Web Application Security for Absolute Beginners *(No coding!)*

10 most common cyber attacks everybody should know because every hackers does!

Why should you follow this course?

- Who should follow this course?
  - Every self-respecting project manager, product owner and developer

- After finishing this course you will
  - Understand the 10 most common web application attacks, their impact and how these attacks can be prevented or mitigated
  - Have access to technical documents that prevent or mitigate these attacks

Introduction: OWASP 10 (2017*)

Open **W**eb **A**pplications **S**ecurity **P**roject

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Broken Access Control
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Insufficient Attack Protection
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Underprotected APIs

* The top 10 changes frequently

Introduction: OWASP 10 *UPDATED*

NEW

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient logging & monitoring

OLD

1. Injection
2. Broken Authentication ~~and Session Management~~
3. Cross-Site Scripting (XSS)
4. ~~Broken Access Control~~
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Insufficient Attack Protection
8. ~~Cross-Site Request Forgery (CSRF)~~
9. Using Components with Known Vulnerabilities
10. ~~Underprotected APIs~~

1. Injection

1. **What is it?** Untrusted user input is interpreted by server and executed
2. **What is the impact?** Data can be stolen, modified or deleted
3. **How to prevent?**
   - Reject untrusted/invalid input data
   - Use latest frameworks
   - Typically found by penetration testers / secure code review

Shar, L. K., & Tan, H. B. K. (2013). Defeating SQL injection. *Computer*, *46*(3), 69-77.

1. Injection (example)

- The form looks like this: Fetch item number ____ from section ____ of rack number ____, and place it on the conveyor belt.

- A normal request might look like this: Fetch item number **222** from section **A2** of rack number **11**, and place it on the conveyor belt.

- What if the user added instructions into them? Fetch item number **223** from section **A2** of rack number **11, and throw it out the window.** and place it on the conveyor belt.

Source: stackexchange.com

More info: Huang, Y. W., Huang, S. K., Lin, T. P., & Tsai, C. H. (2003, May). Web application security assessment by fault injection and behavior monitoring. In *Proceedings of the 12th international conference on World Wide Web* (pp. 148-159). ACM.

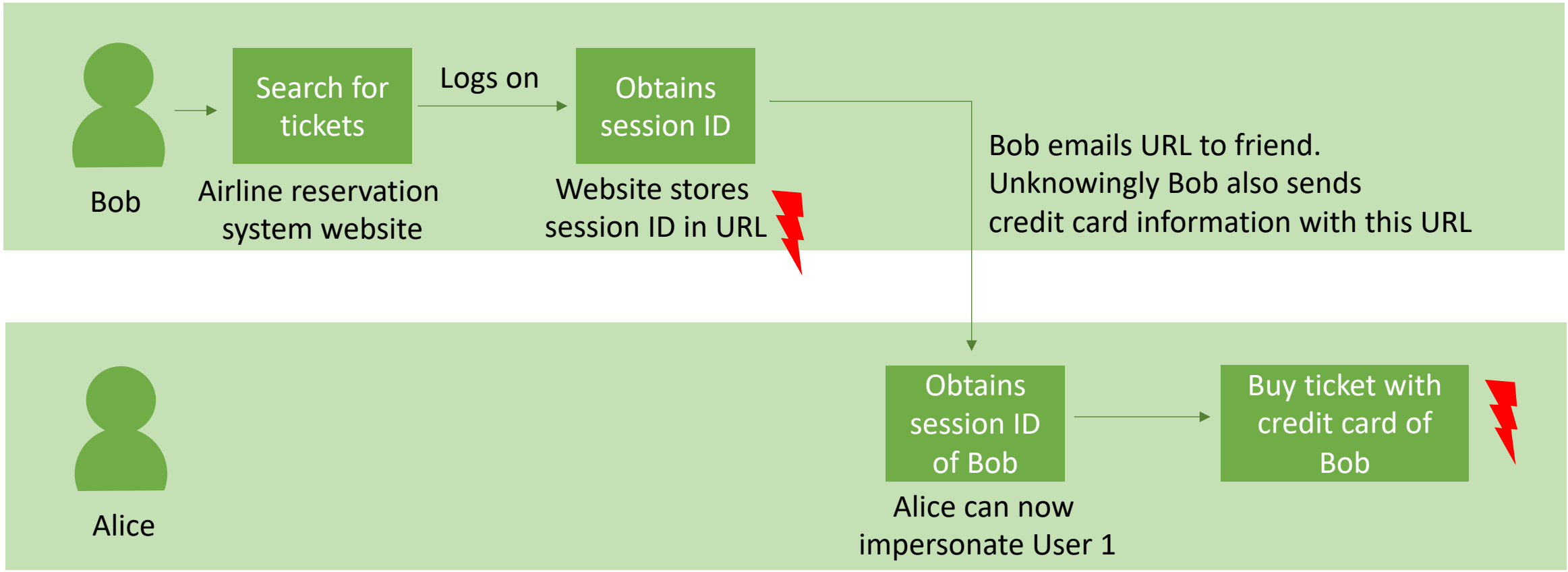
2. Broken Authentication and Session Management

- **What is it?** Incorrectly build auth. and session man. scheme that allows an attacker to impersonate another user
- **What is the impact?** Attacker can take identity of victim
- **How to prevent?** Don't develop your own authentication schemes.
  - Use open source frameworks that are actively maintained by the community.
  - Use strong passwords (incl. upper, lower, number, special characters)
  - Require current credential when sensitive information is requested or changed
  - Multi-factor authentication (e.g. sms, password, fingerprint, iris scan etc.)
  - Log out or expire session after X amount of time
  - Be careful with 'remember me' functionality

Huluka, D., & Popov, O. (2012, June). Root cause analysis of session management and broken authentication vulnerabilities. In *Internet Security (WorldCIS), 2012 World Congress on* (pp. 82-86). IEEE.

2. Broken Authentication and Session Management (example)

Example sessionID: http://airlinewebsite.com/sale/ticket?sessionid=12345&des=Amsterdam

