from 2003 to 2021

2021: Focus on the root cause, rather than the symptom

See: The How and Why of the OWASP Top Ten 2021 - Brian Glas

Securing the user



OWASP Top Ten 2021

A01	Broken Access Control
A02	Cryptographic Failures
A03	Injection
A04	Insecure Design
A05	Security Misconfiguration
A06	Vulnerable and Outdated Components
A07	Identification and Authentication Failures
A08	Software and Data Integrity Failures
A09	Security Logging and Monitoring Failures
A10	Server-Side Request Forgery

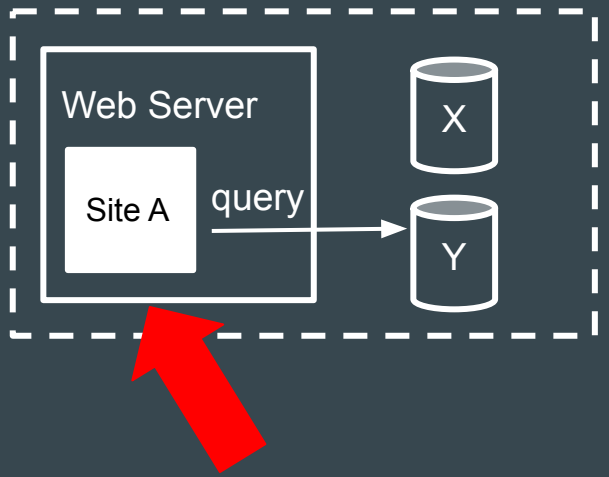
A01 Broken Access Control

A01 Broken Access Control

- Access hidden pages

  `http://site.com/admin/user-management`
- Elevate to an administrative account
- View other people's data

  `http://site.com/user?id=7`
- Modifying cookies or JWT tokens



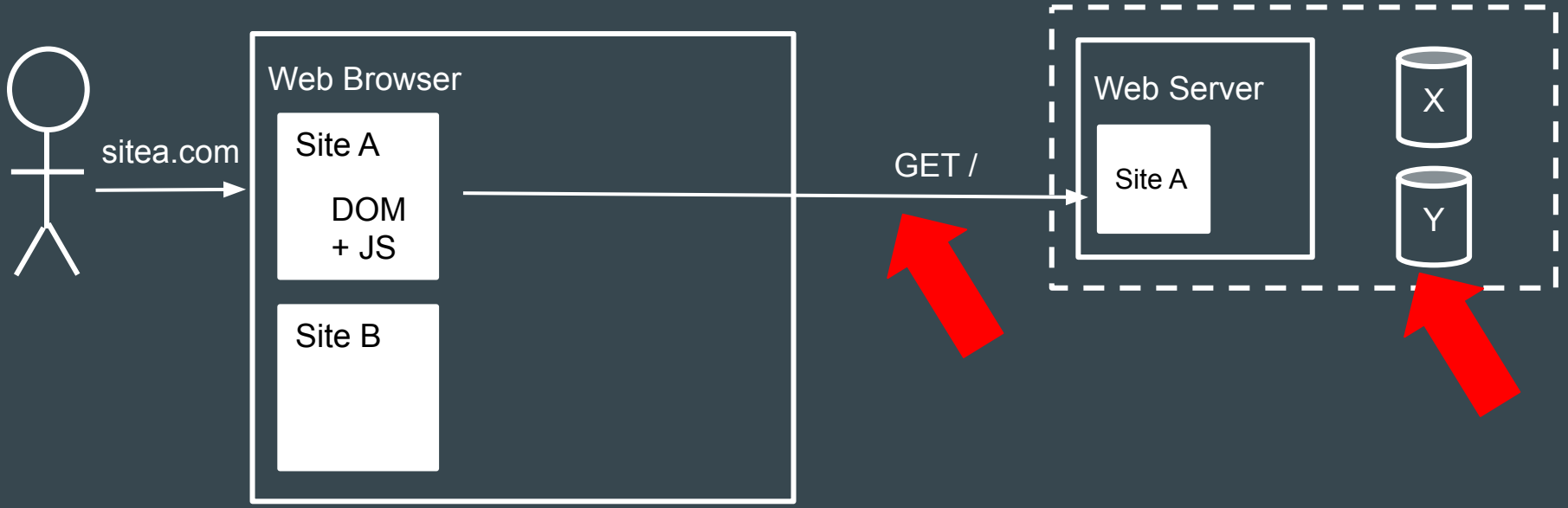
Access Control Demo

A01 Broken Access Control

Prevention:

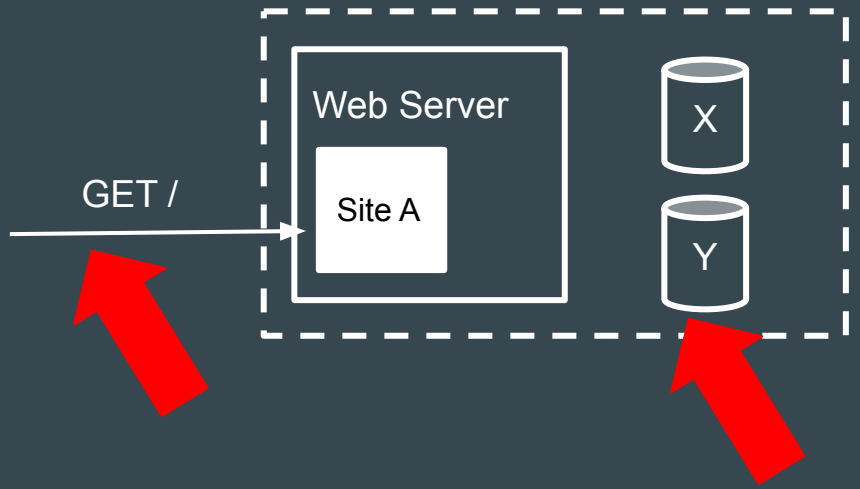
- Implement access control measures centrally
- Use proven code or libraries
- Deny access by default
- Log failures and alert
- Rate limit access to resources

A02 Cryptographic Failure



A02 Cryptographic Failure

- Clear-text data transfer
- Unencrypted storage
- Weak crypto or keys
- Certificates not validated
- Exposing PII or Credit Cards



A02 Cryptographic Failure

Prevention:

- Don’t store data unless you need to!
- Encrypt at rest and in transit
- Use strong crypto

