10 WEEKS TO 0 CRITICAL VULNERABILITIES

INJECTION ATTACKS

Sherif Koussa

@skoussa

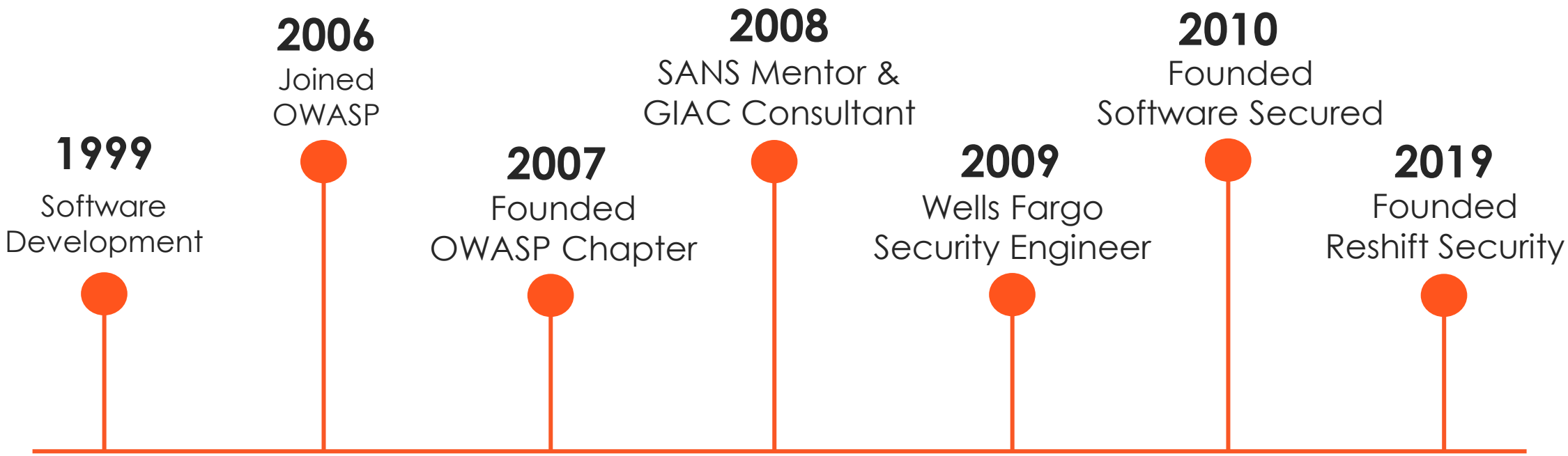
SoftwareSecured

HOUSEKEEPING

- Why we're all here
- Recording for internal training purposes
- Slides will be provided after the session
- Your mic will be muted, please use "Q&A" for any questions

SoftwareSecured

ABOUT ME



Certifications: GSSP-Java, GSSP-NET, GWAPT

SoftwareSecured

Reshift integrates with your modern software development pipeline to help your team find and fix vulnerabilities.

SoftwareSecured

Penetration Testing as a Service company based out of Ottawa, Canada.

10 WEEK SCHEDULE

1. **April 10th: Injection**
2. April 17th : Broken Authentication
3. April 24th: Sensitive data Exposure
4. May 1st : External Entity Injection
5. May 8th : Broken Access Control
6. May 15th: Security Misconfiguration
7. May 22nd: Cross-site Scripting
8. May 29th: Insecure Deserialization
9. June 5th: Using Components with Known Vulnerabilities
10. June 12th: Insufficient Logging and Monitoring

SESSION 1: AGENDA

1. What are Injection Attacks and their impacts

2. Injection Theory

3. Types of Injection Attacks:
   - SQL Injection (Exercise)
   - JavaScript Server Side Injection
   - NoSQL Injection

5. Injection Attacks Mitigation

6. Tools and Resources

WHAT ARE INJECTION ATTACKS

Injection attacks denote a wide range of attacks targeting the server, where the attacker supplies untrusted input to software.

This gets processed by an interpreter as part of a command or query.

In turn, the input alters the course of execution of that program.

