Introduction to Communication Technology and Digital Security

Digital Security – Week 3

Exploitation

Basel Katt

Expectations and Learning Outcome

- By the end of the lecture and lab you will
    - have a basic understanding of the exploitation phase in penetration testing and how to exploit some basic vulnerabilities,
    - be able to use security tools to perform exploitation

Vulnerability Databases and Standards

- NVD (National Vulnerability Database): is a repository of standards based vulnerability management data represented
- SCAP (Security Content Automation Protocol): consists of a suite of specifications for standardizing the format by which software flaw and security configuration information is communicated, both to machines and humans.
- OVAL (Open Vulnerability and Assessment Language): a language for representing system configuration information, assessing machine state, and reporting assessment results

Vulnerability Databases and Standards

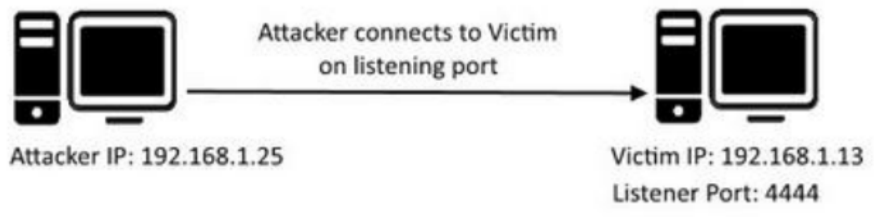
- CWE (Common Weakness Enumeration): is a formal list of common software weaknesses that can occur in software's architecture, design, code or implementation that can lead to exploitable security vulnerabilities
- CVE (Common Vulnerability and Exposure): is a dictionary of public known security vulnerabilities.
  - Each vulnerability has an identifier, CVE ID, and a description
  - CVE's common identifiers enable data exchange and provide a baseline index point for evaluating coverage of tools and services
- CVSS (Common Vulnerability Scoring System), a standard that aims at estimating the severity of a vulnerability

Exploitation

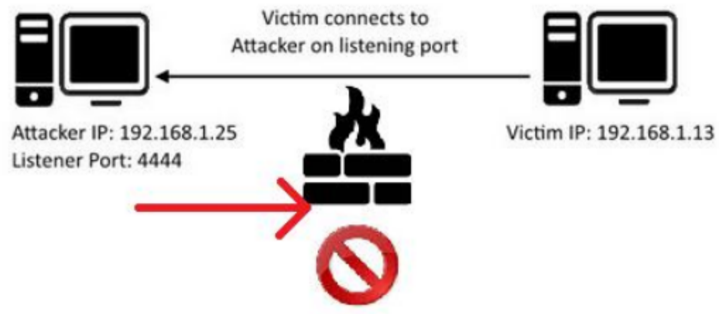
- **Exploitation** is the process of bypassing a security flaw or circumvent security control, e.g., cracking passwords
- It might lead to taking control over the system
- Exploitation is the process of launching an **exploit**, which is the realization of a vulnerability
- An exploit utilizes a vulnerability to execute a **payload**
- A **payload** is the additional functionality or a change in behavior that the pentester wants to accomplish on the target machine
  - bind and reverse SHELL
  - install new software
  - disable running services
  - add new users
  - open backdoors

SHELL

- **Bind shells**


Attacker connects to Victim on listening port
Attacker IP: 192.168.1.25
Victim IP: 192.168.1.13
Listener Port: 4444

- **Reverse shells: 80, 8080, 443**


Victim connects to Attacker on listening port
Attacker IP: 192.168.1.25
Listener Port: 4444
Victim IP: 192.168.1.13

https://irichmore.wordpress.com/2015/06/04/bind-shell-vs-reverse-shell/

How to start compromising a service?

- What kind of services do we have to face from outside?
  - Web, Ftp, ssh, dns, mail (SMTP, POP3, IMAP, Exchange), VPN and many others
- What kind of vulnerabilities can we expect?
  - Configuration related
    - Default credentials, or easy to guess credentials
    - No or inappropriate protection against guessing
    - Unnecessary function
  - SW vulnerability related
    - No input validation
    - Memory handling errors
    - Synchronization related errors,
    - … and many others

How to start compromising a service?

