PROFERO

SECURITY JOES

GLOBAL THREAT CENTER

Secrets Behind Ever101 Ransomware

Published

June 2021

1. Executive Summary

A victim called the incident response teams of Global Threat Center, reporting a seemingly new stream of ransomware attack. Upon investigation, we determined the extension of the encrypted files was certainly new, but the malware displayed significant similarities with several ransomware families–a combination that made attribution an interesting and difficult riddle. The attack's signature was a Music folder containing an arsenal of tools, which the malware dropped and executed on each of the encrypted machines. Throughout our investigation, we primarily focused on the toolset utilized by the threat actor, in order to build an in-depth profile of the incident in hopes of making an attribution. While many of the tools used by the threat actor were not custom, we were still able to assemble a temporary portfolio of tactics, techniques, and procedures (TTPs), which pointed us to potential links to a few existing ransomware groups with similar TTPs. This portfolio was particularly helpful during the negotiation process, as we were able to gain vital information, such as assessing the reliability of the threat actor in terms of providing a working decryption tool. In fact, during the negotiation, the attackers offered a video documenting the decryption process, which also revealed they used a free software from BandiCam and WinRAR, in what seems to be Arabic.

The ransomware had the extension ".ever101," and was using the *CryptoPP*[8] library (an inbuilt C++ library) for encryption. It utilizes Salsa20 for encrypting file data, and RSA-2048 for encrypting file keys. We confirmed many–but not all–of the tools in the arsenal. Because they were encrypted during the attack, we had little hope of discovering their origin. We were able to establish that the *EVER101* ransomware is almost identical to a number of ransomware families, such as *CURATOR* and *Paymen45*, both of which are believed to be developed by the *EverBe* group. Our hypothesis is that this ransomware was built through a "Ransomware-as-a-Service" builder, rather than being fully developed by the threat actor or group, whose identity and location remain unknown.

During our investigation of the bitcoin movement related to the attack, we made an interesting discovery of a transfer of approximately US$600, to a platform of massage providers across major cities in the United States. This gave us a specific lead to the threat actors, and we developed potential explanations for this questionable transfer.

Table of Contents

2. Technical Details

2.1 Discovered Tools

During our investigation of the infected machines, we came across what seemed to be a treasure trove of information stored in the Music folder. It consisted of the ransomware binary itself, along with several other files—some encrypted, some not—that we believe the threat actors used to gather intelligence and propagate through the network.

2.2 Confirmed

xDedicLogCleaner.exe

xDedicLogCleaner[1], as the name suggests, is used to clear any logs on the system—something that commonly occurs during ransomware attacks. Such tools are used to remove all traces of the attacker on the machine. In this case, however, they did not wipe their tools from one of the infected machines. Instead, most of these tools were encrypted once the ransomware was executed. The ransomware binary was still present on the machine that wasn't wiped, a stroke of good luck that allowed us to expand our investigation.

PH64.exe

PH64.exe is a copy of the *ProcessHacker*[2] binary. *ProcessHacker* allows users to gather information about the system, such as running processes, services, incoming and outgoing network connections, and more. It is quite unusual for it to appear in a ransomware incident. We assume it helped the attackers gain basic information about any anti-malware software running on the machine—because *ProcessHacker* is a legitimate binary, it doesn't trigger monitoring software.

2.3 Unconfirmed

SoftPerfect Network Scanner

This tool is considered unconfirmed because a large majority of the attacker's tools were encrypted with the ransomware. Therefore, the file names were our only lead. In the case of the *SoftPerfect Network Scanner,*[3] we discovered three encrypted files on one of the machines that had similar names to files dropped by the scanner. Specifically:

netscan.exe

netscan.xml
netscan.lic

We correlated these file names with a report[4] produced by the US-Cybersecurity and Infrastructure Security Agency, which covered the Five-Hands Ransomware, along with the tools used in this attack.

shadow.bat

Once again, we discovered this file in an encrypted state on one of the infected machines. While decryption was not possible without a decryptor, we can make assumptions based on the file name, as we found a link between this specific file name and a previous ransomware infection.