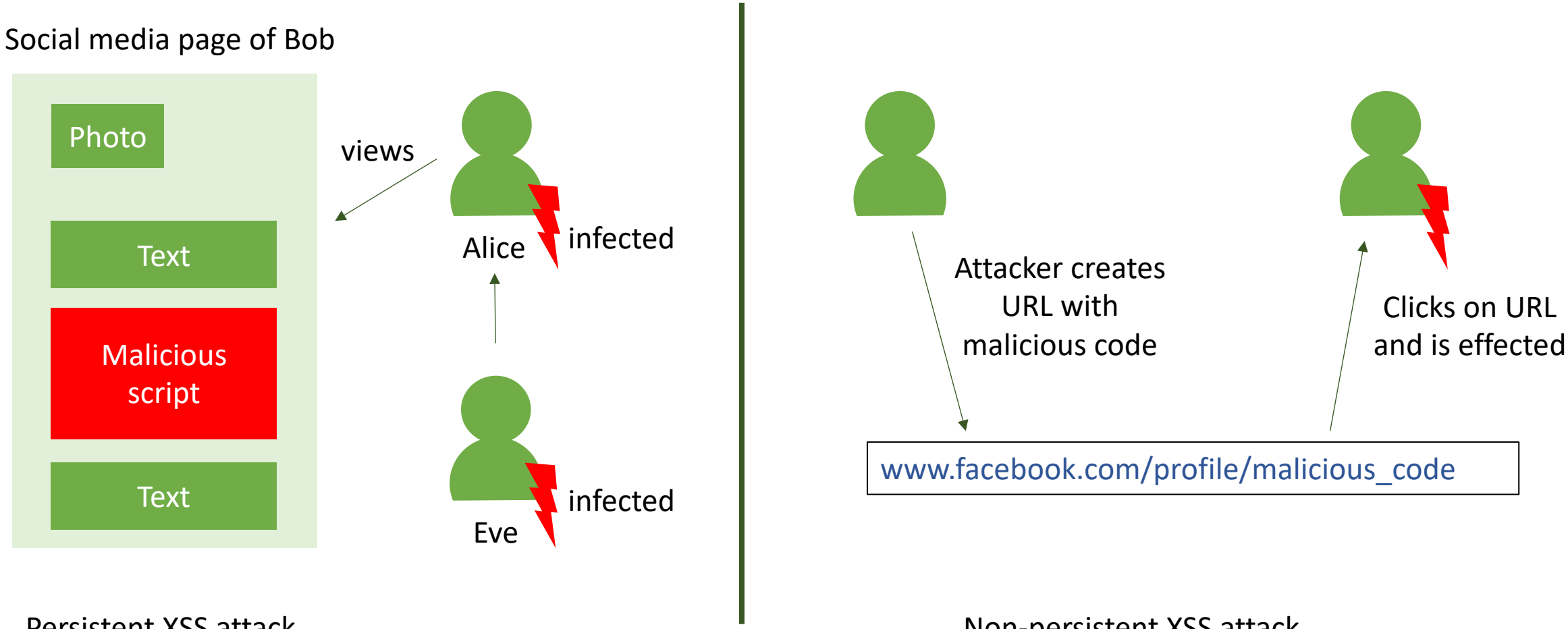


3. Cross-Site Scripting (XSS)

- **What is it?** Untrusted user input is interpreted by browser and executed
- **What is the impact?** Hijack user sessions, deface web sites, change content
- **How to prevent?**
  - Escape untrusted input data,
  - Latest UI framework

https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

3. Cross-Site Scripting (example)

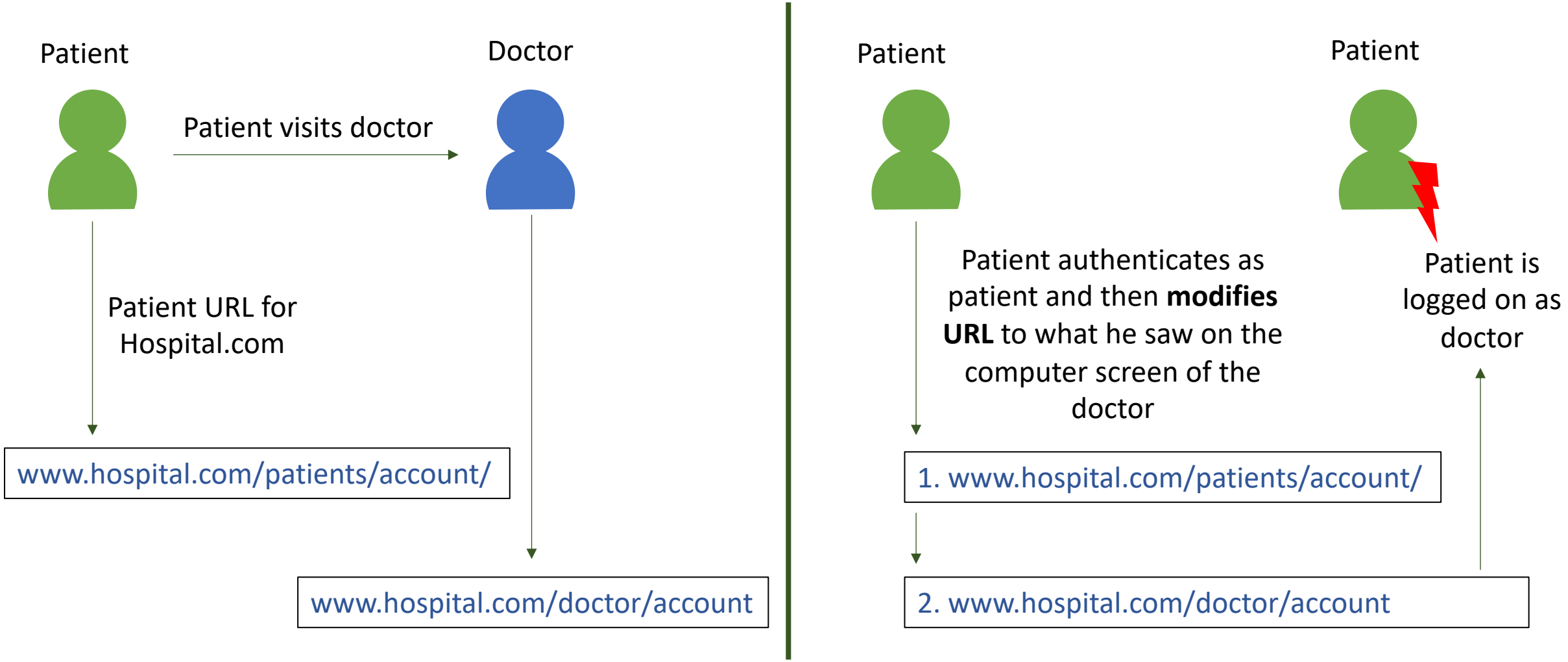


Source: Wassermann, G., & Su, Z. (2008, May). Static detection of cross-site scripting vulnerabilities. In *Proceedings of the 30th international conference on Software engineering* (pp. 171-180). ACM.

4. Broken Access Control

- **What is it?** Restrictions on what authenticated users are allowed to do are not properly enforced.
- **What is the impact?** Attackers can assess data, view sensitive files and modify data
- **How to prevent?**
  - Application should not solely rely on user input; check access rights on UI level and server level for requests to resources (e.g. data)
  - Deny access by default