- First use in the normal way
  - Is there any information disclosure?
  - Error messages, etc.
  - Restrictions
- Then, force it to error and obtain information
  - Provide invalid data (fuzzing)
    - Use it in an invalid way
- Try factory defaults
- Brute-forcing
- Search for known exploits
- Service specific exploitations
- Unique ways

Metasploit

- Metsploit is a framework for launching and developing exploits
- Basic functions
  - **Search**: search for all related exploits in MSF’s database based on the CVE identifiers reported in the Nessus results.
  - **Use**: select the exploit that best matches the CVE identifier
  - **Show Payoads**: review the available payloads for the selected exploit
  - **Set Payload**: select the desired payload for the selected exploit
  - **Show Options**: review the necessary options that must be set
  - **Set Options**: assign value to ALL of the necessary options that must be present
  - **Exploit**

Web Exploitation

Web Application Pentesting

- Issues to take into account when performing web application pentesting
  - Network, platform and application architectures
  - Authentication and authorization functions applied
  - Session management and maintenance
  - Server application main services and application functions
  - Client-side capabilities and ways to communicate and interact with server

Web Application Pentesting

- Some activities specific to web application scanning
  - Identify all hosted sites that run on the server (DNS mapping)
  - Check if there exist any web application firewall in place (waf), or load balancer
  - Check the directory structure and files of the web application
  - Enumerate the attack surface and all possible interfaces that accepts input from users
  - Error handling capabilities, and investigate error messages returned

HTTP

- HTTP is a protocol for web communication using client-server model (request – response)
  - Requests and responses consists of a *header* and a *body*
    - Header includes version, web method, hostname, date, content type, etc.
    - Some of the main web methods are: GET, POST, DELETE, ...etc
- HTTP protocol is fundamentally stateless?
  - however, do web servers need to maintain a state?
- It works by sending the state to the client, then it returns it in subsequent requests
- It uses hidden fields or cookies
  - Hidden is an input type that defines a fields that is not visible to a user
  - We don’t want to pass hidden fields all the time, hard to maintain on all pages, and cannot survive closing the browser