A write-up[5] of an unsuccessful Dharma Ransomware deployment by The DFIR Report describes a case in which a .BAT file named *shadow.bat* was deployed. It deleted the shadow files on the machine, using the following command:

```
vssadmin delete shadows /all
```

While there is a strong likelihood the files are the same, it would be surprising, as the ransomware used in our case has built-in function to destroy shadow files. This could indicate that the threat actor possessed limited knowledge of their ransomware variant, or perhaps they used the script as a "backup" method, in the event the ransomware function failed.

NetworkShare_pre2.exe

This is another unconfirmed tool, as it was also encrypted on disk. However, we were able to find a similarly named tool uploaded to an online sandbox provider, where it was executed and analyzed. The discovered tool was labelled as a "NetTool", and in researching the file, we linked it to the same report by The DFIR Report mentioned above. In this incident, the same hash is found, linked to a binary named "*NS.exe*", which is a network enumeration and pivoting tool. This tool is definitely not custom made – it's commonly used by threat actors to enumerate a network for shared folders and connected devices.

There were several other tools on the machine, but because they were overwritten by the ransomware, and had fairly non-descriptive names, we were unable to pinpoint their exact purpose, aside from disabling Windows Defender and clearing more logs.

3. Infection Chain

3.1 Cobalt Strike

Cobalt Strike[6] is an offensive security tool developed for legitimate purposes, though it has become a "go-to" tool for ransomware operators. It provides remote access to infected machines and allows attackers to gather comprehensive information on the system and surrounding network, as well as execute commands, drop files, escalate privileges, exfiltrate important documents, and so on.

In this particular case, the *Cobalt Strike* beacon was hidden inside a binary masquerading as *WEXTRACT.exe*. The binary had a signature that matched the publisher of the original *WEXTRACT.exe*–Microsoft Windows–however, the certificate expired in 2015. Therefore, any certificate checkers would simply identify the expired certificate rather than the modified code.

```
while ( v30 );
v32 = v27 - 1;
if ( (*(_DWORD *)&aShgetpathfromi_0[17] & 0x522CCFFC) != 0 )
  v32 = -v32;
v33 = (char *)v300 + __ROR4__(~v28, 27);
((void (__fastcall *)(int, int, _DWORD, _DWORD, LPCSTR))((char *)&loc_1007CF5 + 1))(v32, v29, v249[0], v249[1], v250);
*(_DWORD *)&aShbrowseforfol_0[11] ^= v34;
v36 = ~v35;
v37 = sub_100379B();
v39 = __ROL4__(v33, 18);
v250 = "KERNEL32.dll";
*(int *)((char *)&dword_100BD4C + 2) -= v39;
v40 = -v39;
v41 = *(_DWORD *)&aShbrowseforfol_0[17] | v36;
v42 = *(_DWORD *)&aDefaultinstall[1] | v38 ^ 0x2E8A6160;
((void (__thiscall *)(int))nullsub_3)(v41 - v37);
*(int *)((char *)&dword_100BD64 + 1) |= v40;
v43 = ((__int64 (*)(void))((char *)&loc_1002A0F + 1))();
dword_100BC1B -= v44;
v45 = dword_100A2C7 ^ __ROL4__(v41 - v40, 11);
dword_100BCF7 ^= v43;
nullsub_14(v44, HIDWORD(v43));
dword_100BC0B = v40;
LoadLibraryA(v250);
((void (*)(void))((char *)&loc_1005593 + 1))();
nullsub_11();
v46 = dword_100BE7E + __ROL4__(v40, 9);
v47 = ((__int64 (__stdcall *)(int, int, int))((char *)&loc_10082E7 + 1))(v251, v252, v253);
dword_100BDB8 |= v45;
v48 = __ROL4__(v45, 19);
dword_100BC63 |= v48;
*(int *)((char *)&dword_100BDAD + 1) ^= 0xB943809E;
v50 = dword_100BE72 + v46;
*(int *)((char *)&dword_100BD85 + 2) = ((v42 - 1) ^ 0xD9BA1A1E) + 887509181;
v51 = -*(int *)((char *)&dword_100BD85 + 2);
((void (__fastcall *)(int, int, int))((char *)&loc_1003AE4 + 1))(
  *(int *)((char *)&dword_100BF0F + 1) ^ v49,
  (dword_100BED3 | HIDWORD(v47)) - *(int *)((char *)&dword_100BDF9 + 2),
  v254);
v52 = ((int (__stdcall *)(int, int, int))((char *)&loc_1007CF5 + 1))(v255, v256, v257);
*(int *)((char *)&dword_100BC5B + 3) &= v50;
dword_100BCA2 &= 0xA1CFC18F;
dword_100BC54 = v48;
```

Snippet of heavily obfuscated code present in modified WEXTRACT.exe

This method of hiding a beacon is quite interesting: before this case, there is no documentation of its use. It is also not mentioned in the *Cobalt Strike* documentation. This perhaps indicates the presence of a public "builder" who implants obfuscated code into a legitimate binary, which decodes and executes a beacon in the device's memory, or the possibility of a custom tool developed by the ransomware actors. Either way, this knowledge will be hugely beneficial to us in future investigations, as we will be able to leverage this

technique to create rules for further threat intelligence gathering on the threat actors behind this infection.

3.2 SystemBC

SystemBC[7] is a well-known proxy malware used to route incoming and outgoing connections through SOCKS5, during attempts to hide communications between a malware implant and a command and control server. Additionally, it can be used to proxy internal connections. *SystemBC* is a fairly old tool, but it's becoming increasingly popular among ransomware actors as a method of hiding *Cobalt Strike* beacon traffic from network traffic analyzers.

```
; Segment type: Pure data
; Segment permissions: Read/Write
_data           segment para public 'DATA' use32
                assume cs:_data
                ;org 405000h
systemBC_config db 'BEGINDATA',0        ; DATA XREF: sub_402BBF+4↑o
                                        ; sub_402BBF+67↑o
aHost192539084  db 'HOST1:92.53.90.84',0
                                        ; DATA XREF: sub_401549+38↑o
                                        ; sub_401549+7C↑o ...
```

Configuration present in SystemBC

This tool, along with *Cobalt Strike*, was discovered in the Windows Startup folder, which is a common method of persistence. Upon system start-up, any programs within the folder will be immediately executed even without user interaction. This method is fairly simple and extremely easy to detect–raising questions about the aptitude of the threat actors.

3.3 Ransomware Analysis

3.3.1 Capabilities Overview

The Ransomware itself is fairly basic. As mentioned above, it utilizes the *CryptoPP*[8] library (an inbuilt C++ library) for encryption, Salsa20 for encrypting file data, and RSA-2048 for encrypting file keys. This usage of asymmetric algorithms makes decryption of the files impossible without the attackers' RSA-2048 private key. This private key decrypts the encrypted Salsa20 file keys, which in turn enables decryption of the file data.

It also has capabilities allowing it to connect to remote servers, and send information about the system, such as the number of files and disks and data related to the CPU and RAM. In this particular case, a URL was not present in the binary, meaning the malware operated

entirely offline. This is probably due to the fact that *Cobalt Strike* was already on the machine prior to the ransomware roll-out.

```
                ; const WCHAR aSDelimiterName
                aSDelimiterName:                        ; DATA XREF: post_data+147↑o
25 00 73 00+                    text "UTF-16LE", '%s|DELIMITER|Name(domain): %s(%s)',0Dh,0Ah
7C 00 44 00+                    text "UTF-16LE", 'CPU: %S',0Dh,0Ah
45 00 4C 00+                    text "UTF-16LE", 'RAM: %d',0Dh,0Ah
49 00 4D 00+                    text "UTF-16LE", 'Disks count: %d',0Dh,0Ah
49 00 54 00+                    text "UTF-16LE", 'Files count: %d|DELIMITER|',0
00 00 00 00+                    align 8
```

Template used for sending data to a remote server

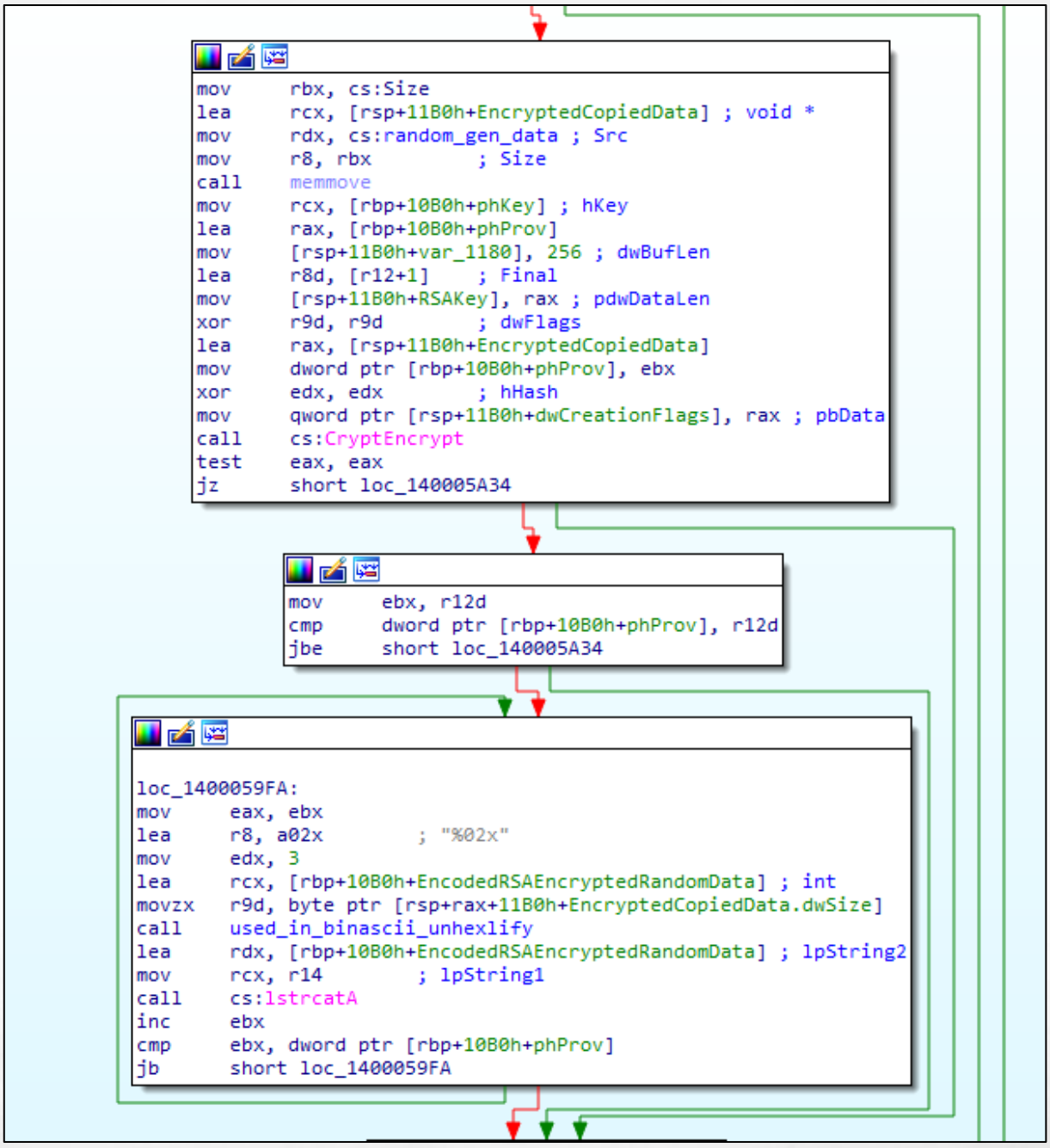
The ransomware is not only able to send information to remote servers, it can also download the Tor browser to gather information from a .ONION site that is most likely hosted by the threat actors. As no URL was present in the sample, we cannot confirm all its capabilities, though it does seem the downloading functionality is used to download a public RSA-2048 key, possibly to prevent the entry of any hardcoded public keys into the file.

Internet Accessibility inside ransomware

Finally, as mentioned above, the ransomware has the ability to clear shadow files, preventing any attempts to backup. The command to execute occurs through Windows API, and is stored in an encoded state before being decoded and executed. The encoding itself was custom developed, rather than originating from well-known algorithms, which enabled us to progress our investigation of further threats.

3.3.2 Algorithm Analysis

Because the ransomware uses RSA-2048 and Salsa20 to perform file encryption, a public RSA key is hardcoded into the binary, and stored in a hexadecimal encoded format–an example of which can be seen below. This RSA key is stored as a Microsoft *PUBLICKEYBLOB*[9], allowing it to be imported using Windows API.



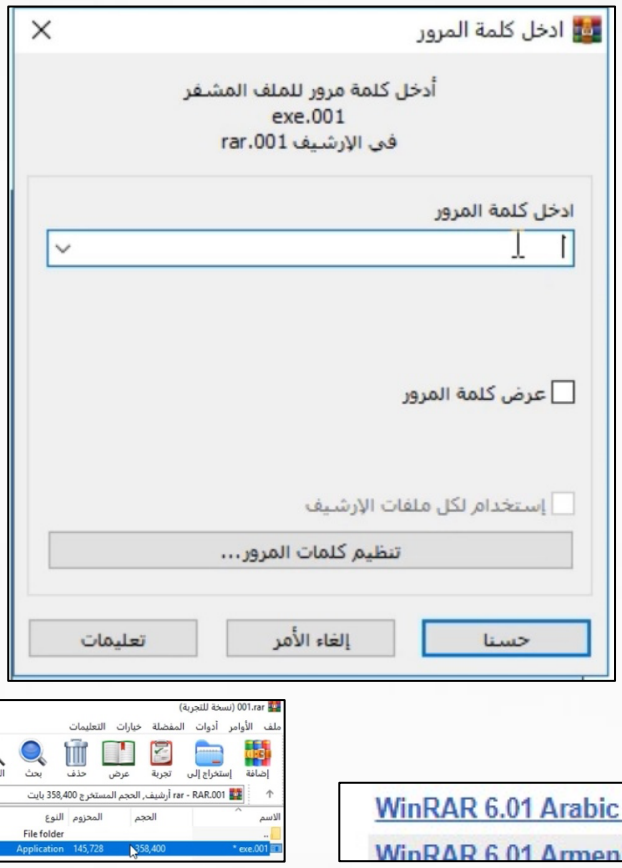
RSA-2048 encryption and hexadecimal encoding of generated Salsa20 key

Unlike several other ransomware variants, this variant generates only a single Salsa20 key on each run. This means all files on a particular system will be encrypted using the same Salsa20 key. The key is generated through several calls in the *CryptoPP* library, alongside the CryptGenRandom() call. Once generated, it is immediately encrypted with the hardcoded RSA-2048 key, and then converted into a hexadecimal format–similar to the format of the stored RSA key. This newly created hexadecimal blob is then appended to all encrypted files, allowing the decryption tool to extract the last 512 bytes of a file, decrypt it using the attackers' RSA private key, and gain the file's Salsa20 key.