4. Broken Access Control (example)



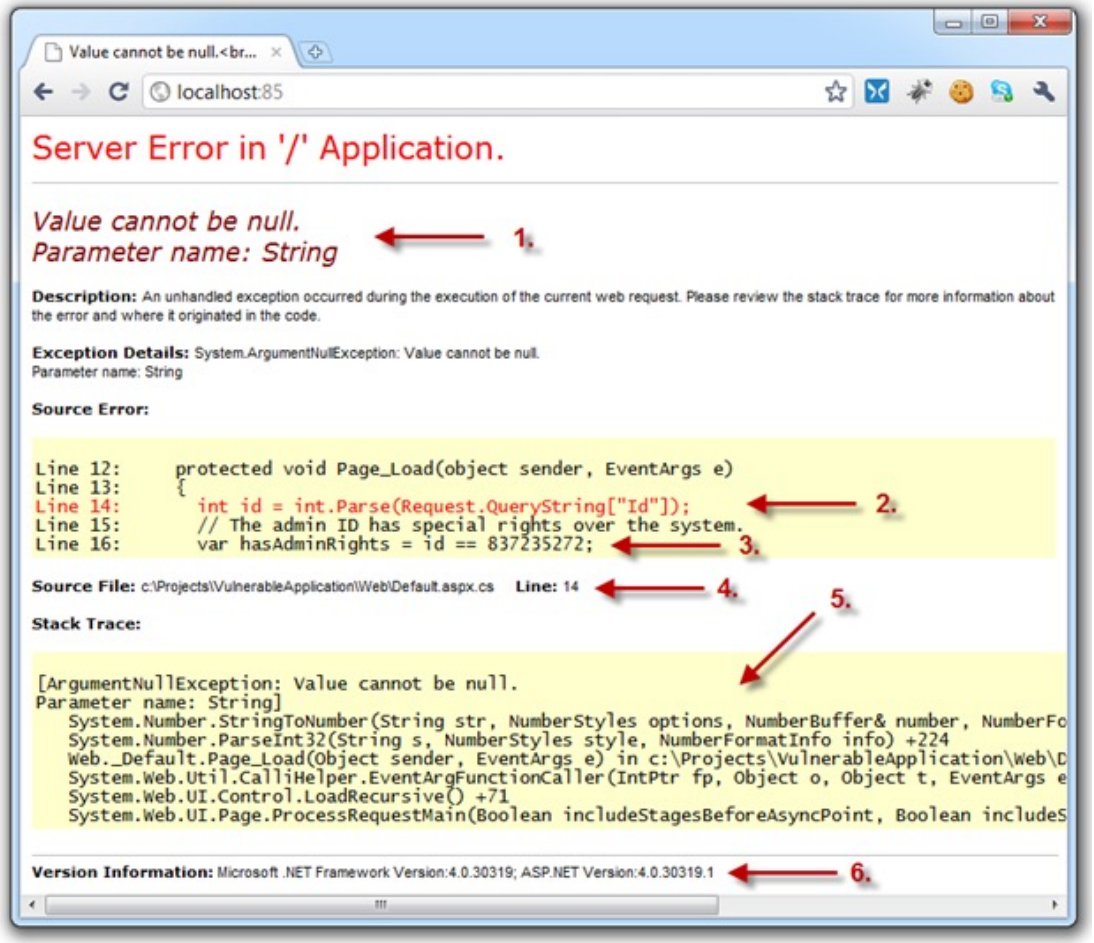
5. Security Misconfiguration

- **What is it?** Human mistake of misconfigurating the system (e.g. providing a user with a default password)
- **What is the impact?** Depends on the misconfiguration. Worst misconfiguration could result in loss of the system
- **How to prevent?**
  - Force change of default credentials
  - Least privilege: turn everything off by default (debugging, admin interface, etc.)
  - Static tools that scan code for default settings
  - Keep patching, updating and testing the system
  - Regularly audit system deployment in production

5. Security Misconfiguration (example)

- Catch exceptions;
  how elegant does the system fail?

1. The expected behaviour of a query string (something we normally don't want a user manipulating
2. The internal implementation of how a piece of untrusted data is handled (possible disclosure of weaknesses in the design)
3. Some very sensitive code structure details
4. The physical location of the file on the developers machine (further application structure disclosure)
5. Entire stack trace of the error (disclosure of internal events and methods)
6. Version of the .NET framework the app is executing on (discloses how the app may handle certain conditions)

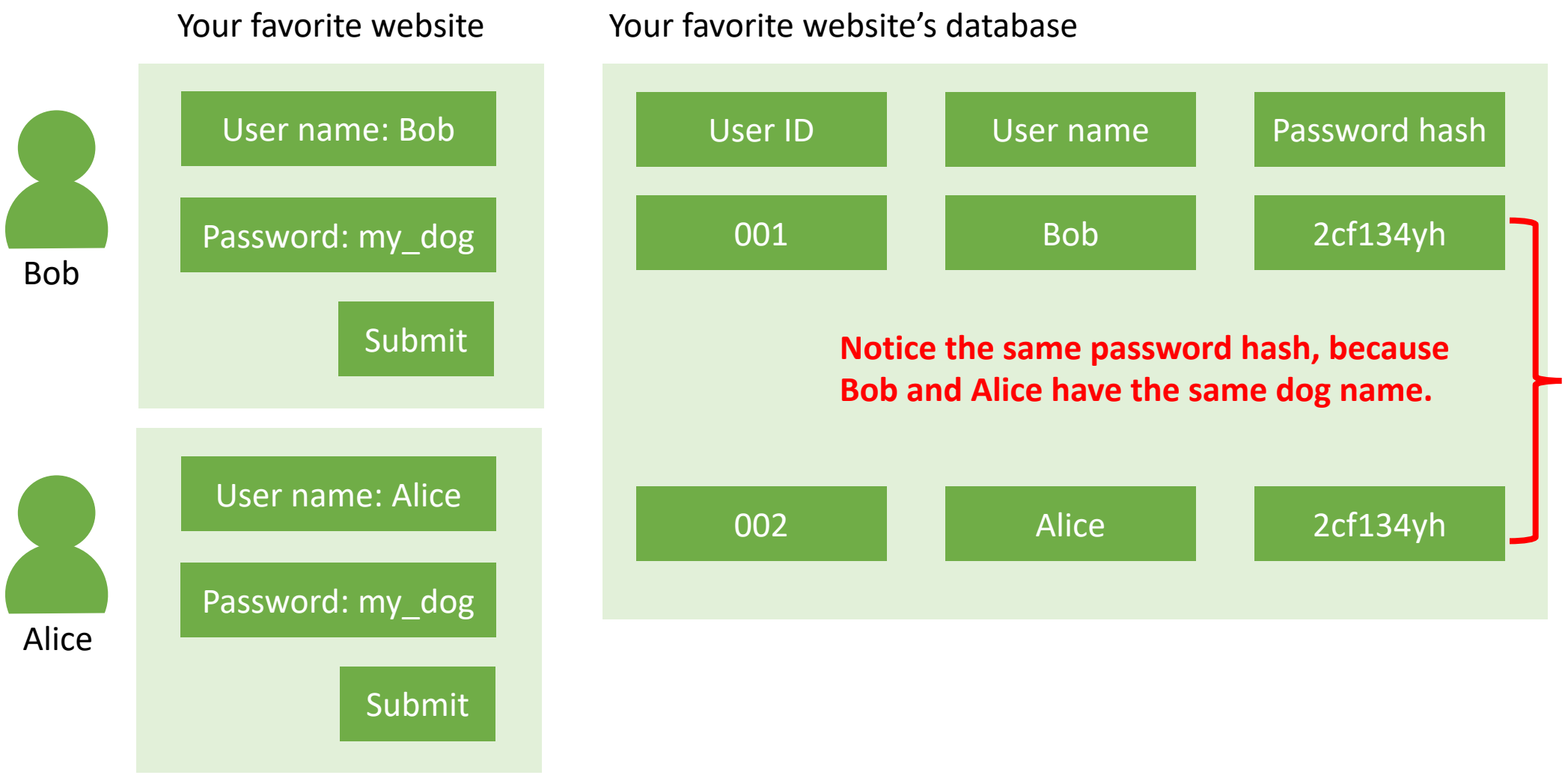


Source: https://www.troyhunt.com/owasp-top-10-for-net-developers-part-6/

6. Sensitive Data Exposure

- **What is it?** Sensitive data is exposed, e.g. social security numbers, passwords, health records
- **What is the impact?** Data that are lost, exposed or corrupted can have severe impact on business continuity
- **How to prevent?**
  - Always obscure data (credit card numbers are almost always obscured)
  - Update cryptographic algorithm (MD5, DES, SHA-0 and SHA-1 are insecure)
  - Use salted encryption on storage of passwords