Cookies

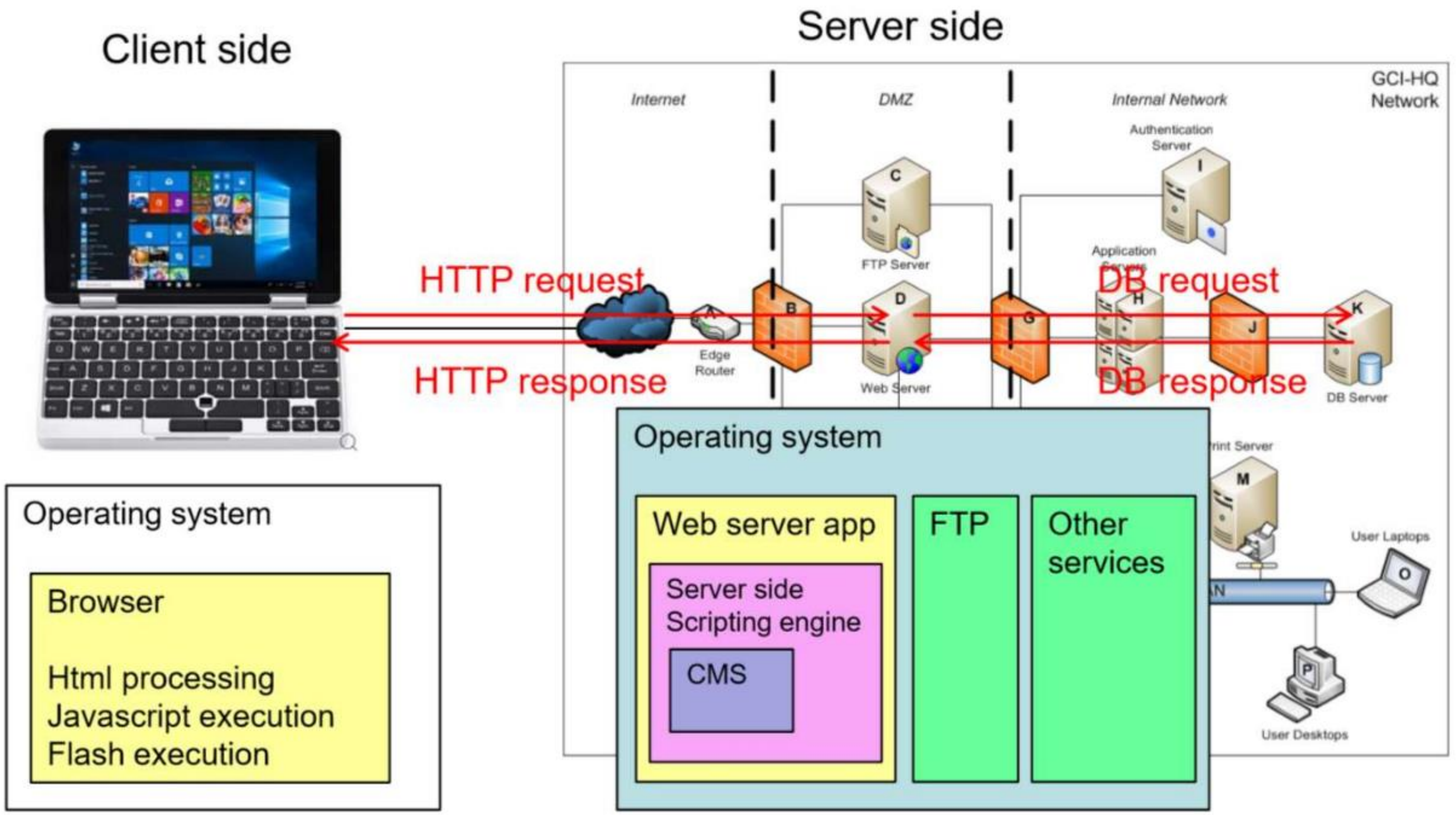
- A small plain text file that is stored by the client and maintain non-executable code
- A cookie consists of name-value pairs
- Servers instruct the clients to store these cookies on their machines
  - by sending an HTTP header set-Cookie
  - Set-Cookie specifies a value and a set of options
- A client sends back the stored cookie value with each subsequent request based on some rules
- A server uses cookies to identify individual users
- Servers that require login typically set a cookie once you are authenticated

```
Set-Cookie: edition=no; domain=.amazon.com; path=/
            expires=Tue, 20-Sep-2016 14:20:16 GMT;
```

Cookies

- Identifier: the cookie serves as an identifier for the client
- Authentication: after the first time a user is authentication, subsequent requests are authenticated provided the cookie
- Personalization: remember specific information about users, e.g., language selection, layout etc.
- Third-party cookies: used by ad networks. Ad-web-site stores cookies on your computer and save information about you that can be used later when you visit third web sites

Accessing a Webpage

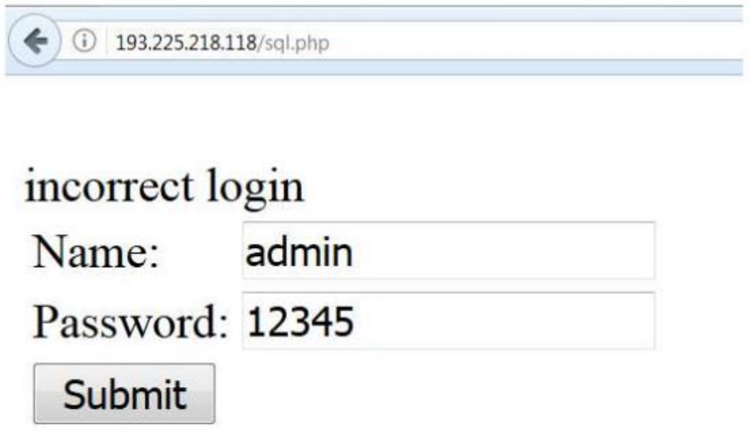


SQL Injection

- Server-side data: Long lived state stored in a separate database - need to protect these data
- Common way is to store in database and query using SQL (standard query language)
- Use SQL to read and write to the DB (SELECT UPDATE INSERT DROP)
- Server-side code interact with the DB using SQL, and usually uses other language like PHP