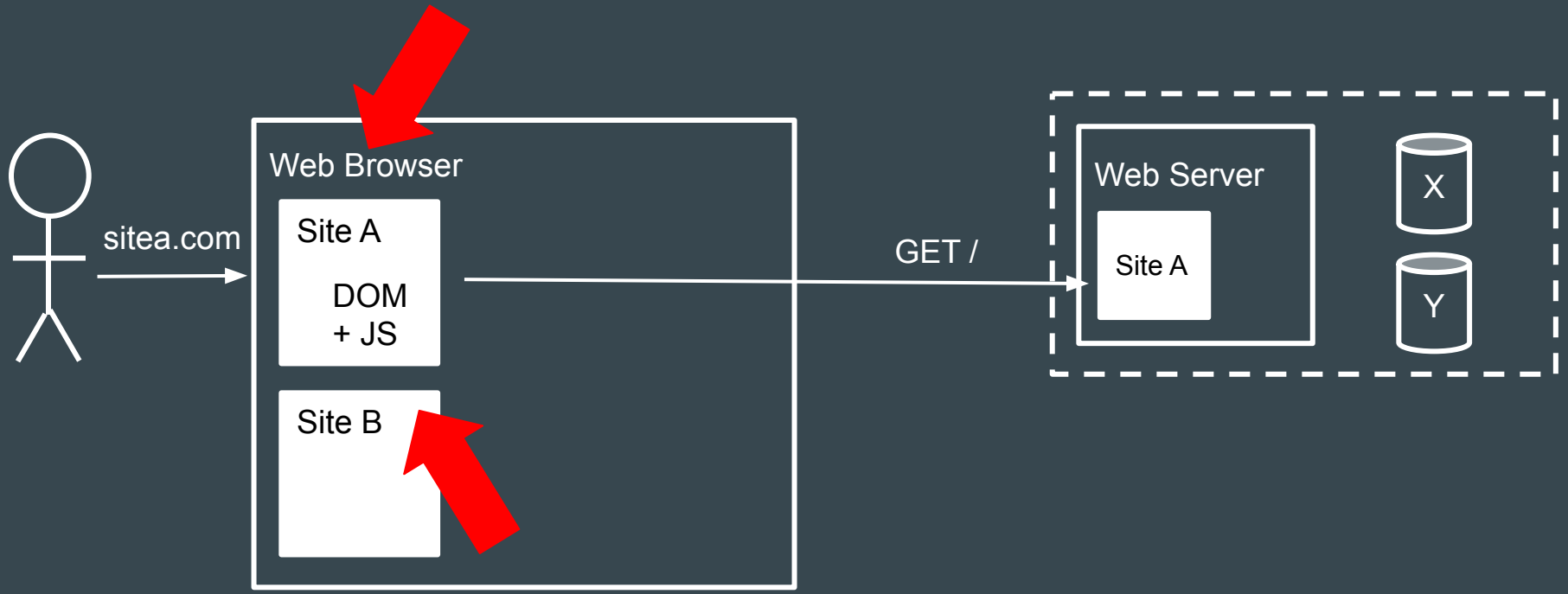
A03 Injection

Injecting attacker-controlled *data* into the *code* you intend to run

Examples:

- Cross-Site Scripting (XSS)
- SQL Injection (SQLi)

A03 Injection - Cross-Site Scripting (XSS)

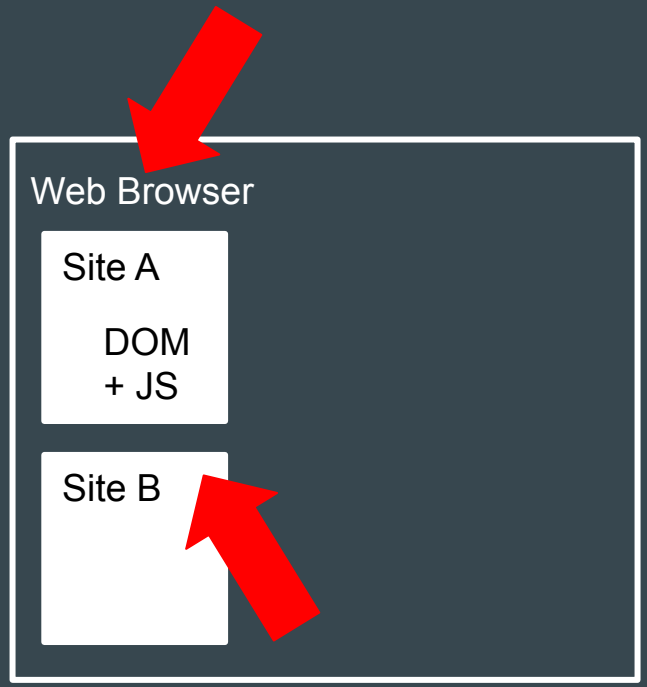


A03 Injection - Cross-Site Scripting (XSS)

HTML mixes content, presentation and code into one string (HTML+CSS+JS)

If an attacker can alter the DOM, they can do *anything* that the user can do.

XSS can be found using automated tools.



XSS Demo

A03 Injection - Cross-Site Scripting (XSS)

Prevention:

- Encode all user-supplied data to render it safe

  `Kirk <script> => Kirk <script>`
- Use appropriate encoding for the context
- Use templating frameworks that assemble HTML safely
- Use Content Security Policy

A03 Injection - SQLi

Sending hostile data to an interpreter
(e.g. SQL, LDAP, command line)

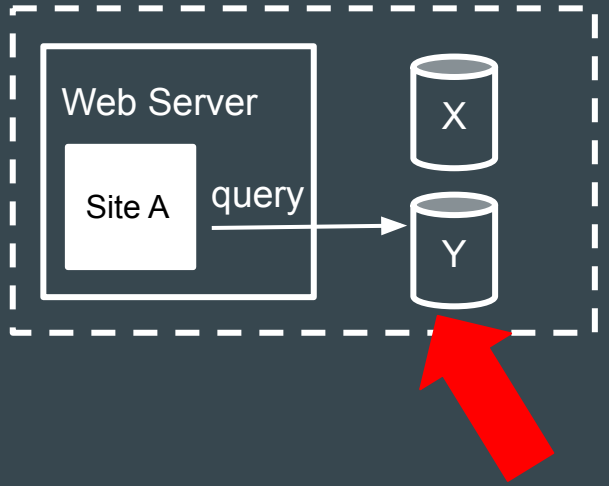
A03 Injection - SQLi

Sending hostile data to an interpreter (e.g. SQL, LDAP, command line)