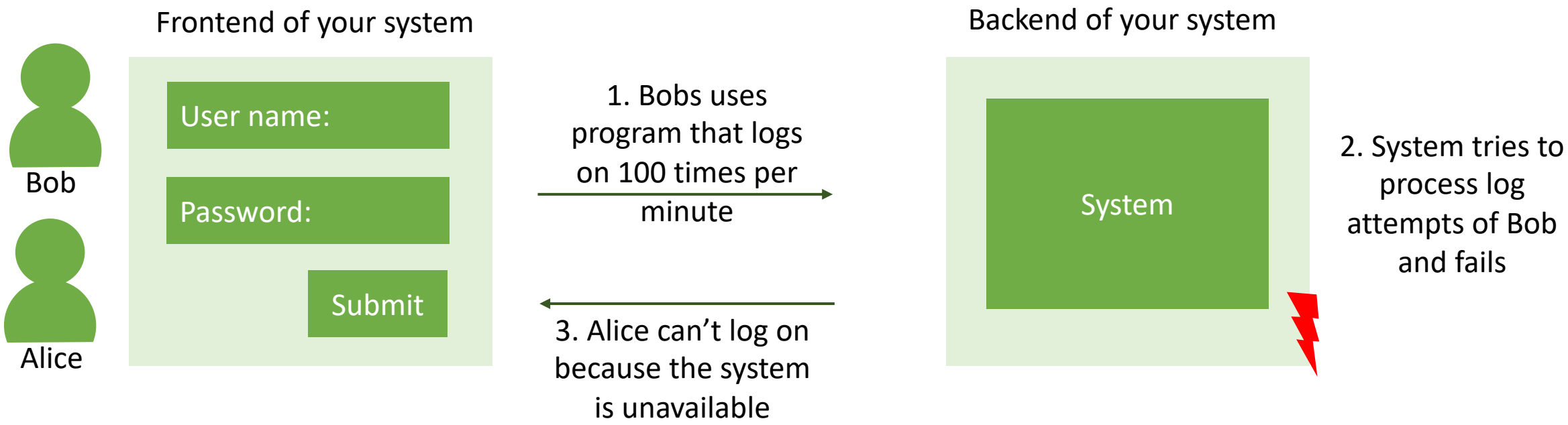
6. Sensitive Data Exposure (example)



7. Insufficient Attack Protection

- **What is it?** Applications that are attacked but do not recognize it as an attack, letting the attacker attack again and again
- **What is the impact?** Leak of data, decrease application availability
- **How to prevent?**
  - Detect and log normal and abnormal use of application
  - Respond by automatically blocking abnormal users or range of IP addresses
  - Patch abnormal use quickly

7. Insufficient Attack Protection (example)



8. Cross-site request forgery (CSRF)

- **What is it?** An attack that forces a victim to execute unwanted actions on a web application in which they're currently authenticated.
- **What is the impact?** Victim unknowingly executes transactions.
- **How to prevent?**
  - Reauthenticate for all critical actions (e.g. transfer money)
  - Include hidden token in request
  - Most web frameworks have built-in CSRF protection, but isn’t enabled by default!

Barth, A., Jackson, C., & Mitchell, J. C. (2008, October). Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security* (pp. 75-88). ACM.

8. Cross-site request forgery (example)

Legit URL: **http://yourbank.com/transferFunds?amount=1000&desAccount=7531948345**

The attacker modifies the legit URL and embeds it into another website that is open in your browser **on a different tab:**

Attack: ****

The attacker hides the attack behind an image (img) and as soon as the victim clicks on the image his funds will be transferred to **1112223333**

9. Using Components with Known Vulnerabilities

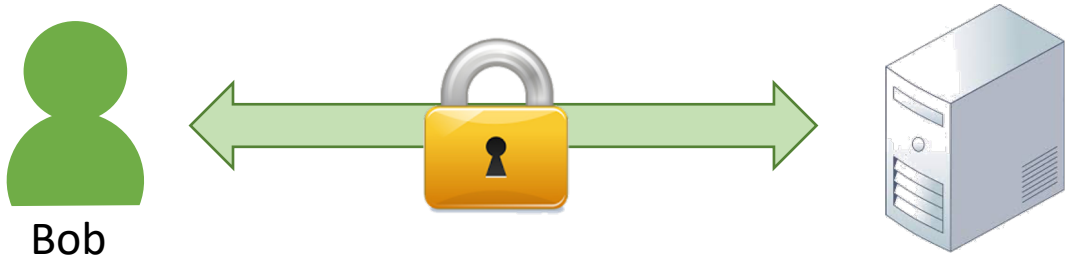
- **What is it?** Third-party components that the focal system uses (e.g. authentication frameworks)
- **What is the impact?** Depending on the vulnerability it could range from subtle to seriously bad
- **How to prevent?**
  - Always stay current with third-party components
  - If possible, follow best practice of virtual patching

9. Using Components with Known Vulnerabilities (example)

- Heartbeat OpenSSL/TLS

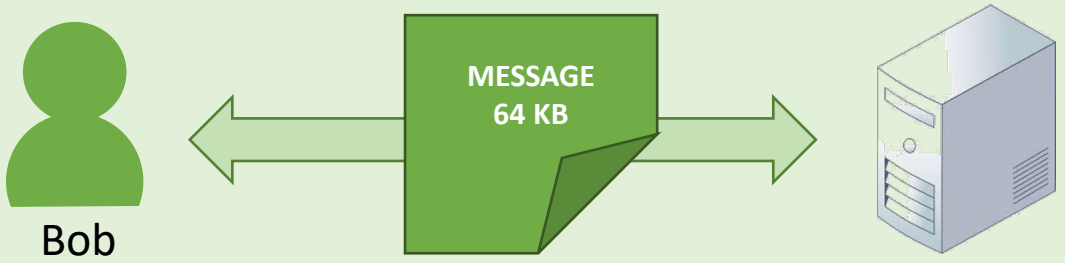
Connection between Bob and website is encrypted

Problem



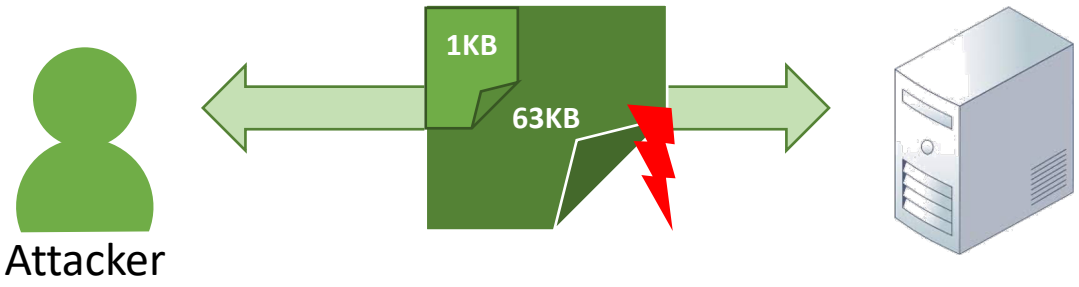
Keeping connection with the user through sockets takes resources.