IMPACTS OF INJECTION ATTACKS

- Authentication bypass
- Privilege escalation
- Application logic bypass
- Database dump
- Denial of service
- Full system compromise

REAL COMPANY INJECTION ATTACKS

- **2011 Sony PlayStation** – 77 Million records - SQL Injection
- **2016 Adult Friend Finder** – 412 Million records – SQL Injection
- **2017 Equifax Breach** – 148 Million users - Code Injection
- **2019 Zynga** – 218 Million user accounts – SQL Injection
- **2019 Canva** – 137 Million user accounts – SQL Injection

INJECTION THEORY

TYPES OF INJECTION ATTACKS

- **SQL Injection**
- **Server-Side JavaScript Injection**
- **NoSQL Injection**
- Command Injection
- XML Injection
- LDAP Injection
- XPath Injection
- Code Injection

SQL INJECTION

WHAT IS SQL INJECTION

SQL injection is a technique where an attacker would alter SQL statements using untrusted input, modifying the intended use of that SQL statement.

SQL INJECTION ATTACK

Consider the code (intended use)

12

```
2
3  public static ResultSet getReviews(Connection con, HttpServletRequest request) throws SQLException {
4
5      ResultSet rst = null;
6      Statement stmt = con.createStatement();
7
8      rst = stmt.executeQuery("SELECT * FROM Reviews WHERE ProductID = " + request.getParameter("id"));
9
10     return rst;
11 }
12
```

Resulting SQL:

```
SELECT * FROM Reviews WHERE ProductID = 12
```

SoftwareSECURED

SQL INJECTION ATTACK

12 UNION SELECT * FROM USERS;

Consider the code (abuse case)

```
2
3  public static ResultSet getReviews(Connection con, HttpServletRequest reque     rows SQLException {
4
5      ResultSet rst = null;
6      Statement stmt = con.createStatement();
7
8      rst = stmt.executeQuery("SELECT * FROM Reviews WHERE ProductID = " + request.getParameter("id"));
9
10     return rst;
11 }
12
```

Resulting SQL:

```
SELECT * FROM Reviews WHERE ProductID = 12 UNION SELECT * FROM USERS;
```

SoftwareSecured

STORED PROCEDURES & SQL INJECTION

Consider the stored procedure (intended use)

```
CREATE PROCEDURE usp_GetEmployee @employeeName varchar(50) = NULL AS
DECLARE @sql nvarchar(4000)
SELECT @sql = ' SELECT EmployeeMiddleName, EmployeeSurname, SSN ' +
              ' FROM HumanResources.Employee ' +
              ' WHERE '
IF @employeeName IS NOT NULL
   SELECT @sql = @sql + ' employeeName LIKE ''' + @employeeName + ''''
EXEC (@sql)
```

John

Resulting Query:

```
SELECT EmployeeMiddleName, EmployeeSurname, SSN FROM
HumanResources.Employee WHERE employeeName LIKE 'John'
```

SoftwareSecured

STORED PROCEDURES & SQL INJECTION

Consider this stored procedure (abuse case)

```
CREATE PROCEDURE usp_GetEmployee @employeeName varchar(50) = NULL AS
DECLARE @sql nvarchar(4000)
SELECT @sql = ' SELECT EmployeeMiddleName, EmployeeSurname, SSN ' +
              ' FROM HumanResources.Employee ' +
              ' WHERE '
IF @employeeName IS NOT NULL
   SELECT @sql = @sql + ' employeeName LIKE ''' + @employeeName + ''''
EXEC (@sql)
```

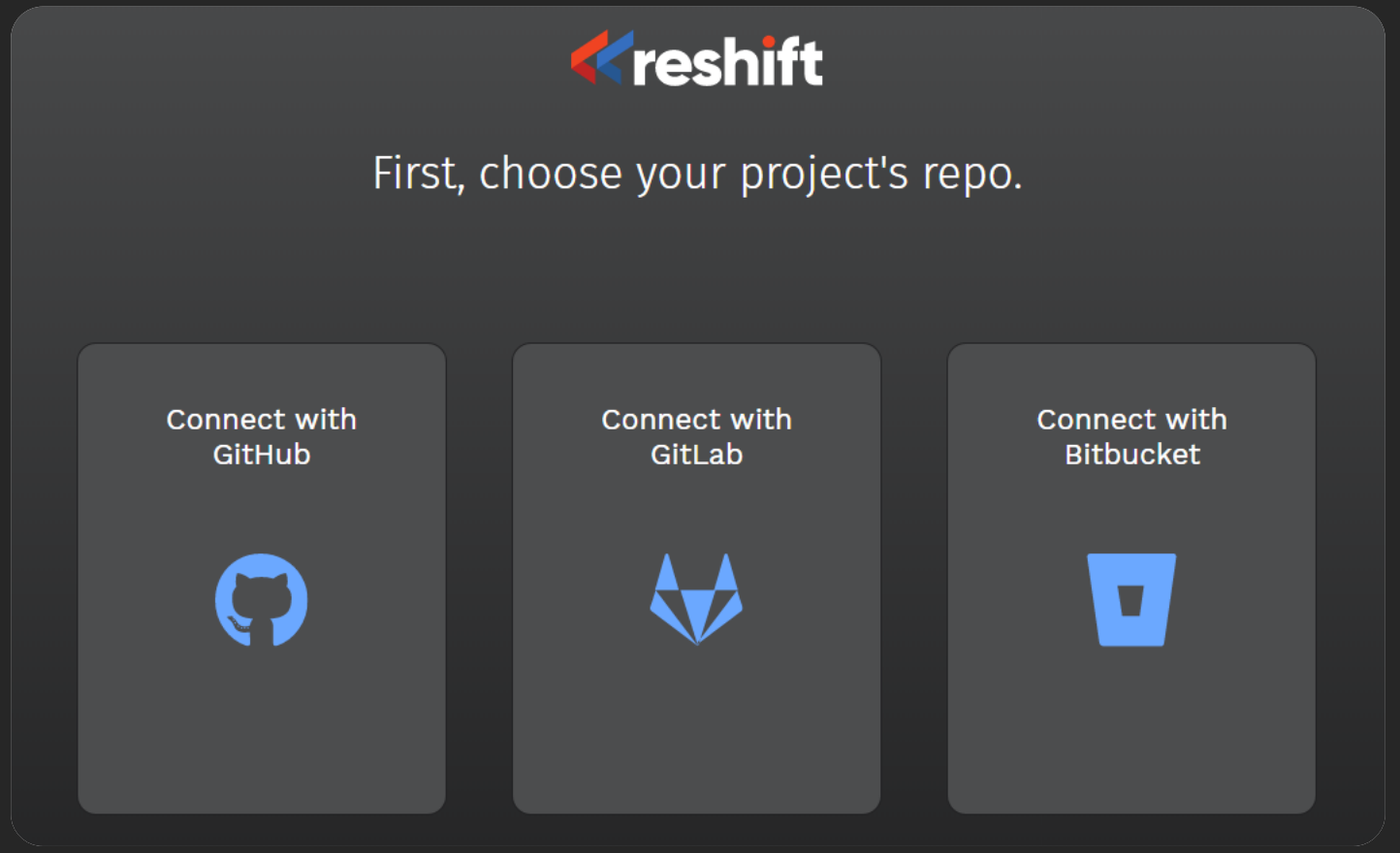
John' or '1'='1

Resulting Query:

```
SELECT EmployeeMiddleName, EmployeeSurname, SSN FROM
HumanResources.Employee WHERE employeeName LIKE 'John' or '1'='1'
```

SoftwareSECURED

EXERCISE: IDENTIFY SQL INJECTION USING RESHIFT



FIXING SQL INJECTION

- Use of Prepared Statements

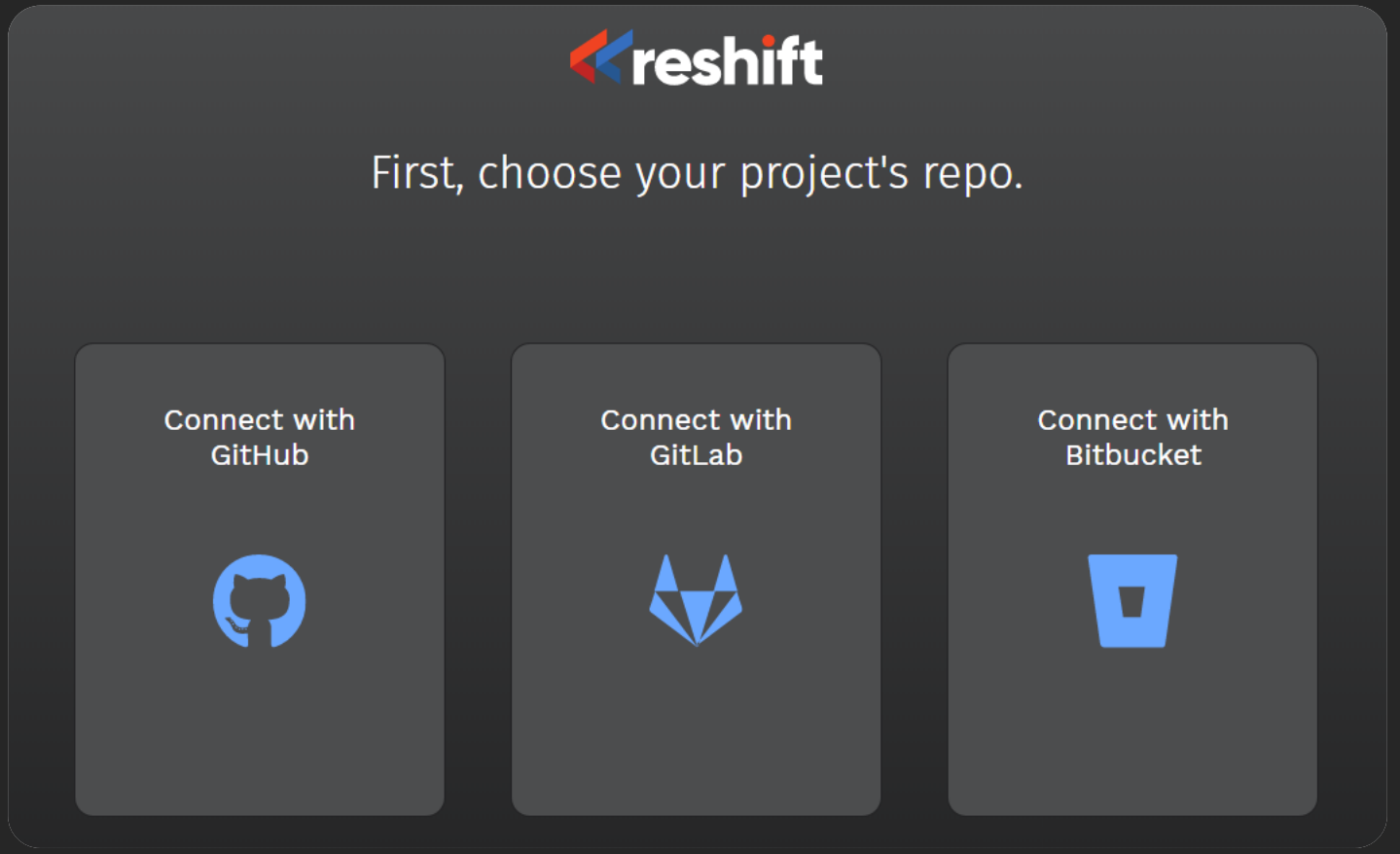
  - **Java EE** – use PreparedStatement() with bind variables
  - **.NET** – use parameterized queries like SqlCommand() or OleDbCommand() with bind variables
  - **PHP** – use PDO with strongly typed parameterized queries (using bindParam())
  - **Hibernate** - use createQuery() with bind variables (called named parameters in Hibernate)
  - **SQLite** - use sqlite3_prepare() to create a statement object

FIXING SQL INJECTION

- Example of Prepared Statements

```
3  public static ResultSet getReviews(Connection con, HttpServletRequest request) throws SQLException {
4
5      ResultSet rst = null;
6      PreparedStatement stmt = con.prepareStatement("SELECT * FROM Reviews WHERE ProductID = ?");
7
8      stmt.setString(1, request.getParameter("id"));
9
10     stmt.executeQuery();
11
12     return rst;
13 }
14
```

EXERCISE: FIXING SQL INJECTION USING RESHIFT



SoftwareSecured

SERVER-SIDE JAVASCRIPT INJECTION

WHAT IS SERVER-SIDE JAVASCRIPT INJECTION

Server-side JavaScript injection happens when an application incorporates untrusted data into a string that is dynamically evaluated by an interpreter.

SERVER-SIDE JAVASCRIPT INJECTION

Consider the server-side JavaScript code (intended use)

```
1  var http = require('http');
2  http.createServer(function (request, response) {
3  if (request.method === 'POST') {
4
5   var data = '';
6
7   request.addListener('data', function(chunk) {
8   data += chunk; });
9
10  request.addListener('end', function() {
11  var stockQuery = eval("(" + data + ")");
12  getStockPrice(stockQuery.symbol);
13  …
```

{"symbol" : "AMZN"}

Resulting Query:

var stockQuery = eval(“{"symbol" : "AMZN"}”);