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";

id = " '; drop table accounts -- "
```

SQL statements combine *code* and *data*



SQLi Demo

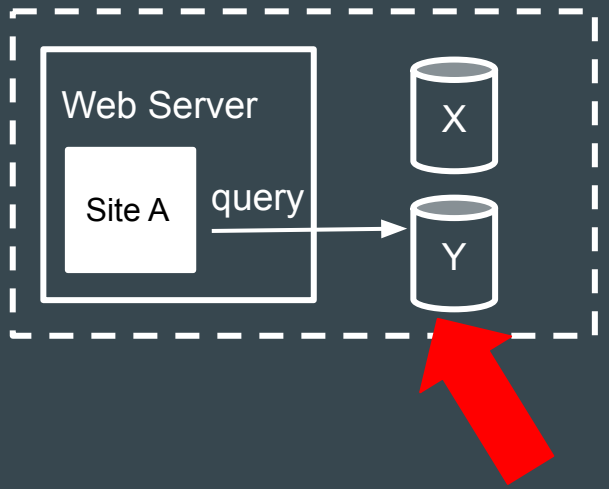
A03 Injection - SQLi

Prevention:

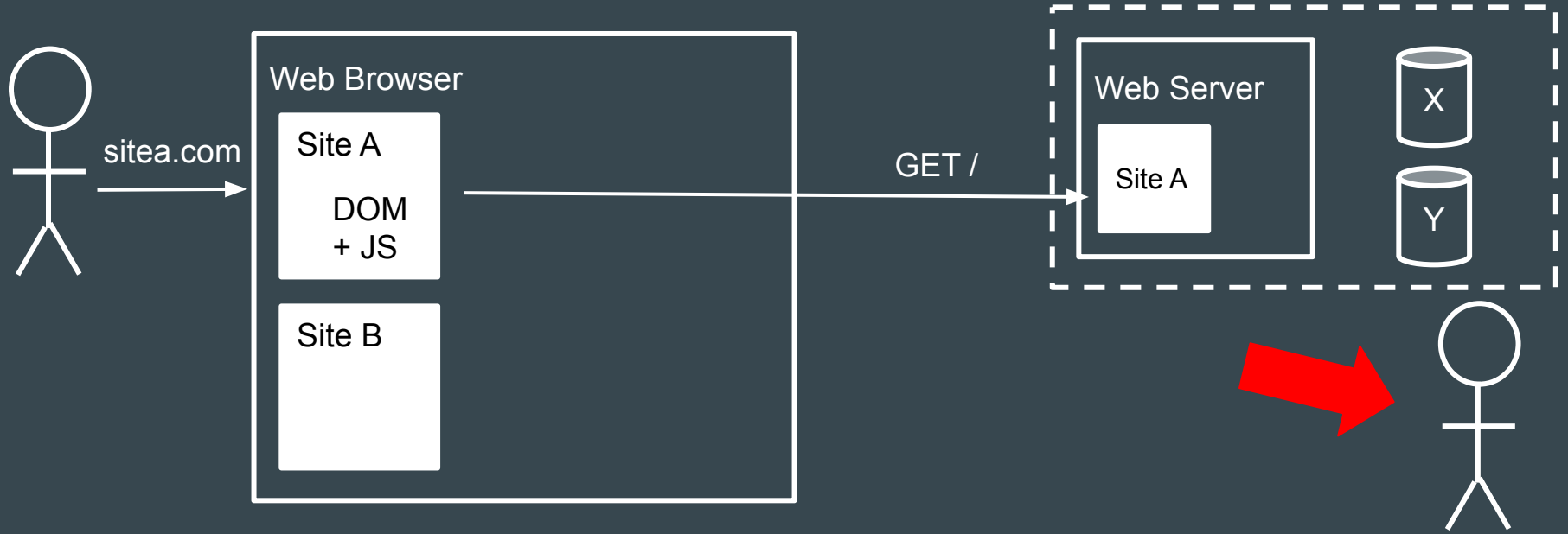
SQL statements combine *code* and *data*

=> Separate code and data

- Parameterise your queries
- Validate which data can be entered
- Escape special characters



A04 Insecure Design



A04 Insecure Design

Risks related to design and architectural flaws

Cannot be fixed by rock-solid implementation

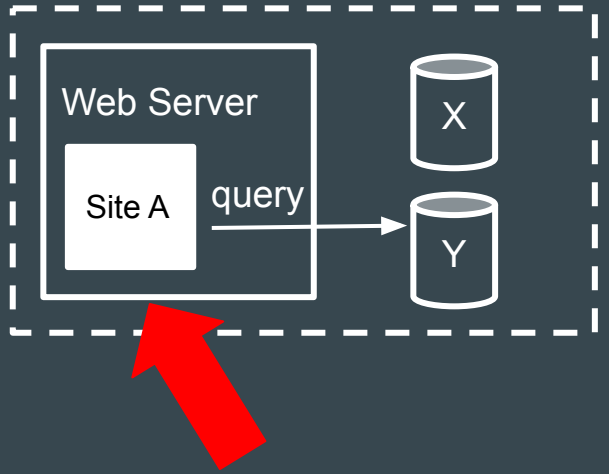
Use:

- Threat modeling
- Secure design patterns
- Reference architectures

A05 Security Misconfiguration

A05 Security Misconfiguration

- Security features not configured properly
- Unnecessary features enabled
- Default accounts not removed
- Error messages expose sensitive information



A05 Security Misconfiguration

Prevention:

- Have a repeatable build process or “gold master”
- Disable all unused services
- Use tools to review settings

