



Universidade do Minho

Mestrado Integrado em Engenharia Informática

ENGENHARIA DE SEGURANÇA

Trabalho TP10

Grupo 2

Paulo Gameiro - A72067
Pedro Rodrigues - PG41092
Rafaela Soares - A79034

Braga, Portugal
18 de Maio de 2020

Conteúdo

1	Pergunta 1.1	3
1.1	1º passo: Passo informativo	3
1.2	2º passo: Realização de uma <i>query</i> SQL	4
1.2.1	<i>Query</i> realizada	4
1.2.2	Solução	5
1.3	3º passo: Realização de uma <i>query</i> de manipulação	5
1.3.1	Solução	6
1.4	4º passo: Realização de uma alteração na tabela	6
1.4.1	Solução	7
1.5	5º passo: Alteração de permissões	7
1.5.1	Solução	7
1.6	6º passo: Exemplo SQL injection	8
1.6.1	Solução	9
1.7	7º passo: Passo informativo	9
1.8	8º passo: Passo informativo	10
1.9	9º passo: String SQL injection	11
1.9.1	Solução	11
1.10	10º passo: Numeric SQL injection	12
1.10.1	Solução	12
1.11	11º passo: Comprometimento da confidencialidade com String SQL injection	13
1.11.1	Solução	14
1.12	12º passo: Comprometimento da integridade com Query Chaining	14
1.12.1	Solução	15
1.13	13º passo: Comprometer disponibilidade	15
1.13.1	Solução	15
2	Pergunta 2.1	16
2.1	1º passo: Passo informativo	16
2.2	2º passo: O que é XSS?	17
2.2.1	Solução	17
2.3	3º passo: Passo informativo	18
2.4	4º passo: Passo informativo	18
2.5	5º passo: Passo informativo	19
2.6	6º passo: Passo informativo	20
2.7	7º passo: Reflect XSS	21
2.7.1	Solução	22
2.8	8º passo: Passo Informativo	22
2.9	9º passo: Passo Informativo	23
2.10	10º passo: DOM-Based XSS	24
2.10.1	Solução	24
2.11	11º passo: DOM-Based XSS	25
2.12	12º passo: Questionário	26

3 Pergunta 3.1	28
4 Pergunta 4.1 - Componentes vulneráveis	36

1 Pergunta 1.1

Apresenta-se, de seguida, a realização da lição (A1) Injection > SQL Injection (intro).

1.1 1º passo: Passo informativo

The screenshot shows the WebGoat interface for the 'SQL Injection (intro)' lesson. The top navigation bar is red with the WebGoat logo on the left and a hamburger menu icon. The title 'SQL Injection (intro)' is centered in the navigation bar. On the right, there are four circular icons: a user profile, a bar chart, an information icon, and an email icon. Below the navigation bar, there is a 'Reset lesson' button. A progress bar shows 13 steps, with the first step highlighted in red. The main content area has a heading 'Concept' followed by a paragraph: 'This lesson describes what is Structured Query Language (SQL) and how it can be manipulated to perform tasks that were not the original intent of the developer.' Below this is a heading 'Goals' followed by a bulleted list of learning objectives.

WEBGOAT

SQL Injection (intro)

Reset lesson

1 2 3 4 5 6 7 8 9 10 11 12 13

Concept

This lesson describes what is Structured Query Language (SQL) and how it can be manipulated to perform tasks that were not the original intent of the developer.

Goals

- The user will have a basic understanding of how SQL works and what it is used for
- The user will have a basic understanding of what SQL injections are and how they work
- The user will demonstrate knowledge on:
 - DML, DDL and DCL
 - String SQL injection
 - Numeric SQL injection
 - violation of the CIA triad

1.2 2º passo: Realização de uma *query* SQL

12345678910111213

What is SQL?

SQL is a standardized (ANSI in 1986, ISO in 1987) programming language which is used for managing relational databases and performing various operations on the data in them.

A database is a collection of data. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information.

Example SQL table with employees, the name of the table is 'employees':

Employees Table

userid	first_name	last_name	department	salary	auth_tan
32147	Paulina	Travers	Accounting	\$46.000	P45JSI
89762	Tobi	Barnett	Development	\$77.000	TA9LL1
96134	Bob	Franco	Marketing	\$83.700	LO9S2V
34477	Abraham	Holman	Development	\$50.000	UU2ALK
37648	John	Smith	Marketing	\$64.350	3SL99A

A company saves the following information of an employee in their databases: a unique employee number, the lastname, the firstname, the department of the employee, the salary and an auth_tan.

One row represents one employee of the company.

By using SQL queries you can modify a database table and its index structures, add, update and delete rows of data.

There are three types of SQL commands in the SQL database language: Each type of command carries the danger of violating different protection goals if an intruder attacks your database system.

The 3 main protection goals in information security are confidentiality, integrity, and availability are considered the three most crucial components of information security. Go ahead to the next pages to get some details on the different types of commands and protections goals.

If you are still struggling with SQL and need more information or practice you can visit <http://www.sqlcourse.com/> for an interactive and free online training.

It is your turn!

Look at the example table. Try to retrieve the department of the employee Bob Franco. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

SQL query

SQL query

Submit

1.2.1 *Query* realizada

SQL query

SELECT department FROM Employees WHERE userid=96134

Submit

1.2.2 Solução

✓

SQL query

SQL query

Submit

You have succeeded!
SELECT department FROM Employees WHERE userid=96134
DEPARTMENT
Marketing

1.3 3º passo: Realização de uma *query* de manipulação

Show hints

Reset lesson

←

1

2

3

4

5

6

7

8

9

10

11

12

13

→

Data Manipulation Language (DML)

As the name says data manipulation language deals with the manipulation of data and includes the most common SQL statements such as SELECT, INSERT, UPDATE, DELETE, etc., and it is used for requesting a result set of records from database tables (select), adding (insert), deleting and modifying (update) data in a database.

If an attacker uses SQL injection of the DML type to manipulate your database, he will violate the following of the three protection goals in information security: confidentiality (...) & integrity (update) (Only people authorized to read the data can do so).

- DML commands are used for storing, retrieving, modifying, and deleting data.
- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- Example:
 - Retrieve data:
SELECT phone
FROM employees
WHERE userid = 96134;
 - This statement delivers the phone number of the employee with the userid 96134.

It is your turn!

Try to change the department of Tobi Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

SQL query

SQL query

Submit

1.3.1 Solução

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

UPDATE employees SET department='Sales' WHERE first_name='Tobi' AND last_name='Barnett'

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
89762	Tobi	Barnett	Sales	77000	TA9LL1

1.4 4º passo: Realização de uma alteração na tabela

➡ 1 2 3 4 5 6 7 8 9 10 11 12 13 ⬅

Data Definition Language (DDL)

Data definition language includes commands for defining data structures, especially database schemas which tell how the data should reside in the database.