Faulty Solution



To combat this, the web browser sends a heartbeat message to the server; please keep the connection open. This heartbeat message can contain anything < 65 KB.

Attack

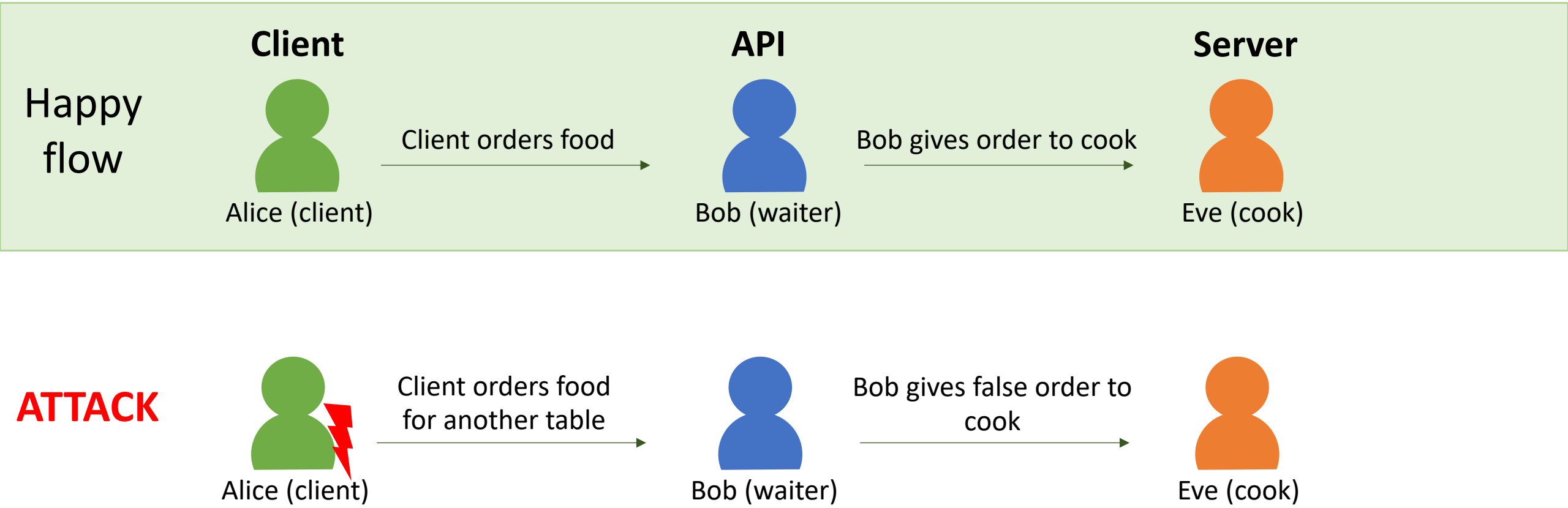


Attacker sends a 1 KB message to the server, but **tells the server the message is 64 KB big.** The server will return the 1 KB message **and ADD 63 KB out of memory!** Data from memory can be anything: e.g. username and passwords

10. Underprotected APIs

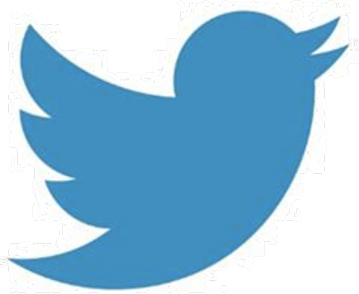
- **What is it?** Applications expose rich connectivity options through APIs, in the browser to a user. These APIs are often unprotected and contain numerous vulnerabilities.
- **What is the impact?** Data theft, corruption, unauthorized access, etc.
- **How to prevent?**
  - Ensure secure communication between client browser and server API
  - Reject untrusted/invalid input data
  - Use latest framework
  - Vulnerabilities are typically found by penetration testers and secure code reviewers

10. Underprotected APIs (example)



4. XML External Entities (XXE) NEW

- **What is it?** Many older or poorly configured XML processors evaluate external entity references within XML documents
- **What is the impact?** Extraction of data, remote code execution and denial of service attack
- **How to prevent?**
  - Use JSON, avoid avoiding serialization of sensitive data
  - Patch or upgrade all XML processors and libraries
  - Disable XXE and implement whitelisting
  - Detect, resolve and verify XXE with static application security testing tools



4. XML External Entities (XXE) (example)

Happy flow

Client

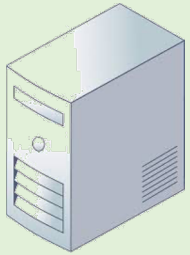
Alice (client)

Sends message

XML parser

Instruction send to server

Server



ATTACK

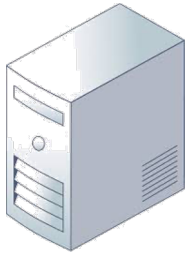
Billion laughs

Alice (client)

Send malicious message with an infinite ‘lol’ loop

XML parser

Instruction send to server



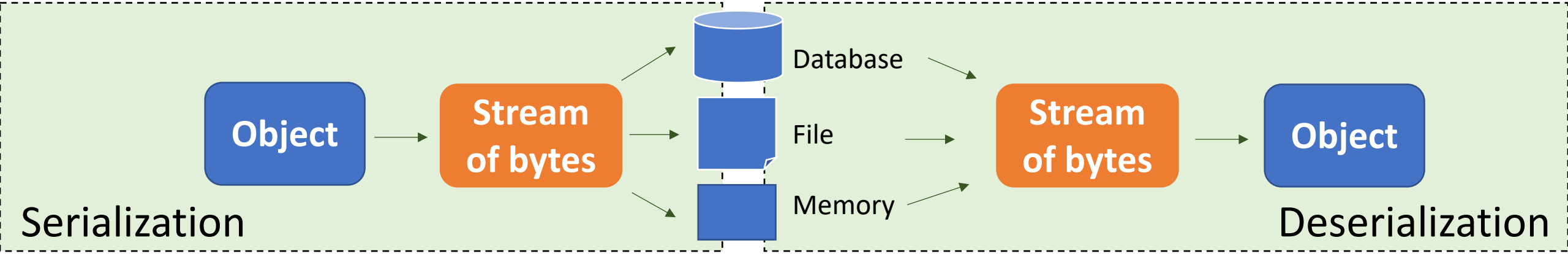
Result: server is down

8. Insecure deserialization NEW

- **What is it?** Error in translations between objects
- **What is the impact?** Remote code execution, denial of service. Impact depends on type of data on that server
- **How to prevent?**
  - Validate user input
  - Implement digital signatures on serialized objects to enforce integrity
  - Restrict usage and monitor deserialization and log exceptions and failures

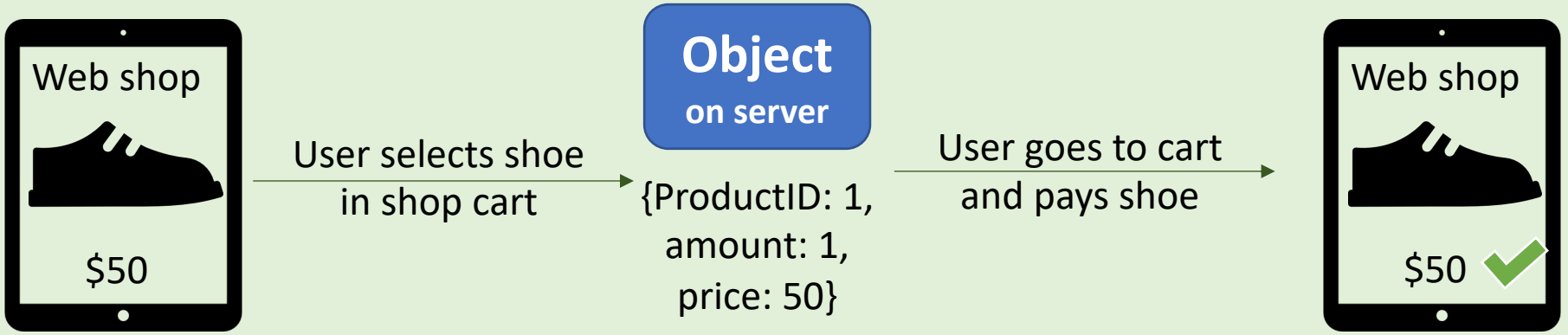
8. Insecure deserialization (explanation)

What is (de)serialization?

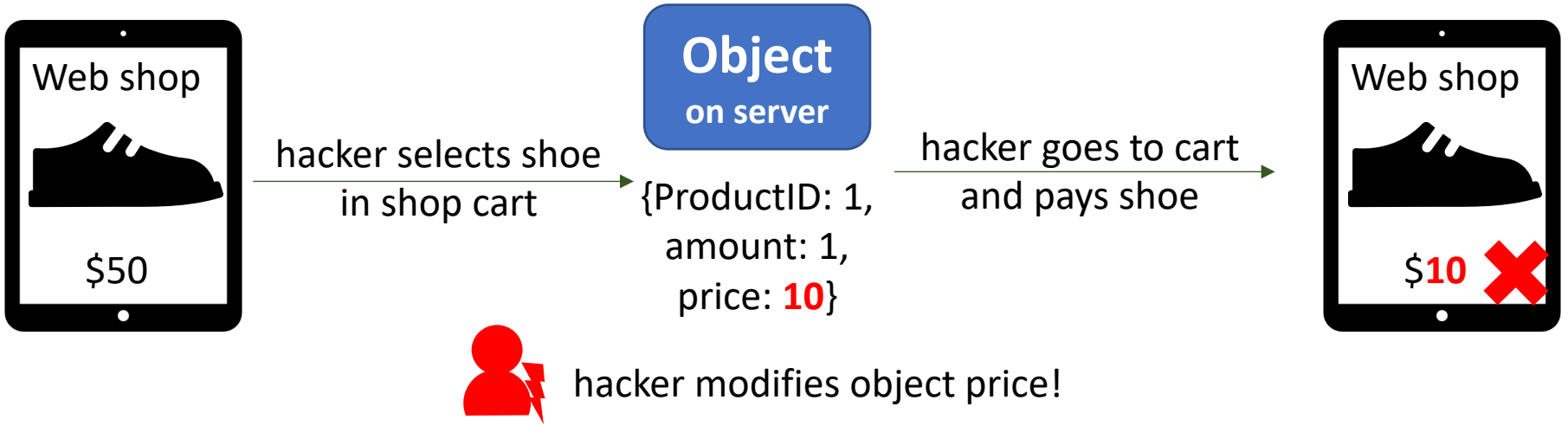


8. Insecure deserialization (example)

Happy flow



ATTACK



10. Insufficient logging & monitoring NEW

- **What is it?** Not able to witness or discover an attack when it happens or happened
- **What is the impact?** Allows attacker to persist and tamper, extract, or destroy your data without you noticing it
- **How to prevent?**
  - Log login, access control and server-side input validation failures
  - Ensure logs can be consumed easily, but cannot be tampered with
  - Continuously improve monitoring and alerting process
  - Mitigate impact of *breach*: Rotate, Repave and Repair

10. Insufficient logging & monitoring (example)

Happy flow

Client

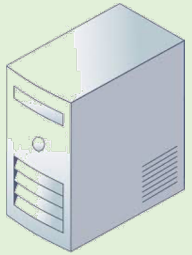
Alice (client)

Login

Web app

Instruction send to server

Server



Result: Meta data collected: occurrences, timestamp, location, etc.

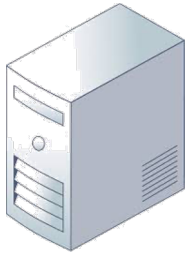
ATTACK

Alice (client)

Tries to log on using brute force

Web app

Instructions send to server



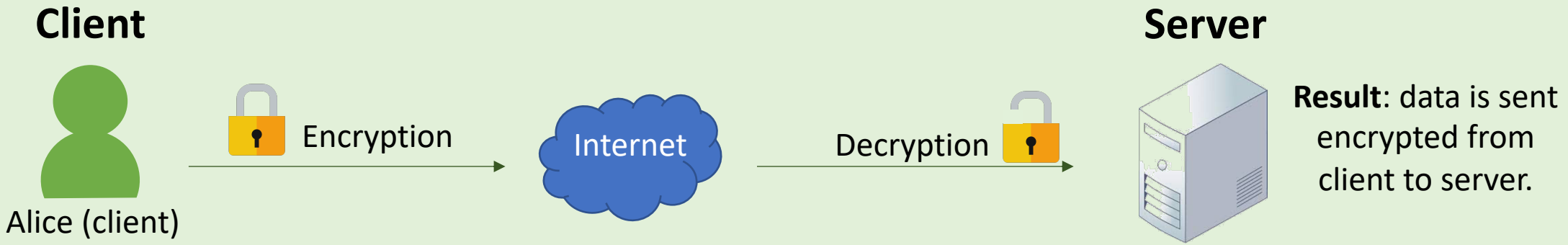
Result: Server down. *No data (meta) collected of Alice and Alice's activity*

2. Cryptographic Failures NEW

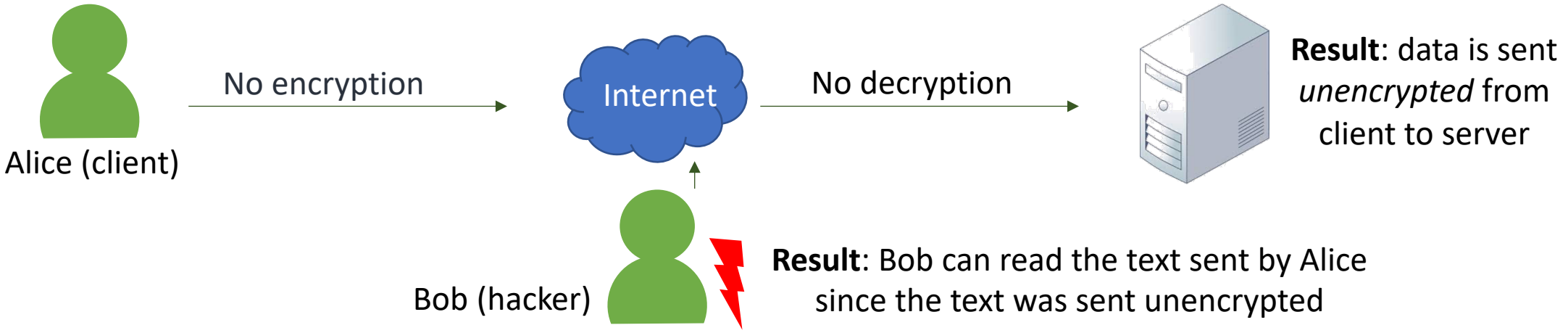
- **What is it?** Ineffective execution & configuration of cryptography (e.g. FTP, HTTP, MD5, WEP)
- **What is the impact?** Sensitive data exposure
- **How to prevent?**
  - Never roll your own crypto! Use well-known open source libraries
  - Static code analysis tools can discover this issue.
  - Key management (creation, destruction, distribution, storage and use)

2. Cryptographic Failures (example)

Happy flow



Attack



4. Insecure design NEW

- **What is it?** A failure to use security by design methods/principles resulting in a weak or insecure design
- **What is the impact?** Breach of confidentiality, integrity and availability
- **How to prevent?**
  - Secure lifecycle (embed security in each phase; requirements, design, development, test, deployment, maintenance and decommissioning)
  - Use manual (e.g. code review, threat modelling) and automated (e.g. SAST and DAST) methods to improve security

Meltdown

Spectre

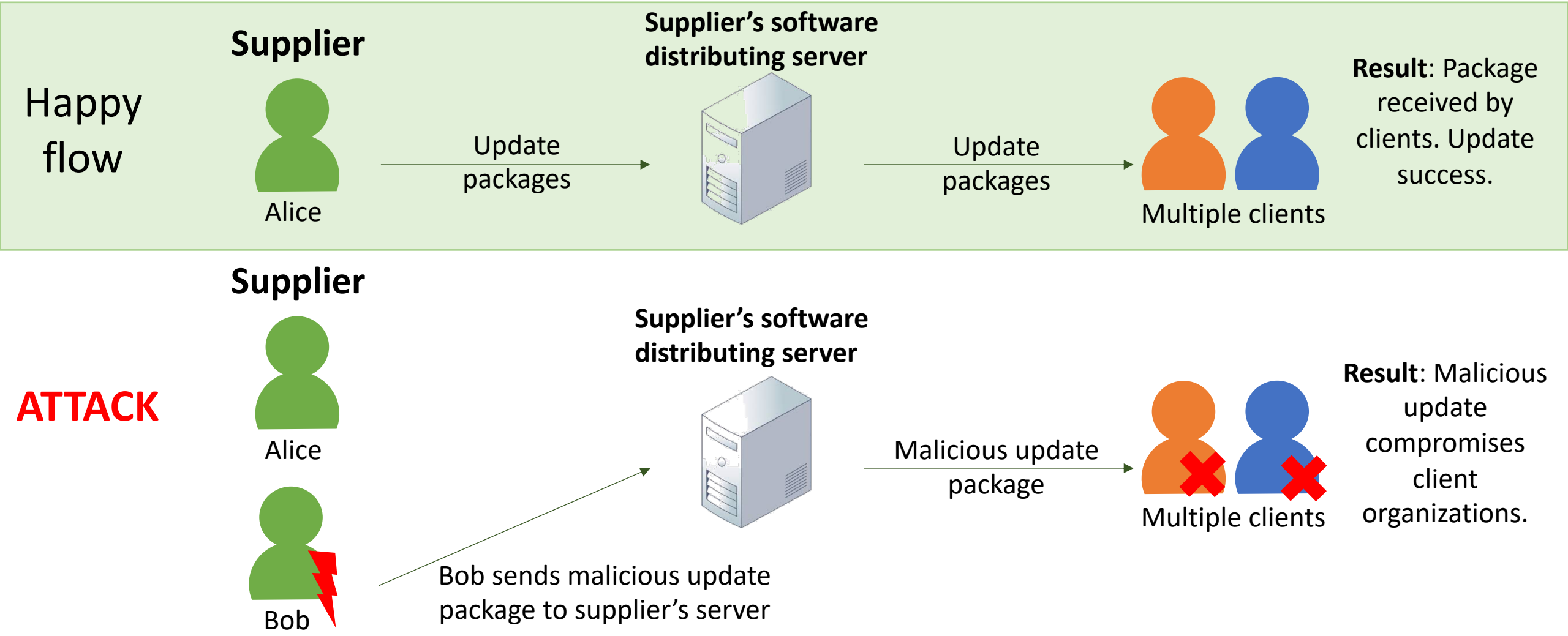
4. Insecure design (example)

8. Software and Data Integrity Failures NEW

- **What is it?** E.g. an application that relies on updates from a trusted external source, however the update mechanism is compromised
- **What is the impact?** Supply chain attack; data exfiltration, ransomware, etc.
- **How to prevent?**
  - Verify input (in this case software updates with digital signatures)
  - Continuously check for vulnerabilities in dependencies
  - Use Software Bill of materials
  - Unconnected back ups

solarwinds

8. Software and Data Integrity Failures (example)



10. Server-Side Request Forgery NEW

- **What is it?** Misuse of prior established trust to access other resources. A web application is fetching a remote resource without validating the user-supplied URL.
- **What is the impact?** Scan and connect to internal services. In some cases the attacker could access sensitive data
- **How to prevent?**
  - Sanitize and validate all client-supplied input data
  - Segment remote server access functionality in separate networks to reduce the impact
  - Limiting connections to specific ports only (e.g. 443 for https)

10. Server-Side Request Forgery (example)

Happy flow

Client

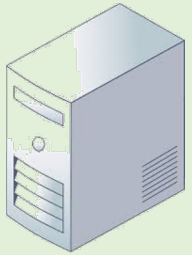
Alice (client)

Requests data

Web server

Sends request to internal server

DB server



Result: web server is trusted by internal database (db) server and it executes the request

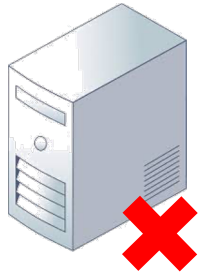
ATTACK

Bob (client)

Sends malformed request to impersonate the web server

Web server

Sends request to internal server



Result: web server is trusted by internal database server and it executes the malicious request

';--have i been pwned?

Check if you have an account that has been compromised in a data breach

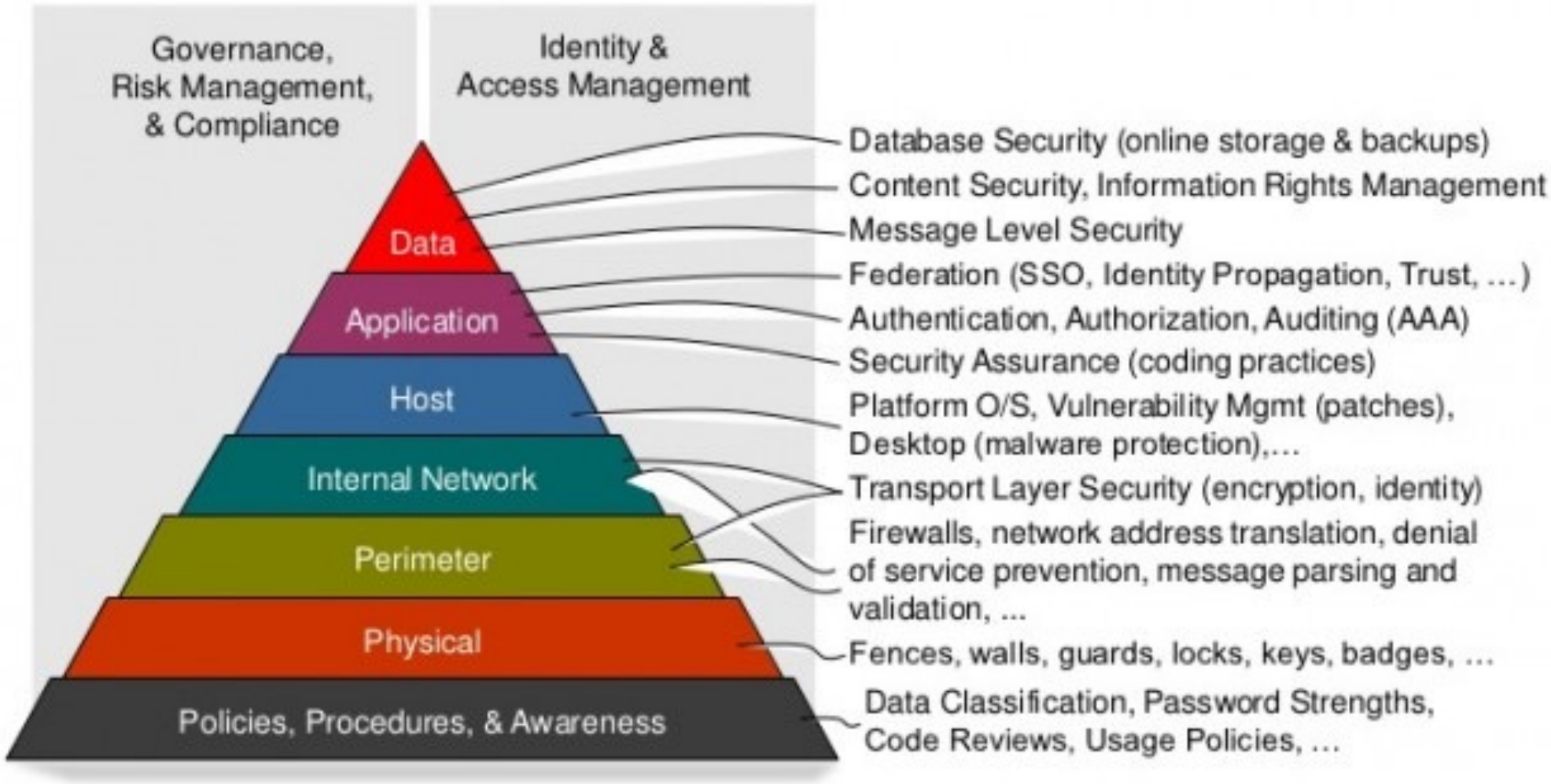
email address pwned?

431	9,543,096,417	109,651	133,271,802
pwned websites	pwned accounts	pastes	paste accounts

Bonus

A. Defense in depth
B. STRIDE (basics)
C. Secure development processes
D. FAQ

A. Defense in depth



Source: Oracle

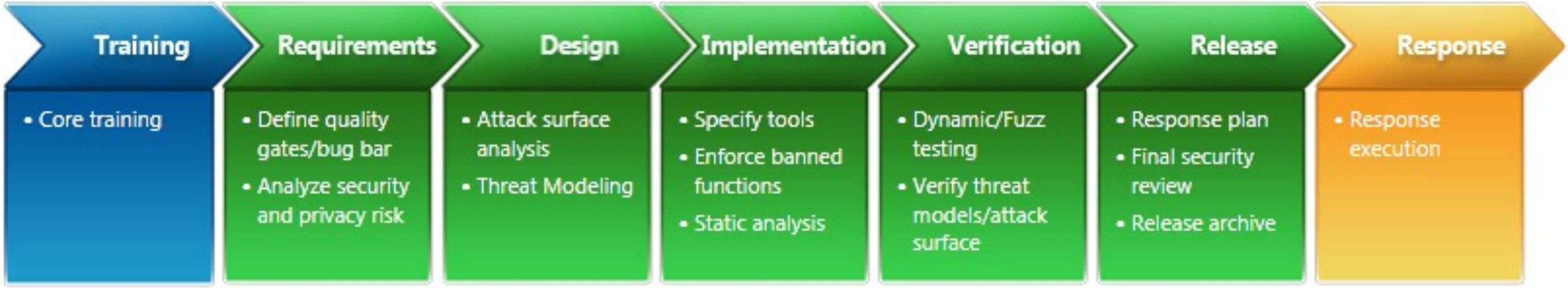
B. STRIDE - basics

- Why?
  - Examine what can go wrong
  - What are you going to do about it
  - Determine whether you doing a good job

- STRIDE
  - Spoofing
  - Tampering
  - Repudiation
  - Information disclosure
  - Denial of service
  - Elevation of privilege

C. Secure development processes

Microsoft Security Development Lifecycle (MS SDL)



Other secure development processes are:

- Software Assurance Maturity Model (previous called CLASP)
- Touchpoints for software security

FAQ - How can you test whether your website uses the latest security protocols?

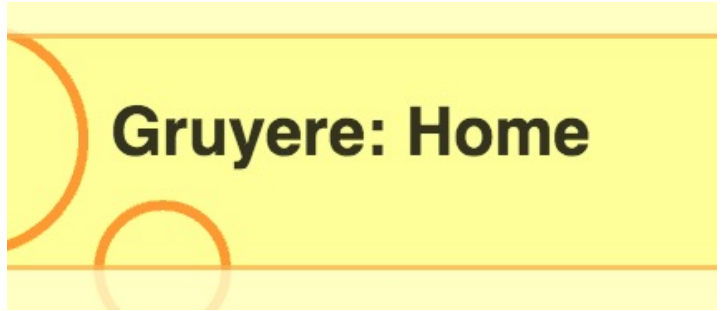
- Navigate to ssllabs.com to test the security protocols of your website for free!

yourresearchtutor@gmail.com

FAQ - Where can I (legally) test my hacking skills for free?

There are several websites specifically for this need, for free!

- http://google-gruyere.appspot.com/



yourresearchtutor@gmail.com

FAQ. What are *Insecure Direct Object References?*

- **What is it?** A reference to a file, database or directory exposed to user via the browser
- **What is the impact?** Any user can navigate to almost any part of the system and attack the system by modifying the URL through the browser
- **How to prevent?**
  - Check access rights (e.g. proper authorization)
  - Input validation

FAQ. What are *Insecure Direct Object References?*

Example of direct reference: http://airlinewebsite.com/file.jsp?file=report.txt

The attacker modifies the URL to the root folder:

Example of direct reference: http://airlinewebsite.com/file.jsp?file=**../../../root