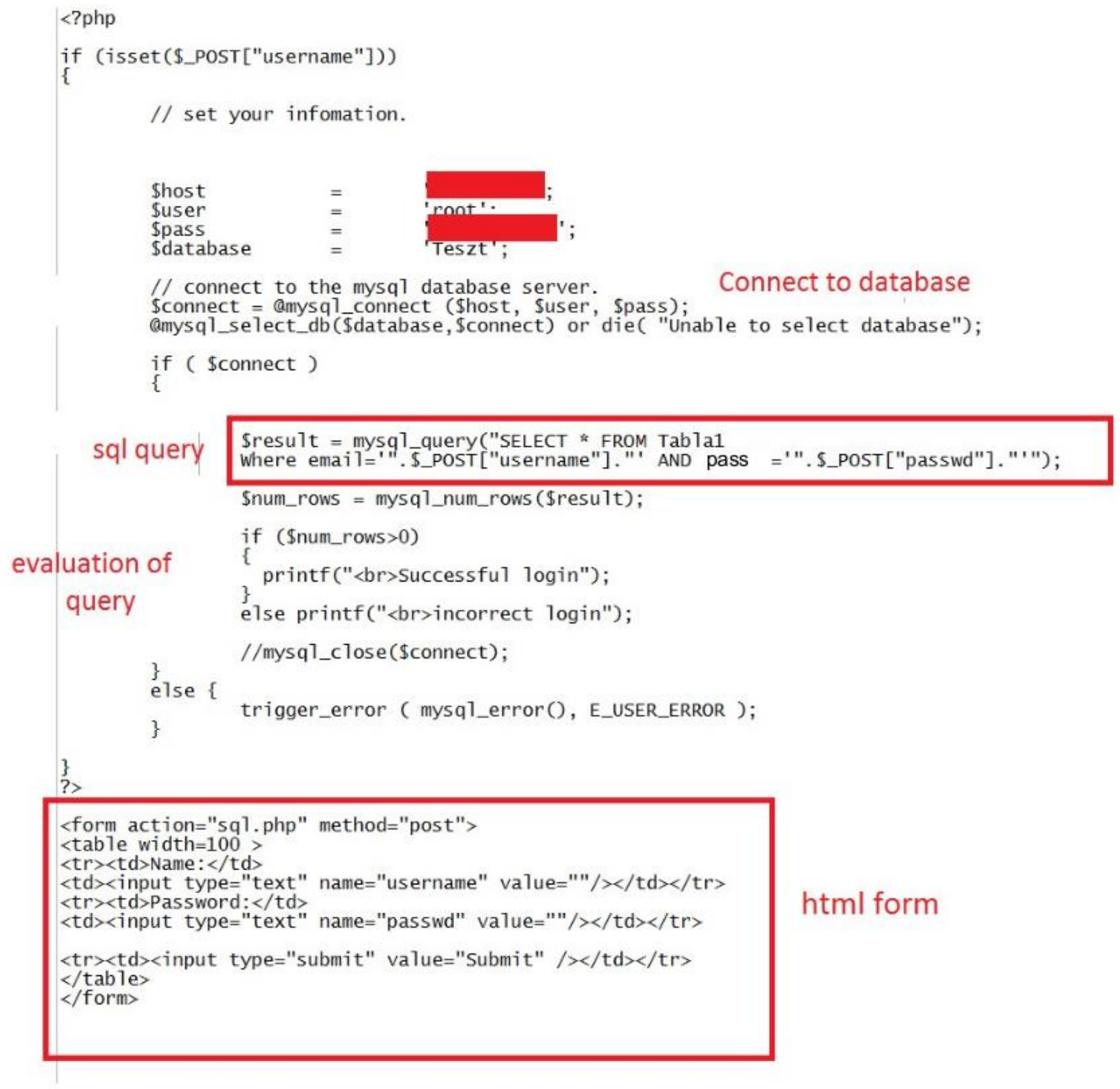
SQL with PHP example

Php uses the *mysql_connect, mysql_select_db, mysql_query, mysql_num_rows mysql_fetch_array* Etc. commands



```
193.225.218.118/sql.php

incorrect login
Name:      admin
Password:  12345
Submit
```



```
<?php

if (isset($_POST["username"]))
{

        // set your infomation.


        $host           =       [redacted];
        $user           =       'root';
        $pass           =       [redacted]';
        $database       =       'Teszt';

        // connect to the mysql database server.
        $connect = @mysql_connect ($host, $user, $pass);
        @mysql_select_db($database,$connect) or die( "Unable to select database");

        if ( $connect )
        {

                $result = mysql_query("SELECT * FROM Tabla1
                Where email='".$_POST["username"]."' AND pass  ='".$_POST["passwd"]."'");

                $num_rows = mysql_num_rows($result);

                if ($num_rows>0)
                {
                  printf("<br>Successful login");
                }
                else printf("<br>incorrect login");

                //mysql_close($connect);

        }
        else {
                trigger_error ( mysql_error(), E_USER_ERROR );
        }

}
?>

<form action="sql.php" method="post">
<table width=100 >
<tr><td>Name:</td>
<td><input type="text" name="username" value=""/></td></tr>