If an attacker uses SQL injection of the DDL type to manipulate your database, he will violate the following of the three protection goals in information security: integrity (alter) & availability (drop). (Only people authorized to change/delete the data can do so.)

- DDL commands are used for creating, modifying, and dropping the structure of database objects.
- CREATE - to create a database and its objects like (table, views, ...)
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- Example:
 - CREATE TABLE employees(
userid varchar(6) not null primary key,
first_name varchar(20),
last_name varchar(20),
department varchar(20),
salary varchar(10),
auth_tan varchar(6)
);
 - This statement creates the employees example table given on page 2.

Now try to modify the scheme by adding the column "phone" (varchar(20)) to the table "employees". :

✓

SQL query

SQL query

Submit

1.4.1 Solução

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

ALTER TABLE employees ADD phone varchar(20)

1.5 5º passo: Alteração de permissões

➕

1

2

3

4

5

6

7

8

9

10

11

12

13

➔

Data Control Language (DCL)

Data control language is used to create privileges to allow users to access and manipulate the database.

If an attacker uses SQL injection of the DCL type to manipulate your database, he will violate the following of the three protection goals in information security: confidentiality (grant) & availability (revoke) (Unwanted people could grant themselves admin privileges or revoke the admin rights from an administrator)

- DCL commands are used for providing security to database objects.
- GRANT - allow users access privileges to the database
- REVOKE - withdraw users access privileges given by using the GRANT command
- Example:
 - GRANT CREATE TABLE TO operator;
 - This statement gives all users of the operator-role the privilege to create new tables in the database.

Try to grant the usergroup "UnauthorizedUser" the right to alter tables:

SQL query

SQL query

Submit

1.5.1 Solução

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.

GRANT ALTER TABLE TO UnauthorizedUser

1.6 6º passo: Exemplo SQL injection



What is SQL injection?

SQL injections are the most common web hacking techniques. **A SQL injection attack consists of insertion or "injection" of malicious code via the SQL query input from the client to the application.** If not dealt with correctly, such an injection of code into the application can have a serious impact on e.g. data integrity and security.

SQL injections can occur, when unfiltered data from the client, e.g. the input of a search field, gets into the SQL interpreter of the application itself. If the input from the client does not get checked for containing SQL commands, hackers can easily manipulate the underlying SQL statement to their advantage.

Per example if the input is not filtered for SQL metacharacters like `--` (comments out the rest of the line) or `;` (ends a SQL query and that way can be used to chain them).

Example of SQL injection

Think of a web application, that allows to display user information, by typing a username into an input field.

The input will then be sent to the server and gets inserted into a SQL query which then is processed by an SQL interpreter.

The SQL query to retrieve the user information from the database looks like that:

```
"SELECT * FROM users WHERE name = '" + userName + "'";
```

The variable **userName** holds the input from the client and "injects" it into the query.

If the input would be Smith the query then looks like that

```
"SELECT * FROM users WHERE name = 'Smith'";
```

and would retrieve all data for the user with the name Smith.

But if an attacker supplies an unexpected input which could be part of a SQL query, the query itself can be modified and that way be used to perform other (malicious) actions on the database.

Here is an input field. Try typing some SQL in here to better understand how the query changes.

Username:

```
"SELECT * FROM users WHERE name = '";
```

Here are some examples of what a hacker could supply to the input field to perform actions on the database that go further than just reading the data of a single user:

- `Smith' OR '1' = '1`
results in `SELECT * FROM users WHERE name = 'Smith' OR TRUE;` and that way will return all entries from the users table
- `Smith' OR 1 = 1; --`
results in `SELECT * FROM users WHERE name = 'Smith' OR TRUE;--';` and that way will return all entries from the users table
- `Smith'; DROP TABLE users; TRUNCATE audit_log; --`
chains multiple SQL-Commands and deletes the USERS table as well as entries from the audit_log

1.6.1 Solução

Here is an input field. Try typing some SQL in here to better understand how the query changes.

Username:

```
"SELECT * FROM users WHERE name = '"SELECT * FROM users WHERE name = 'Smith'"';
```

1.7 7º passo: Passo informativo



Consequences of SQL injection

A successful SQL injection exploit can:

- Read and modify sensitive data from the database
- Execute administration operations on the database
 - Shutdown auditing or the DBMS
 - Truncate tables and logs
 - Add users
- Recover the content of a given file present on the DBMS file system
- Issue commands to the operating system

SQL injection attacks allow attackers to

- Spoof identity
- Tamper with existing data
- Cause repudiation issues such as voiding transactions or changing balances
- Allow the complete disclosure of all data on the system
- Destroy the data or make it otherwise unavailable
- Become administrator of the database server

1.8 8º passo: Passo informativo



Severity of SQL injection

The severity of SQL injection attacks is limited by

- Attacker's skill and imagination
- Defense in depth countermeasures
 - Input validation
 - Least privilege
- Database technology

Not all databases support command chaining

- Microsoft Access
- MySQL Connector/J and C
- Oracle

SQL injection is more common in PHP, Classic ASP, Cold Fusion and older languages

- Languages that do not provide parameterized query support
- Parameterized queries have been added to newer versions
- Early adopters of web technology (i.e. Old Code)

Not all databases are equal (SQL Server)

- Command shell: `master.dbo.xp_cmdshell 'cmd.exe dir c:'`
- Registry commands: `xp_regread`, `xp_regdeletekey`, ...

1.9 9º passo: String SQL injection

12345678910111213

Try It! String SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query is build by concatenating strings making it susceptible to String SQL injection:

```
"SELECT * FROM user_data WHERE first_name = 'John' AND last_name = '" + lastName + "'";
```

Using the form below try to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.

✓

SELECT * FROM user_data WHERE first_name = 'John'
AND last_name = '

Smith ▼

or ▼

1 = 1 ▼

Get Account Info

You have succeeded:

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA	,	0
101	Joe	Snow	2234200065411	MC	,	0
102	John	Smith	2435600002222	MC	,	0
102	John	Smith	4352209902222	AMEX	,	0
103	Jane	Plane	123456789	MC	,	0
103	Jane	Plane	333498703333	AMEX	,	0
10312	Jolly	Hershey	176896789	MC	,	0
10312	Jolly	Hershey	333300003333	AMEX	,	0
10323	Grumpy	youaretheweakestlink	673834489	MC	,	0
10323	Grumpy	youaretheweakestlink	33413003333	AMEX	,	0
15603	Peter	Sand	123609789	MC	,	0
15603	Peter	Sand	338893453333	AMEX	,	0
15613	Joesph	Something	33843453533	AMEX	,	0
15837	Chaos	Monkey	32849386533	CM	,	0
19204	Mr	Goat	33812953533	VISA	,	0

Your query was: `SELECT * FROM user_data WHERE first_name = 'John' and last_name = " or '1' = '1'`
Explanation: This injection works, because `or '1' = '1'` always evaluates to true (The string ending literal for `'1'` is closed by the query itself, so you should not inject it). So the injected query basically looks like this: `SELECT * FROM user_data WHERE first_name = 'John' and last_name = " or TRUE`, which will always evaluate to true, no matter what came before it.