SoftwareSECURED

SERVER-SIDE JAVASCRIPT INJECTION

Consider the server-side JavaScript code (abuse case)

```
1  var http = require('http');
2  http.createServer(function (request, response) {
3  if (request.method === 'POST') {
4
5   var data = '';
6
7   request.addListener('data', function(chunk) {
8   data += chunk; });
9
10  request.addListener('end', function() {
11  var stockQuery = eval("(" + data + ")");
12  getStockPrice(stockQuery.symbol);
13  …
```

{while(1)}

Resulting Query:

```
var stockQuery = eval("{while(1)} ");
```

SoftwareSECURED

SERVER-SIDE JAVASCRIPT INJECTION

Consider the server-side JavaScript code (2nd abuse case)

```
1  var http = require('http');
2  http.createServer(function (request, response) {
3  if (request.method === 'POST') {
4
5   var data = '';
6
7   request.addListener('data', function(chunk) {
8   data += chunk; });
9
10  request.addListener('end', function() {
11  var stockQuery = eval("(" + data + ")");
12  getStockPrice(stockQuery.symbol);
13  …
```

{process.exit()}

Resulting Query:

```
var stockQuery = eval("{process.exit()} ");
```

SoftwareSECURED

FIXING JAVASCRIPT SERVER SIDE INJECTION

- Avoid *eval()* at all cost and use *JSON.parse()* instead.
- Avoid creating "ad-hoc" JavaScript commands by concatenating script with user input.
- Validate untrusted input used with server-side JS commands using regular expressions.

NO-SQL INJECTION

WHAT IS NOSQL INJECTION

NoSQL injection is a technique where an attacker would alter NoSQL queries using untrusted input, modifying the intended use of that query.

NOSQL INJECTION

Consider the NoSQL Injection code (intended use)

```
3
4
5 db.accounts.find({username: username, password: password});
6
7
```

{
“username”: “admin”,
“Password”: “MySecurePass”
}

db.accounts.find({username: “admin”, password: “MySecurePass”});

SoftwareSecured

NOSQL INJECTION

Consider the NoSQL Injection code (abuse case)

```
3
4
5 db.accounts.find({username: username, password: password});
6
7
```

{
“username”: “admin”,
“Password”: {'&ne': “”}
}

db.accounts.find({username: “admin”, password: {'&ne': “”});

SoftwareSecured

FIXING NOSQL INJECTION

- Don't use the Mongo *where*, *mapReduce*, or *group* with user supplied data.
- Use a typed model. Typed models will automatically stop some injections by converting user input to the expected type, such as string or int.
- Set *javascriptEnabled* to false in your mongod.conf, if you can. This will disable JavaScript execution in your instance and remove that class of attacks.
- Always strongly validate user supplied data - this will help prevent a lot more than just NoSQL attacks.

TOOLS

- Reshift – a developer-first code security tool that finds and automatically fix vulnerabilities: www.reshiftsecurity.com
- SpotBugs – a static code analysis that detects Java vulnerabilities with a security plugin: FindSecBugs
- ZAP Proxy – is an open-source web app security scanner.
- Burp Suite – used for penetration testing of web applications.
- SQLMap – an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers
- NoSqlMap – an open-source Python tool to audit NoSQL databases

RESOURCES:

- Introduction to SQL Injection Mitigation by Software Secured
- SQL Injection Prevention Cheat Sheet by OWASP
- A NoSQL Injection Primer (Mongo)
- A tool and a tutorial to practice NoSQLi and Server-Side JavaScript injection examples
- An extensive list of ways NoSQL databases could be abused
- LDAP Injection Prevention Cheat Sheet by OWASP
- OS Command Injection Prevention Cheat Sheet by OWASP

SESSION 1: AGENDA

1. What are Injection Attacks and their impacts

2. Injection Theory

3. Types of Injection Attacks:
   - SQL Injection (Exercise)
   - JavaScript Server Side Injection
   - NoSQL Injection

5. Injection Attacks Mitigation

6. Tools and Resources

10 WEEKS TO 0 VULNERABILITIES PROGRAM

1. April 10th: Injection
2. **April 17th : Broken Authentication**
3. April 24th: Sensitive data Exposure
4. May 1st : External Entity Injection
5. May 8th : Broken Access Control
6. May 15th: Security Misconfiguration
7. May 22nd: Cross-site Scripting
8. May 29th: Insecure Deserialization
9. June 5th: Using Components with Known Vulnerabilities
10. June 12th: Insufficient Logging and Monitoring

Sign up for next week's session:

Register

SoftwareSecured

SoftwareSecured

THANK YOU ☺

sherif@softwaresecured.com
@skoussa