<tr><td>Password:</td>
<td><input type="text" name="passwd" value=""/></td></tr>

<tr><td><input type="submit" value="Submit" /></td></tr>
</table>
</form>
```

SQL Injection Attacks

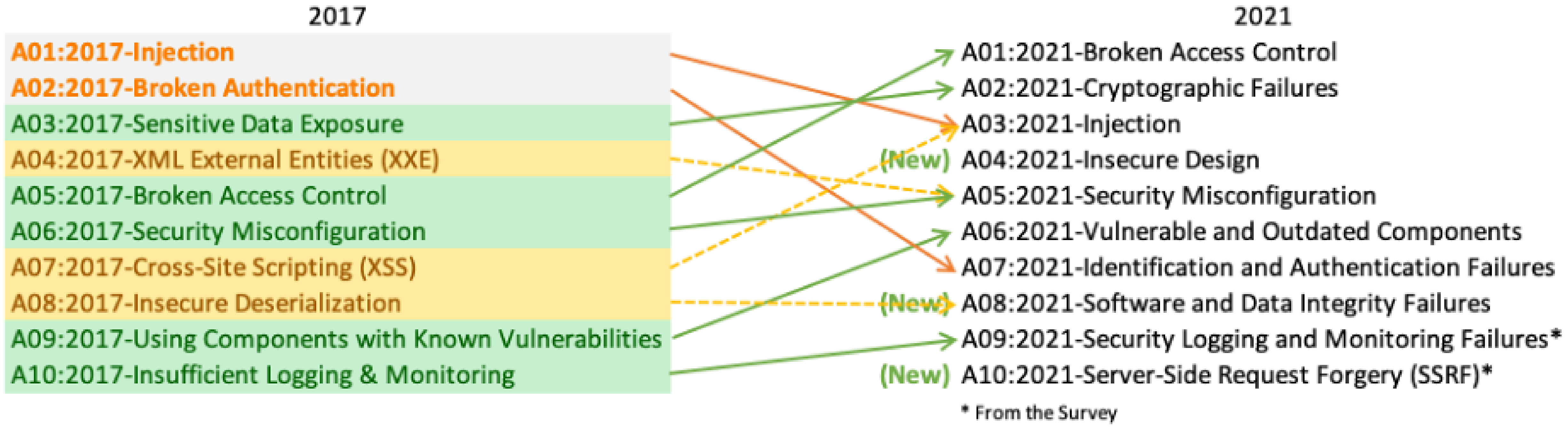
- Crash the application to prove that our input dictates the application’s behavior.
- Retrieve usernames from the database for a targeted attack to bypass authentication.
- Extract out useful information from the database (we will be gathering password hashes).
- Crack the password hashes so we know the username and password of each of the application users.

OWASP – Open Source Foundation for Application Security

- "*OWASP is an online community that produces freely available articles, methodologies, documentation, tools, and technologies in the fields of IoT, system software and web application security*"[1]
- OWASP top ten is a regularly updated list of the most critical risks.
- Some of the other OWASP published work:
    - OWASP- development guide, testing guide, code review guide,
    - OWASP Application Security Verification Standard (ASVS)
    - OWASP Top 10 Incident Response Guidance
    - OWASP Software Assurance Maturity Model (SAMM)

[1] https://en.wikipedia.org/wiki/OWASP

OWASP Top Ten



2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

https://owasp.org/www-project-top-ten/

Questions?

Basel.katt@ntnu.no