A06 Vulnerable and Outdated Components

A06 Vulnerable and Outdated Components

Modern applications contain a *lot* of third-party code.

It's hard to keep it all up to date.

Attackers can enumerate the libraries you use, and develop exploits.

A06 Vulnerable and Outdated Components

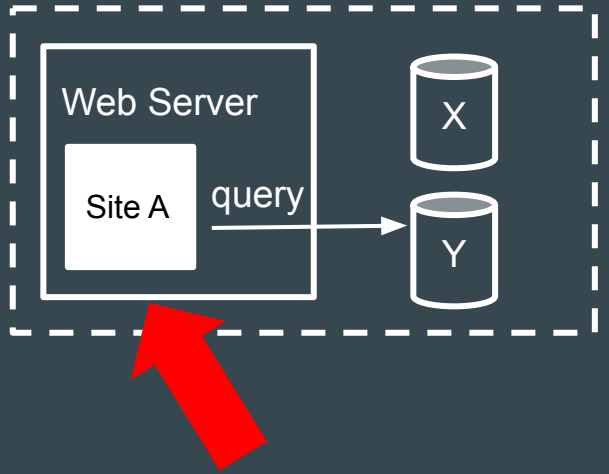
Prevention:

- Inventory management
- Reduce dependencies
- Patch management
- Scan for out-of-date components
- Budget for ongoing maintenance for all software projects

A07 Identification and Authentication Failures

A07 Identification and Authentication Failures

- Weak session management
- Credential stuffing
- Brute force
- Forgotten password
- No multi-factor authentication
- Sessions don’t expire



Identity and Authentication Failures Demo

A07 Identification and Authentication Failures

Prevention:

- Use good authentication libraries
- Use MFA
- Enforce strong passwords
- Detect and prevent brute force or stuffing attacks

A08 Software and Data Integrity Failures

A08 Software and Data Integrity Failures

Software integrity:

- Downloading code from untrustworthy sources
- No integrity checks
- Insecure CI/CD pipeline

Data integrity:

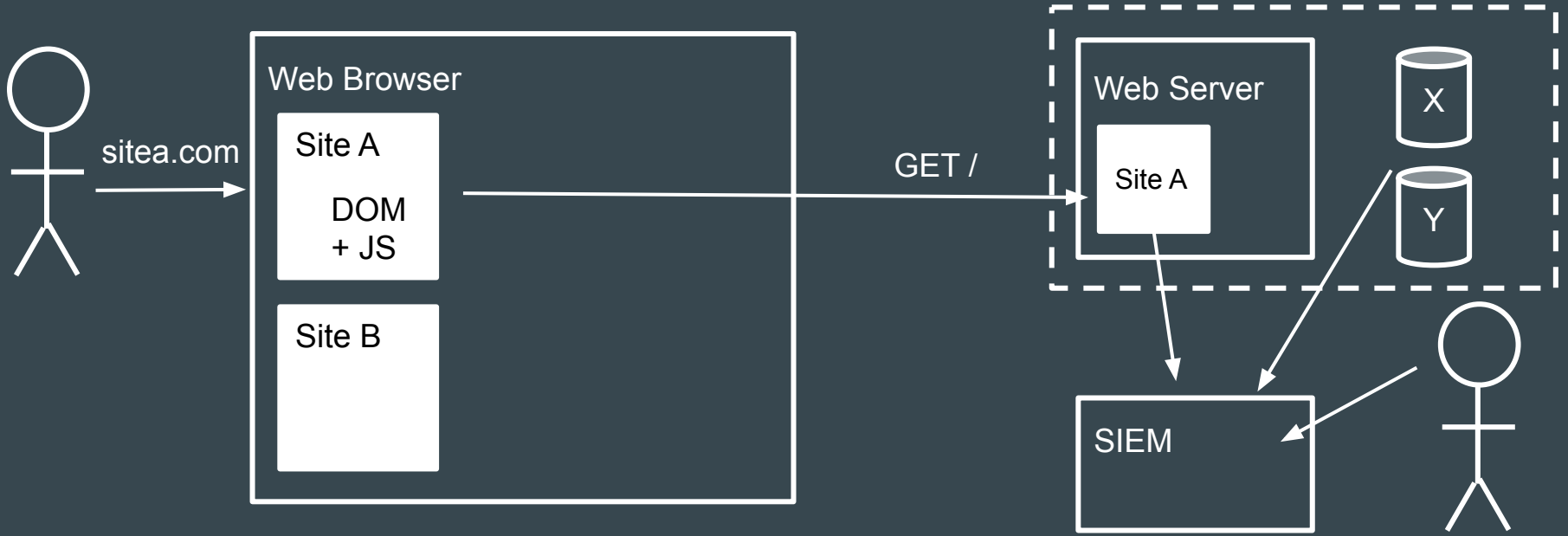
- Data may be modified for deserialisation attack

A08 Software and Data Integrity Failures

Prevention:

- Digital signatures for libraries and executables
- Use trustworthy repositories
- Supply chain dependency check
- Encrypt data, and check integrity

A09 Security Logging and Monitoring Failures

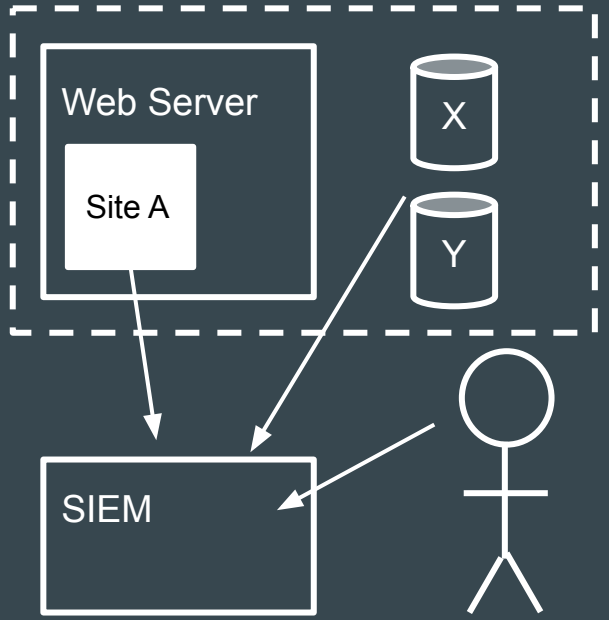


A09 Security Logging and Monitoring Failures

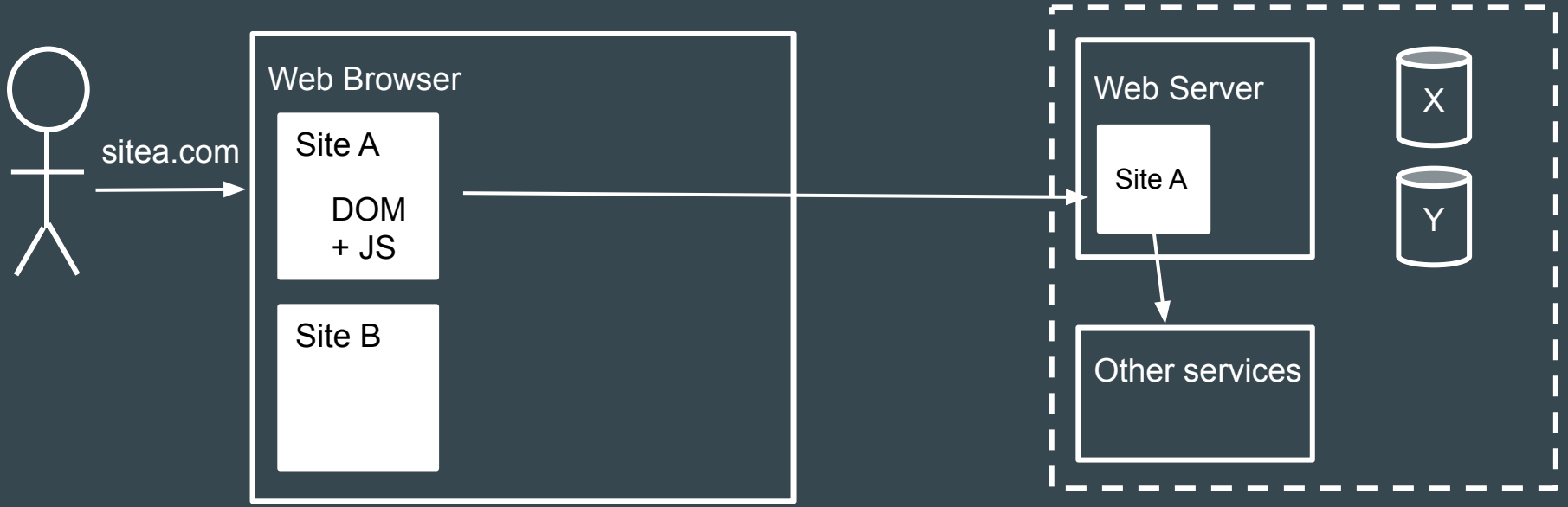
You can’t react to attacks that you don’t know about.

Logs are important for:

- Detecting incidents
- Understanding what happened
- Proving who did something



A10 Server-Side Request Forgery

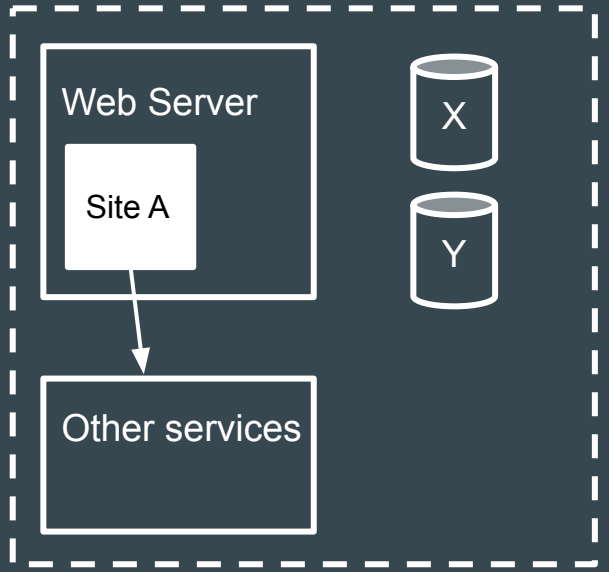


A10 Server-Side Request Forgery

Tricking an application to fetch something by url

E.g.

- Access an internal service
- Port-scan a network
- Access cloud metadata service
- Proxy attacks to other targets



SSRF Demo

A10 Server-Side Request Forgery

Prevention:

- Segment networks, firewall restrictions
- Don’t trust input data
- Do not display raw HTTP responses to clients
- Don’t follow redirects

Server-Side Request Forgery (SSRF): The community push to the OWASP Top 10
Nick Lauder, Track Two - Thursday, 15:30

OWASP Top Ten 2021

A01	Broken Access Control
A02	Cryptographic Failures
A03	Injection
A04	Insecure Design
A05	Security Misconfiguration
A06	Vulnerable and Outdated Components
A07	Identification and Authentication Failures
A08	Software and Data Integrity Failures
A09	Security Logging and Monitoring Failures
A10	Server-Side Request Forgery

Next Steps

Next Steps

- Attend OWASP events
- Search for OWASP Top Ten category names and your framework
  E.g. “C# XSS protection”
- Watch youtube or Pluralsight videos
- Use the terms when discussing bugs with colleagues
- Keep track of which issues affect you the most
- Go beyond the Top Ten

Introduction to the OWASP Top Ten

...

Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

Recordings:
https://goo.gl/a2VSG2

OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz