cphbusiness

COPENHAGEN BUSINESS ACADEMY





Security, Fall 2018

Outline

- Follow up on Honey pot and firewall
- SQL injection
  - Exercise
- Prepared statements
- Cross side scripting XSS

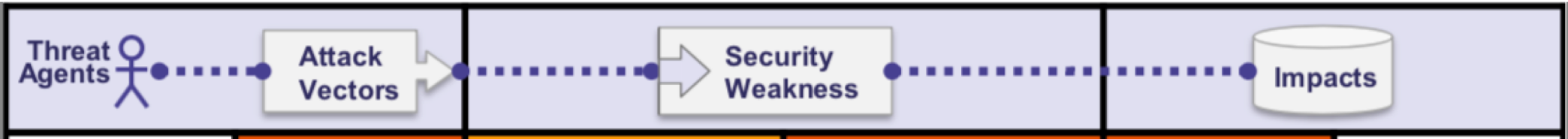
OWASP top 10, 2013 & 2017

OWASP Top 10 - 2013		OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Detection

A1:2017-Injection

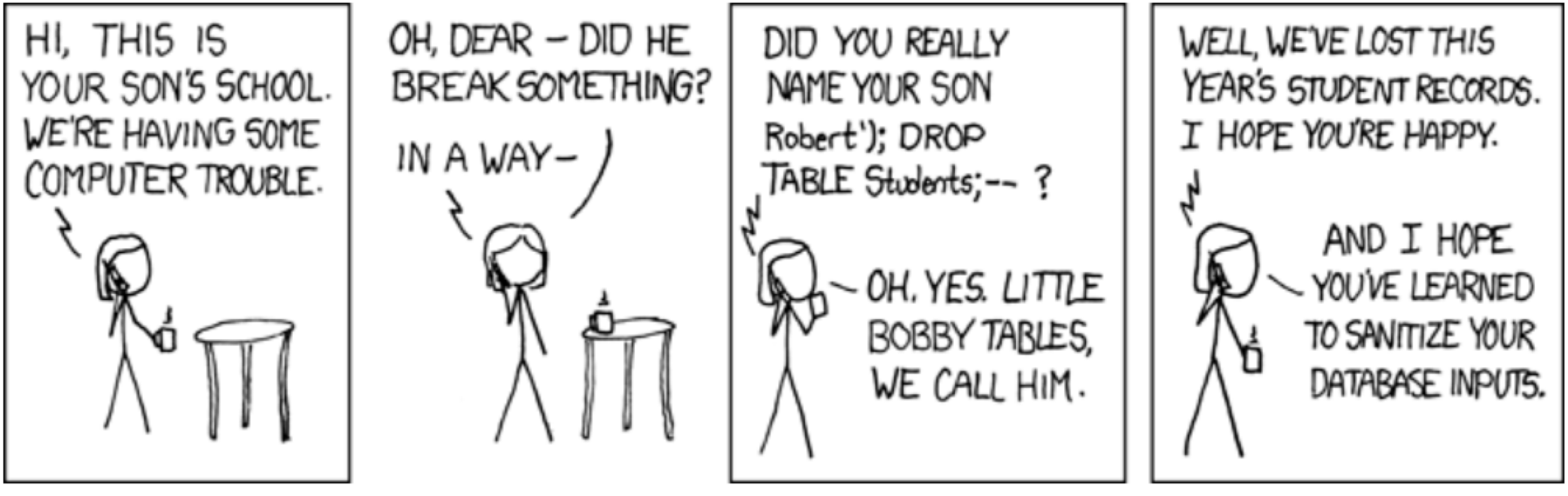
(See page 7 of OWASP 2017)



App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.		Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.	

SQL injection

- SQL Injection Based on 1=1 is Always True
  - `userId = "7 or 1 = 1"`
- SQL Injection Based on ""="" is Always True
  - `username = "Kurt' or '' = ‘”`
- SQL Injection Based on Batched SQL Statements
  - `userId = "7; DROP TABLE Students; --”`



How to find db object names

- SQL Exceptions
- Guessing
- Insider knowledge

How to prevent SQL injection

- Manual formatting (not good)
- Escaping
- Parametrized queries
  - Prepared statements
  - Works for static queries
- Placeholders
  - For dynamic queries
  - For arrays
  - Whitelisting

Placeholders

- identifier placeholder (single identifier)
- identifier list (comma separated identifiers)
- integer list (comma separated integers)
- strings list (comma separated strings)
- the special SET type consists of a comma-separated idientifier=string value pairs
- you name it

Why manual formatting is bad

- Manual formatting can be incomplete
- Manual formatting can be applied to the wrong literal
- Manual formatting is essentially a non-obligatory measure
- Manual formatting can be separated from the actual query execution by a considerable distance

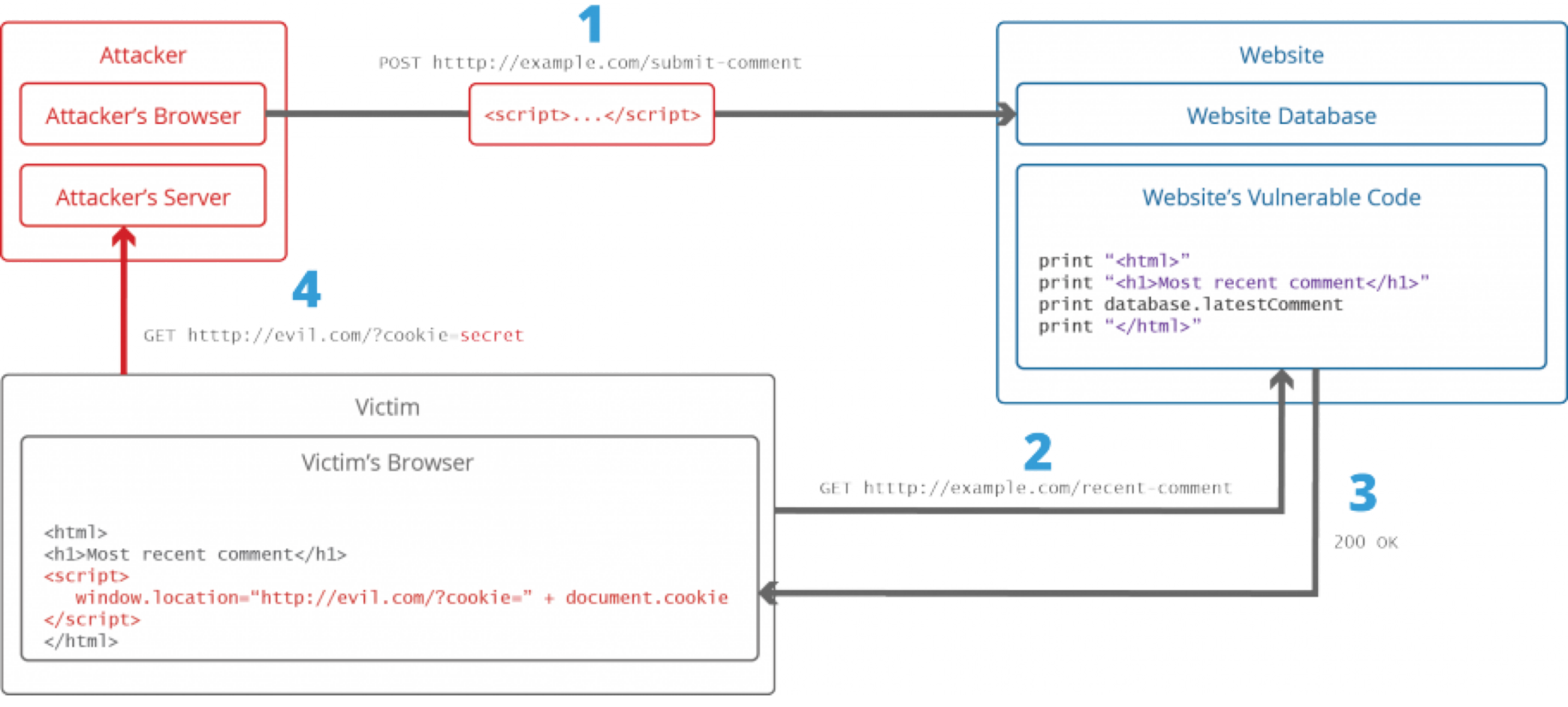
Prepared Statements

1. Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters ("?"). Example: `INSERT INTO MyGuests VALUES(?, ?, ?)`
2. The database parses, compiles, and performs query optimization on the SQL statement template. Stores the result without executing it
3. Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

- Can be emulated for certain dbs.

From w3schools.com

Cross Site Scripting



cphbusiness

XSS consequenses (it's only JS?)

- Malicious JavaScript has access to objects, including access to cookies. If an attacker can obtain a user's session cookie, they can impersonate that user.
- JavaScript can read and make modifications to the DOM.
- JavaScript can use XMLHttpRequest to send HTTP requests to arbitrary destinations.
- JavaScript in modern browsers can leverage HTML5 APIs such as accessing a user's geolocation, webcam, microphone and even the specific files from the user's file system. While most of these APIs require user opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.