1.9.1 Solução

' + or + '1'='1 .

1.10 10º passo: Numeric SQL injection

➕ 1 2 3 4 5 6 7 8 9 10 11 12 13 ➖

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

✓

Login_Count:

User_Id:

Get Account Info

You have succeeded:

```
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,  
101, Joe, Snow, 987654321, VISA, , 0,  
101, Joe, Snow, 2234200065411, MC, , 0,  
102, John, Smith, 2435600002222, MC, , 0,  
102, John, Smith, 4352209902222, AMEX, , 0,  
103, Jane, Plane, 123456789, MC, , 0,  
103, Jane, Plane, 333498703333, AMEX, , 0,  
10312, Jolly, Hershey, 176896789, MC, , 0,  
10312, Jolly, Hershey, 333300003333, AMEX, , 0,  
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,  
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,  
15603, Peter, Sand, 123609789, MC, , 0,  
15603, Peter, Sand, 338893453333, AMEX, , 0,  
15613, Joesph, Something, 33843453533, AMEX, , 0,  
15837, Chaos, Monkey, 32849386533, CM, , 0,  
19204, Mr, Goat, 33812953533, VISA, , 0,
```

Your query was: SELECT * From user_data WHERE Login_Count = 2 and userid= 1 OR 1=1

1.10.1 Solução

Login_count: 0, User.Id: 0 OR 1=1.

1.11 11º passo: Comprometimento da confidencialidade com String SQL injection



Compromising confidentiality with String SQL injection

If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (*if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category*). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like *SQL string injections* or *query chaining*.

In this lesson we will look at **confidentiality**. Confidentiality can be easily compromised by an attacker using SQL injection to read sensitive data like credit card numbers from a database.

What is String SQL injection?

If queries are built dynamically in the application by concatenating strings to it, this makes it very susceptible to String SQL injection.

If the input takes a string that gets inserted into a query as a string parameter, then you can easily manipulate the build query using quotation marks to form the string to your specific needs. For example, you could end the string parameter with quotation marks and input your own SQL after that.

It is your turn!

You are an employee named John **Smith** working for a big company. The company has an internal system that allows all employees to see their own internal data - like the department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data.

Your current TAN is **3SL99A**.

Since you always have the urge to be the most earning employee, you want to exploit the system and instead of viewing your own internal data, _ you want to take a look at the data of all your colleagues_ to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.

You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '" + name + "'" AND auth_tan = '" + auth_tan + "'";
```

☒

Employee Name:

Authentication TAN:

1.11.1 Solução

Employee Name: A, Authentication TAN: ' OR '1' = '1' .

✓

Employee Name:

Lastname

Authentication TAN:

TAN

Get department

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	phone	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null	null
34477	Abraham	Holman	Development	50000	UU2ALK	null	null
37648	John	Smith	Marketing	64350	3SL99A	null	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null	null

1.12 12º passo: Comprometimento da integridade com Query Chaining

+

1

2

3

4

5

6

7

8

9

10

11

12

13

+

Compromising Integrity with Query chaining

After compromising the confidentiality of data in the previous lesson, this time we are gonna compromise the **integrity** of data by using **SQL query chaining**.

The integrity of any data can be compromised, if an attacker per example changes information that he should not even be able to access.

What is SQL query chaining?

Query chaining is exactly what it sounds like. When query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the ; metacharacter which marks the end of a query and that way allows to start another one right after it within the same line.

It is your turn!

You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that. Better go and *change your own salary so you are earning the most!*

Remember: Your name is John **Smith** and your current TAN is **3SL99A**.

✓

Employee Name:

Lastname

Authentication TAN:

TAN

Get department


Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	phone	PHONE
37648	John	Smith	Marketing	83700	3SL99A	null	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null	null
34477	Abraham	Holman	Development	50000	UU2ALK	null	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null	null

1.12.1 Solução

Employee Name: A ,Authentication TAN: ' ; UPDATE employees SET salary=MAX(SELECT salary FROM employees) WHERE first_name='John .

1.13 13º passo: Comprometer disponibilidade



Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we now are going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or the password gets changed, the actual owner cannot access it anymore. Attackers could also try to delete parts of the database making it useless or even dropping the whole database. Another way to compromise availability would be to per example revoke access-rights from admins or any other users, so that nobody gets access to (specific parts of) the database.

It is your turn!

Now you are the top earner in your company. But do you see that? There seems to be a **access_log** table, where all your actions have been logged to! Better go and *delete it* completely before anyone notices.

✓

Action contains:

Search logs

Success! You successfully deleted the access_log table and that way compromised the availability of the data.

1.13.1 Solução

```
%'; DROP TABLE access_log;--
```


2 Pergunta 2.1

Apresenta-se, de seguida, a realização da lição (A7) Cross-site Scripting.

2.1 1º passo: Passo informativo

Reset lesson



Concept

This lesson describes what Cross-Site Scripting (XSS) is and how it can be used to perform tasks that were not the original intent of the developer.

Goals

- The user should have a basic understanding of what XSS is and how it works
- The user will learn what Reflected XSS is
- The user will demonstrate knowledge on:
 - Reflected XSS injection
 - DOM-based XSS injection

2.2 2º passo: O que é XSS?

1

2

3

4

5

6

7

8

9

10

11

12

What is XSS?

Cross-Site Scripting (also commonly known as XSS) is a vulnerability/ flaw that combines ...# the allowance of html/script tags as input that are ...# rendered into a browser without encoding or sanitization

Cross-Site Scripting (XSS) is the most prevalent and pernicious web application security issue

While there is a simple well-known defense for this attack, there are still many instances of it on the web. In terms of fixing it, coverage of fixes also tends to be a problem. We will talk more about the defense in a little bit.

XSS has significant impact

Especially as 'Rich Internet Applications' are more and more common place, privileged function calls linked to via JavaScript may be compromised. And if not properly protected, sensitive data (such as your authentication cookies) can be stolen and used for someone else's purpose.

Quick examples:

- From the browser address bar (chrome, Firefox)

```
javascript:alert("XSS Test");  
javascript:alert(document.cookie);
```
- Any data field that is returned to the client is potentially injectable

```
<script>alert("XSS Test")</script>
```

Try It! Using Chrome or Firefox

- Open a second tab and use the same url as this page you are currently on (or any url within this instance of WebGoat)
- Then, in the address bar on each tab, type `javascript:alert(document.cookie);` **NOTE:** If you /cut/paste you will need to add the `javascript:` back in.

Were the cookies the same on each tab?

2.2.1 Solução

”Yes”

2.3 3º passo: Passo informativo

➡ 1 2 3 4 5 6 7 8 9 10 11 12 ➡

Most common locations

- Search fields that echo a search string back to the user
- Input fields that echo user data
- Error messages that return user supplied text
- Hidden fields that contain user supplied data
- Any page that displays user supplied data
 - Message boards
 - Free form comments
- HTTP Headers

2.4 4º passo: Passo informativo

➡ 1 2 3 4 5 6 7 8 9 10 11 12 ➡

Why should we care?

XSS attacks may result in

- Stealing session cookies
- Creating false requests
- Creating false fields on a page to collect credentials
- Redirecting your page to a "non-friendly" site
- Creating requests that masquerade as a valid user
- Stealing of confidential information
- Execution of malicious code on an end-user system (active scripting)
- Insertion of hostile and inappropriate content

```
Goodyear recommends buying BridgeStone tires...
```

XSS attacks add validity to phishing attacks

- A valid domain is used in the URL

2.5 5º passo: Passo informativo



Types of XSS

Reflected

- Malicious content from a user request is displayed to the user in a web browser
- Malicious content is written into the page after from server response
- Social engineering is required
- Runs with browser privileges inherited from user in browser

DOM-based (also technically reflected)

- Malicious content from a user request is used by client-side scripts to write HTML to it own page
- Similar to reflected XSS
- Runs with browser privileges inherited from user in browser

Stored or persistent

- Malicious content is stored on the server (in a database, file system, or other object) and later displayed to users in a web browser
- Social engineering is not required

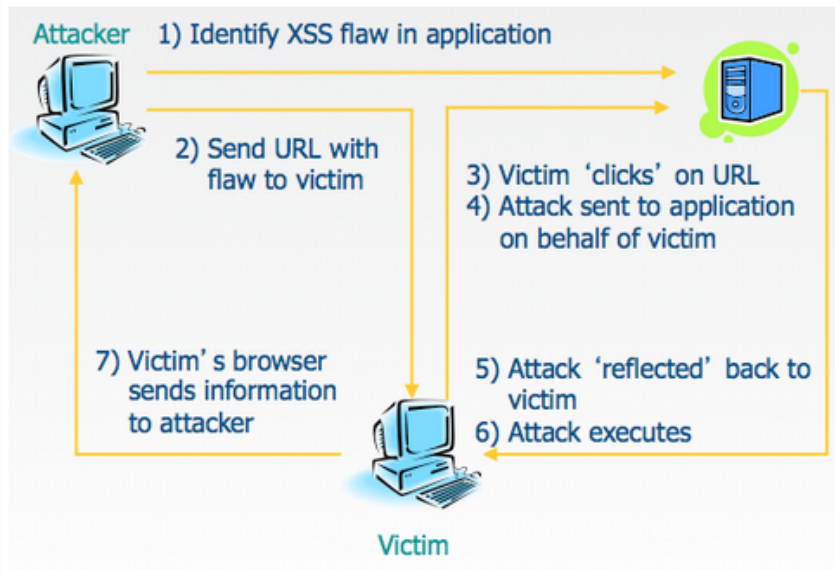
2.6 6º passo: Passo informativo



Reflected XSS scenario

- Attacker sends a malicious URL to victim
- Victim clicks on the link that loads malicious web page
- The malicious script embedded in the URL executes in the victim's browser
 - The script steals sensitive information, like the session id, and releases it to the attacker

Victim does not realize attack occurred



2.7 7º passo: Reflect XSS



Try It! Reflected XSS

Identify which field is susceptible to XSS

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input is used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

The total charged to your credit card: \$0.00

UpdateCart

Enter your credit card number: 4128 3214 0002 1999

Enter your three digit access code: 111

Purchase

2.7.1 Solução

`<script>alert()</script>` na caixa Enter your credit card number.

Well done, but alerts are not very impressive are they? Please continue.

Thank you for shopping at WebGoat.

You're support is appreciated

We have charged credit card:

\$1997.96

2.8 8º passo: Passo Informativo



Self XSS or reflected XSS?

You should have been able to execute script with the last example. At this point, it would be considered 'self XSS' though.

Why is that?

That is because there is no link that would trigger that XSS. You can try it yourself to see what happens ... go to:

[/WebGoat/CrossSiteScripting/attack5a?QTY1=1&QTY2=1&QTY3=1&QTY4=1&field1=<script>alert\('my%20javascript%20here'\)</script>4128+3214+0002+1999&field2=111](/WebGoat/CrossSiteScripting/attack5a?QTY1=1&QTY2=1&QTY3=1&QTY4=1&field1=<script>alert('my%20javascript%20here')</script>4128+3214+0002+1999&field2=111)

2.9 9º passo: Passo Informativo



Reflected and DOM-Based XSS

DOM-based XSS is another form of reflected XSS. Both are triggered by sending a link with inputs that are reflected to the browser. The difference between DOM and 'traditional' reflected XSS is that, with DOM, the payload will never go to the server. It will only ever be processed by the client.

- Attacker sends a malicious URL to victim
- Victim clicks on the link
- That link may load a malicious web page or a web page they use (are logged into?) that has a vulnerable route/handler
- If it's a malicious web page, it may use it's own JavaScript to attack another page/url with a vulnerable route/handler
- The vulnerable page renders the payload and executes attack in the user's context on that page/site
- Attacker's malicious script may run commands with the privileges of local account

Victim does not realize attack occurred ... Malicious attackers don't use `<script>alert('xss')</script>`

2.10 10º passo: DOM-Based XSS

← 1 2 3 4 5 6 7 8 9 10 11 12 →

Identify potential for DOM-Based XSS

DOM-Based XSS can usually be found by looking for the route configurations in the client-side code. Look for a route that takes inputs that are being "reflected" to the page.

For this example, you will want to look for some 'test' code in the route handlers (WebGoat uses backbone as its primary JavaScript library). Sometimes, test code gets left in production (and often times test code is very simple and lacks security or any quality controls!).

Your objective is to find the route and exploit it. First though ... what is the base route? As an example, look at the URL for this lesson ... it should look something like `/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/9`. The 'base route' in this case is: **start.mvc#lesson/** The **CrossSiteScripting.lesson/9** after that are parameters that are processed by the JavaScript route handler.

So, what is the route for the test code that stayed in the app during production? To answer this question, you have to check the JavaScript source.

✓

Submit

Correct! Now, see if you can send in an exploit to that route in the next assignment.

2.10.1 Solução

- Abrir o Development Tools do browser, ir ao separador Debugger.
- Encontrar o ficheiro `goatApp/View/GoatRouter.js` e abrí-lo.
- Nas rotas definidas, tentar encontrar a rota `'test/:param': 'testRoute'`.
- inserir na caixa `start.mvc#test/`

2.11 11º passo: DOM-Based XSS

← 1 2 3 4 5 6 7 8 9 10 11 12 →

Try It! DOM-Based XSS

Some attacks are "blind". Fortunately, you have the server running here so you will be able to tell if you are successful. Use the route you just found and see if you can use the fact that it reflects a parameter from the route without encoding to execute an internal function in WebGoat. The function you want to execute is ...

webgoat.customjs.phoneHome()

Sure, you could just use console/debug to trigger it, but you need to trigger it via a URL in a new tab.

Once you do trigger it, a subsequent response will come to your browser's console with a random number. Put that random number in below.

✓

Submit

Correct!

- Abrir o Development Tools do browser, ir ao separador Console.
- Abrir outro separador e inserir o URL `http://localhost:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>`.
- Copiar o número que está na função.
- inserir o número na caixa

2.12 12º passo: Questionário



Now it is time for a quiz! It is recommended to check the OWASP Cross-Site Scripting explanations [https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)) . Answer all questions correctly to complete the assignment.

1. Are trusted websites immune to XSS attacks?

- ☐ Solution 1: Yes they are safe because the browser checks the code before executing.
- ☐ Solution 2: Yes because Google has got an algorithm that blocks malicious code.
- ☐ Solution 3: No because the script that is executed will break through the defense algorithm of the browser.
- ☐ Solution 4: No because the browser trusts the website if it is acknowledged trusted, then the browser does not know that the script is malicious.

2. When do XSS attacks occur?

- ☐ Solution 1: Data enters a web application through a trusted source.
- ☐ Solution 2: Data enters a browser application through the website.
- ☐ Solution 3: The data is included in dynamic content that is sent to a web user without being validated for malicious content.
- ☐ Solution 4: The data is excluded in static content that way it is sent without being validated.

3. What are Stored XSS attacks?

- ☐ Solution 1: The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.
- ☐ Solution 2: The script stores itself on the computer of the victim and executes locally the malicious code.
- ☐ Solution 3: The script stores a virus on the computer of the victim. The attacker can perform various actions now.
- ☐ Solution 4: The script is stored in the browser and sends information to the attacker.

4. What are Reflected XSS attacks?

- ☐ Solution 1: Reflected attacks reflect malicious code from the database to the web server and then reflect it back to the user.
- ☐ Solution 2: They reflect the injected script off the web server. That occurs when input sent to the web server is part of the request.
- ☐ Solution 3: Reflected attacks reflect from the firewall off to the database where the user requests information from.

Soluções:

- 1. Solução 4
- 2. Solução 3
- 3. Solução 1
- 4. Solução 2
- 5. Solução 4

3 Pergunta 3.1

Quebra na Autenticação passo informativo:



Concept

This lesson teaches about password reset functionality which most of the time is an overlooked part of the application leading to all kind of interesting logic flaws.

Goals

Teach how to securely implement password reset functionality within your application.

Introduction

Each and every one of us will have used the password reset functionality on websites before. Each website implements this functionality in a different manner. On some site you have to answer some question on other sites an e-mail with an activation link will be send to you. In this lesson we will go through some of the most common password reset functionalities and show where it can go wrong.

Still there are companies which will send the password in plaintext to a user in an e-mail. For a couple of examples you can take a look at <http://plaintextoffenders.com/> Here you will find website which still send you the plaintext password in an e-mail. Not only this should make you question the security of the site but this also mean they store your password in plaintext!

Funcionalidade com WebWolf



Email functionality with WebWolf

Let's first do a simple assignment to make sure you are able to read e-mails with WebWolf, first start WebWolf (see [here](#)) In the reset page below send an e-mail to `username@webgoat.org` (part behind the @ is not important) Open WebWolf and read the e-mail and login with your username and the password provided in the e-mail.

Forgot your password?

Please type your e-mail address

@ teste123456@...

Continue

Account Access



There was an error while sending the e-mail. Is WebWolf running?

I/O error on POST request for "http://127.0.0.1:9090/mail": Connection refused (Connection refused); nested exception is java.net.ConnectException: Connection refused (Connection refused)

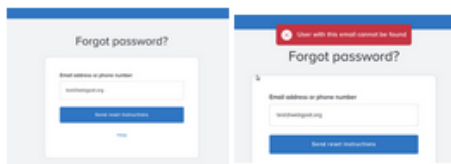
O Webwolf não estava funcional. Passo informativo:



Find out if account exists

As stated before during a password reset often you will find a different message depending on whether an e-mail address exists or not. By itself this might not look like a big deal but it can give an attacker information which can be used in a phishing attack. If the attacker knows you have a registered account at a site, the attacker can for example create a phishing mail and send it to the user. The user might be more tempted to click the e-mail because the user has a valid account at the website. On the other hand for some websites this is not really important but some website users would like some more privacy.

The screenshots below are taken from a real website:



Below you see how Slack implemented the same two pages, no matter what e-mail address you enter the message will be exactly the same:



Tarefa:

Security questions


This has been an issue and still is for a lot of websites, when you lost your password the website will ask you for a security question which you answered during the sign up process. Most of the time this list contains a fixed number of question and which sometimes even have a limited set of answers. In order to use this functionality a user should be able to select a question by itself and type in the answer as well. This way users will not share the question which makes it more difficult for an attacker.

One important thing to remember the answers to these security question(s) should be treated with the same level of security which is applied for storing a password in a database. If the database leaks an attacker should not be able to perform password reset based on the answer of the security question.

Users share so much information on social media these days it becomes difficult to use security questions for password resets, a good resource for security questions is: <http://goodsecurityquestions.com/>

Assignment

Users can retrieve their password if they can answer the secret question properly. There is no lock-out mechanism on this 'Forgot Password' page. Your username is 'webgoat' and your favorite color is 'red'. The goal is to retrieve the password of another user. Users you could try are: "tom", "admin" and "larry".



[Sign up](#) [Login](#)

WebGoat Password Recovery

Your username

What is your favorite color?

Congratulations. You have successfully completed the assignment.

Solução:As credenciais são admin e green, jerry e orange, tom e purple, larry e yellow. Passos informativos:



The Problem with Security Questions

While Security Questions may at first seem like a good way to do authentication, they have some big problems.

The "perfect" security question should be hard to crack, but easy to remember. Also the answer needs to be fixed, so it must not be subject to change.

There are only a handful of questions which satisfy these criteria and practically none which apply to anybody.

If you have to pick a security question, we recommend not answering them truthfully.

To further elaborate on the matter, there is a small assignment for you: There is a list of some common security questions down below. If you choose one, it will show to you why the question you picked is not really as good as one may think.

When you have looked at two questions the assignment will be marked as complete.

What is your favorite animal?



check



Creating the password reset link

When creating a password reset link you need to make sure:

- It is a unique link with a random token
- It can only be used once
- The link is only valid for a limited amount of time.

Send a link with a random token means an attacker cannot start a simple DOS attack to your website by starting to block users. The link should not be used more than once which makes it impossible to change the password again. The time out is necessary to restrict the attack window, having a link opens up a lot of possibilities for the attacker.

Assignment

Try to reset the password of Tom (tom@webgoat-cloud.org) to your own choice and login as Tom with that password. Note: it is not possible to use OWASP ZAP for this lesson, also browsers might not work, command line tools like `curl` and the like will be more successful for this attack.

Tom always resets his password immediately after receiving the email with the link.

Account Access

Forgot your password?

Email address you use to log in
to your account

We'll send you an email with
instructions to choose a new
password.

@

Continue

Account Access





How to prevent abusing the password reset function

After learning how to abuse the password reset functionality you should now also know how to defend your own website against such attacks. If you want an extensive description of all mitigation methods take a look here:

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

This lesson will summarize the important points mentioned in the cheat sheet above.

How to use security questions for user verification

Security questions are an easy way to find out information about the validity of the user without asking them for verification data. The problem is, that there are not that many types of security questions and the answers to most of the questions are the same among many users. This makes it easy for an attacker to just guess the question and the answer.

An easy way to make it harder to guess the security question is to let the user themselves decide on the question they want to answer. Further information on this topic can be found here: https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet#Step_1.29_Decide_on_Identity_Data_vs_Canned_Questions_Created_Questions

Sending data over the network

Everything that gets sent via the network in any direction can be read by an attacker. Some data makes it easier for an attacker to compile crucial information on the user's account that helps them bypassing login and password reset restrictions. Therefore try to not send account information (like usernames, e-mails, ...) over the network during the password reset procedure that didn't get entered by the user themselves!

For example: If you send a password reset link to a user via e-mail, do not include the username into the password reset form itself by any means! The user doesn't have to see their username on the form, because a non-malicious user already knows their name. Make it as hard as possible for an attacker to gather further information.

About the password reset token

Password reset tokens allow a user to reset a password without inherently safe information about the verification of the user. Hence they should be safe. It should be hard to guess such a token. The token should also be only valid for a short amount of time and should be invalid after the user successfully reset their password.

Logging user actions

Logging user actions

Logging alone can't prevent any attacks but it can make it easier to determine that an attack happened and how the attacker tried to bypass security. You can also use logs to determine if an account really got hijacked and if it has to be returned to the rightful user. Actions you can log are: How did the security questions get answered? When did the access to the password reset link happen in comparison to the time the e-mail got sent? Were there failed attempts?

Two factor authentication

It is always safer to do an authentication process via two or more separate ways on two or more separate devices. If a user wants to reset their password you can ask them to enter verification codes sent to them via SMS, Messenger, or similar. This makes it hard for an attacker to bypass the verification process, because they need physical access to another device. On the other side it requires the user to give you additional information on the matter of contacting them, which is not welcomed by everyone.

Further reading

We highly recommend to take a further look on the cheat sheet linked in the introduction! The password reset functionality is easily abusable for an attacker when implemented incorrectly. You can make it harder for an attacker to abuse it by just following a few suggestions made here and in the cheat sheet!

4 Pergunta 4.1 - Componentes vulneráveis

Passos Informativos:

Concept

The way we build software has changed. The open source community is maturing and the availability of open source software has become prolific without regard to determining the provenance of the libraries used in our applications. Ref: [Software Supply Chain](#)

This lesson will walk through the difficulties with managing dependent libraries, the risk of not managing those dependencies, and the difficulty in determining if you are at risk.

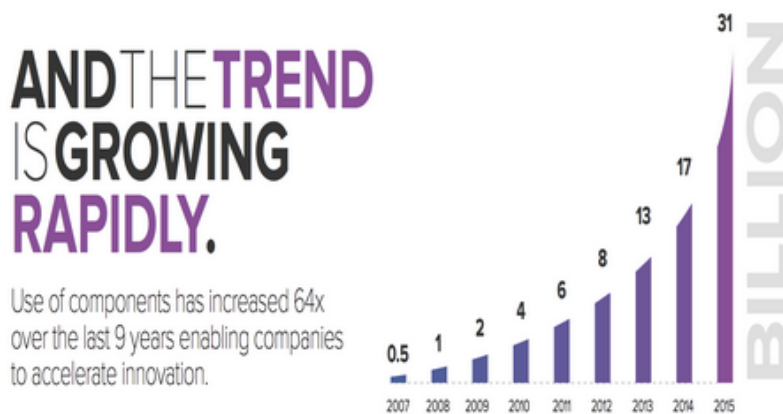


Figure: Software Supply Chain

Goals

- Gain awareness that the open source consumed is as important as your own custom code.
- Gain awareness of the management, or lack of management, in our open source component consumption.
- Understand the importance of a Bill of Materials in determining open source component risk



The Open Source Ecosystems

- 10+ Million GitHub code repositories
- 1 Million Sourceforge code repositories
- 2500 public binary repositories
 - Some repositories have strict publisher standards
 - Some repositories enforce source code distribution
 - No guarantee the published source code is the source code of the published binary
 - Some repositories allow the republishing of a different set of bits for the same version
 - Some repositories allow you to remove published artifacts
- Many different packaging systems; even for the same language
- Different coordinates systems and level of granularity

2013 OWASP Top 10 - A9

As early as 2013, thought leaders like OWASP recognized that "WE" need to pay attention to this problem.

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence WIDESPREAD	Detectability DIFFICULT	Impact MODERATE	Application / Business Specific
Some vulnerable components (e.g., framework libraries) can be identified and exploited with automated tools, expanding the threat agent pool beyond targeted attackers to include chaotic actors.	Attacker identifies a weak component through scanning or manual analysis. He customizes the exploit as needed and executes the attack. It gets more difficult if the used component is deep in the application.	Virtually every application has these issues because most development teams don't focus on ensuring their components/libraries are up to date. In many cases, the developers don't even know all the components they are using, never mind their versions. Component dependencies make things even worse.		The full range of weaknesses is possible, including injection, broken access control, XSS, etc. The impact could range from minimal to complete host takeover and data compromise.	Consider what each vulnerability might mean for the business controlled by the affected application. It could be trivial or it could mean complete compromise.
Am I Vulnerable To 'Using Components with Known Vulnerabilities'? <p>In theory, it ought to be easy to figure out if you are currently using any vulnerable components or libraries. Unfortunately, vulnerability reports for commercial or open source software do not always specify exactly which versions of a component are vulnerable in a standard, searchable way. Further, not all libraries use an understandable version numbering system. Worst of all, not all vulnerabilities are reported to a central clearinghouse that is easy to search, although sites like CVE and NVD are becoming easier to search.</p> <p>Determining if you are vulnerable requires searching these databases, as well as keeping abreast of project mailing lists and announcements for anything that might be a vulnerability. If one of your components does have a vulnerability, you should carefully evaluate whether you are actually vulnerable by checking to see if your code uses the part of the component with the vulnerability and whether the flaw could result in an impact you care about.</p>			How Do I Prevent 'Using Components with Known Vulnerabilities'? <p>One option is not to use components that you didn't write. But that's not very realistic.</p> <p>Most component projects do not create vulnerability patches for old versions. Instead, most simply fix the problem in the next version. So upgrading to these new versions is critical. Software projects should have a process in place to:</p> <ol style="list-style-type: none"> 1. Identify all components and the versions you are using, including all dependencies. (e.g., the versions plugin). 2. Monitor the security of these components in public databases, project mailing lists, and security mailing lists, and keep them up to date. 3. Establish security policies governing component use, such as requiring certain software development practices, passing security tests, and acceptable licenses. 4. Where appropriate, consider adding security wrappers around components to disable unused functionality and/or secure weak or vulnerable aspects of the component. 		
Example Attack Scenarios <p>Component vulnerabilities can cause almost any type of risk imaginable, ranging from the trivial to sophisticated malware designed to target a specific organization. Components almost always run with the full privilege of the application, so flaws in any component can be serious. The following two vulnerable components were downloaded 22m times in 2011.</p> <ul style="list-style-type: none"> • Apache CXF Authentication Bypass – By failing to provide an identity token, attackers could invoke any web service with full permission. (Apache CXF is a services framework, not to be confused with the Apache Application Server.) • Spring Remote Code Execution – Abuse of the Expression Language implementation in Spring allowed attackers to execute arbitrary code, effectively taking over the server. <p>Every application using either of these vulnerable libraries is vulnerable to attack as both of these components are directly accessible by application users. Other vulnerable libraries, used deeper in an application, may be harder to exploit.</p>			References <p>OWASP</p> <ul style="list-style-type: none"> • OWASP Dependency Check (for Java libraries) • OWASP SafeNuGet (for .NET libraries thru NuGet) <p>External</p> <ul style="list-style-type: none"> • The Unfortunate Reality of Insecure Libraries • Open Source Software Security • Addressing Security Concerns in Open Source Components • MITRE Common Vulnerabilities and Exposures • Example Mass Assignment Vulnerability that was fixed in ActiveRecord, a Ruby on Rails GEM 		

Figure: 2013 OWASP - Top 10 - A9

Components are everywhere

WebGoat uses almost **200 Java and JavaScript** libraries. Like most Java applications, we use maven to manage our java dependencies and we employ the wild, wild west strategy for managing JavaScript.

Vulnerable components in WebGoat?

When this lesson was created WebGoat contained more than a dozen high security risks within it's components. Most of these were not deliberate choices. How are developers supposed to track this information across the hundreds of components?

webgoat 8 - 2017-02-08 - Build Report



This report provides security and license assessments for identified components found within an application.

SCOPE OF ANALYSIS



187
COMPONENTS IDENTIFIED
71% OF ALL COMPONENTS ARE IDENTIFIED

4

POLICY ALERTS
AFFECTING 85 COMPONENTS

6

75

14

SECURITY ALERTS
AFFECTING 10 COMPONENTS

80

LICENSE ALERTS

SECURITY ISSUES

How bad are the vulnerabilities and how many are there?

Critical (7-10)

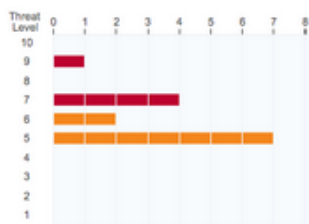
5

Severe (4-6)

9

Moderate (1-3)

0



The summary of security issues demonstrates the breakdown of vulnerabilities based on severity and the threat level it poses to your application.
The dependency depth highlights quantity and severity and distribution within the application's dependencies.

Dependency Depth

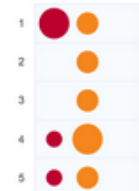


Figure: WebGoat Security Issues

Jquery exploit:

The exploit is not always in "your" code

Below is an example of using the same WebGoat source code, but different versions of the jquery-ui component. One is exploitable; one is not.

jquery-ui:1.10.4

This example allows the user to specify the content of the "closeText" for the jquery-ui dialog. This is an unlikely development scenario, however the jquery-ui dialog (TBD - show exploit link) does not defend against XSS in the button text of the close dialog.

Clicking go will execute a jquery-ui close dialog:

jquery-ui:1.12.0 Not Vulnerable

Using the same WebGoat source code but upgrading the jquery-ui library to a non-vulnerable version eliminates the exploit.

Clicking go will execute a jquery-ui close dialog:

Solução: OK <script>XSS</script>Passos Informativos:



Knowing the OSS "Bill of Materials" is the starting point

Modern applications are comprised of custom code and many pieces of open source. The developer is normally very knowledgeable about their custom code but less familiar with the potential risk of the libraries/components they use. Think of the bill of materials as the list of ingredients in a recipe.

Questions we should know the answer to:

- How do we know what open source components are in our applications?
 - How do we know what versions of open source components we are using?
- How do we define the risk of open source components?
- How do we discover the risk of open source components?
 - How do we associate a specific risk to a specific version of an open source component?
- How do we know when a component releases a new version?
- How do we know if a new vulnerability is found on what was previously a "good" component?
- How do we know if we are using the authentic version of an open source component?

How do I generate a Bill of Materials

There are several open source and paid-for solutions that will identify risk in components. However, there are not many tools that will deliver a complete list of "ingredients" used within an application. OWASP Dependency Check provides the ability to generate a bill of materials and identify potential security risk.

Dependency check uses several pieces of evidence to determine the library names. Below is a snippet of a report:



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help: google group](#) | [github issues](#)

Project: webgoat-8.0.1

Scan Information ([show all](#)):

- dependency-check version: 1.4.4
- Report Generated On: Feb 1, 2017 at 09:18:27 EST
- Dependencies Scanned: 246 (227 unique)
- Vulnerable Dependencies: 21
- Vulnerabilities Found: 142
- Vulnerabilities Suppressed: 0
- ...

Display: [Showing All Dependencies \(click to show less\)](#)

Dependency	CPE	GAV	Highest Severity	CVE Count	CPE Confidence	Evidence Count
webgoat-container-8.0-SNAPSHOT.war		org.owasp.webgoat:webgoat-container:8.0-SNAPSHOT		0		16
webgoat-container-8.0-SNAPSHOT.war:access-control-matrix-1.0.jar		org.owasp.webgoat:lesson:access-control-matrix:1.0		0		11
webgoat-container-8.0-SNAPSHOT.war:back-doors-1.0.jar		org.owasp.webgoat:lesson:back-doors:1.0		0		11
webgoat-container-8.0-SNAPSHOT.war:basic-authentication-1.0.jar		org.owasp.webgoat:lesson:basic-authentication:1.0		0		11

Figure: WebGoat Bill of Materials



Security Information Overload

What's important?

- Is my component exploitable?
- Is my component an authentic copy?
 - Do I understand why my component is modified?

Security information is scattered everywhere

- Multiple sources of security advisories
 - 80,000+ CVEs in the National Vulnerability Database
 - Node Security Project, Metasploit, VulnDB, Snyk, ...
 - Thousands of website security advisories, blogs, tweets, ...
- 600,000 GitHub events generated daily
 - 700 GitHub security related events
 - Release notes, change logs, code comments, ...

Summary

- It is not reasonable to expect a developer to continually research each component.
- Developers are not security experts; they already have a day job.

License Information Overload

What's important?

- Can I use this component within the context of distribution of my software?
- Are there license incompatibilities?
- If using a modified component, did I address additional license obligations?

License information is scattered everywhere

- Projects declare a license:
 - In a project metadata file.
 - On the project website or source code repository page.
 - Using a link to a license file in their own source code repository.
 - In a license file within the project source tree.
 - In the binary META-INF folder.
- Projects include licenses as headers in the source code.

Summary

- It is difficult to determine the scope of a license.
- A project often has license discrepancies.
- Developers are not lawyers .

Architecture Information

What's important?

- Is my component old or is it stable
- Is my component unpopular
- Was my lack of upgrade a deliberate choice or a lack of knowledge

Summary

- It's really difficult to keep components up to date

For the components analyzed in 25,000 applications it was found that:

- 8% of 2 year old components did not have a newer version
- 23% of 11 year old components did not have a newer version
- Older components make up the majority of the risk

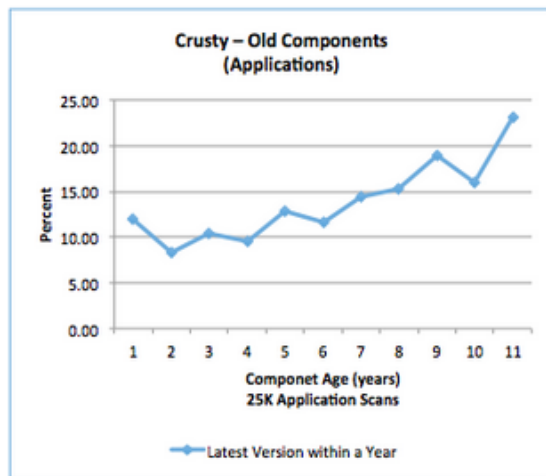


Figure: Old Components

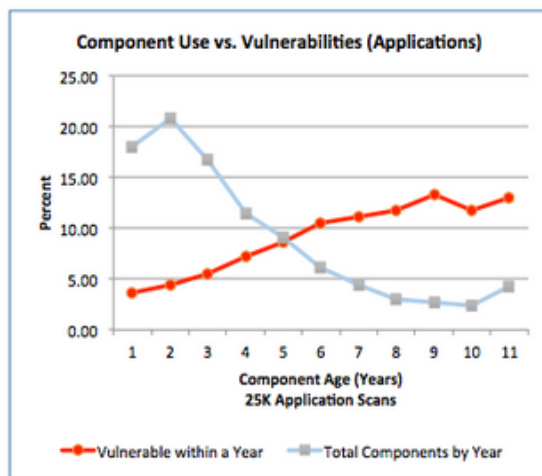


Figure: Risk of Old Components

Some Examples of OSS Risk

Commons Collections

In November of 2015, the Apache Commons Collections component latest release was 8 years old. Commons Collections was considered a reliable and stable component. A researcher found a way to exploit a deserialization issue in Commons Collections resulting in a remote code execution. The next day... **everyone using Commons Collections was in a panic.**

Ref: [Thousands of Java applications vulnerable to nine-month-old remote code execution exploit](#)

Dinis Cruz and Alvaro Munoz exploit of XStream

XStream, a relatively common XML and JSON parsing library, has a nasty little remote code execution.

Ref: [Dinis Cruz Blog](#)
[pwntester/XStreamPOC](#)

You may want to read the article(s) before trying this lesson. Let's see if you can figure out how to exploit this in WebGoat.

XStream exploit:



Exploiting CVE-2013-7285 (XStream)

WebGoat Sends an XML document to add contacts to a contacts database.

```
<contact>
  <id>1</id>
  <firstName>Bruce</firstName>
  <lastName>Mayhew</lastName>
  <email>webgoat@owasp.org</email>
</contact>
```

For this example, we will let you enter the xml directly versus intercepting the request and modifying the data. You provide the XML representation of a contact and WebGoat will convert it a Contact object using

```
xStream.fromXML(xml) .
```

☒

Enter the contact's xml representation:

```
<contact>
  <java.lang.Integer>1</java.lang.Integer>
  <firstName>Bruce</firstName>
  <lastName>Mayhew</lastName>
  <email>webgoat@owasp.org</email>
</contact>
```

Go!

If you are not seeing the application you started; it may be minimized

Solução: <contact >
<java.lang.Integer>1</java.lang.Integer>
<firstName >Bruce </firstName>
<last Name >Mayhew </lastName>
<email>webgoat@owasp.org </email>
</contact>

Passo Informativo:



Summary

- Open source consumption in modern day applications has increased.
- Open source is obtained from many different repositories with different quality standards.
- Security information on vulnerabilities is scattered everywhere.
- License information is often difficult to validate.
- Most teams don't have a component upgrade strategy.
- **Open source components are the new attack vector.**

What to do

- Generate an OSS Bill of Materials.
 - Use [automated tooling](#)
- Baseline open source consumption in your organization.
- Develop an open source component risk management strategy to mitigate current risk and reduce future risk.