4. Threat Intelligence

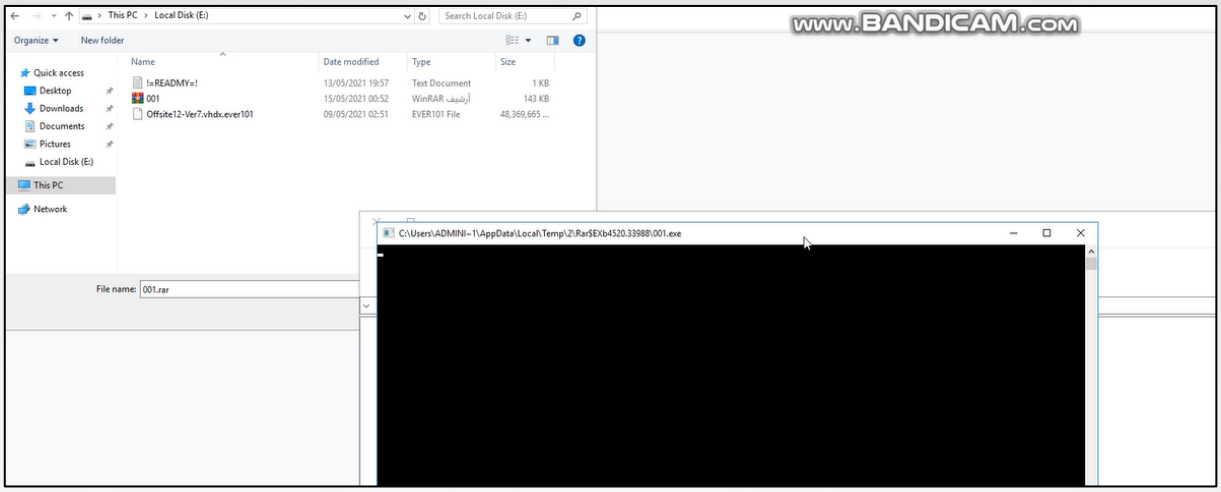
4.1 Decryption Tutorial

After requesting proof-of-concept for the decryption process, the attackers did something unexpected. They sent an mp4 file with the name “LOOK!” containing a short video (1:29 min) of how the files are decrypted. The tutorial contains a few interesting details, including the fact that they chose to use WinRaR in Arabic. Although it does not offer a specific lead, it is worth mentioning as their language is one of the few hints we have for attribution.



Threat actor using WinRaR in Arabic

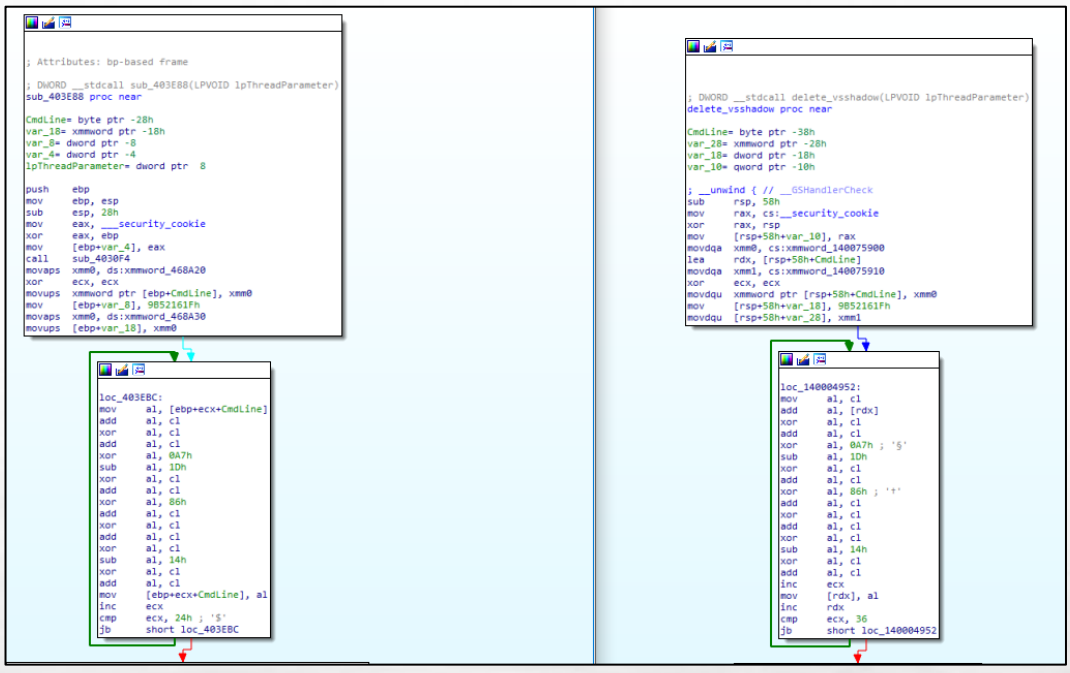
In addition, the video was captured using a free software called BandiCam. This software was spotted on other occasions when attackers used it to record tutorials.



Threat Actor using BandiCam free software for video tutorial

4.2 Code Similarity

During our investigation, we noticed the encoding algorithm for the command to delete shadow files did not match any well-known encoding algorithms, leading us to look into its origins. We were able to locate this algorithm in a number of files, all red-flagged as ransomware. We reverse-engineered these files, and confirmed that the *EVER101* ransomware is almost identical to a number of ransomware families, most especially *CURATOR* and *Paymen45*. Therefore, we feel confident in concluding that this particular ransomware was most likely built through a "Ransomware-as-a-Service" builder, rather than being fully developed by the threat actors.



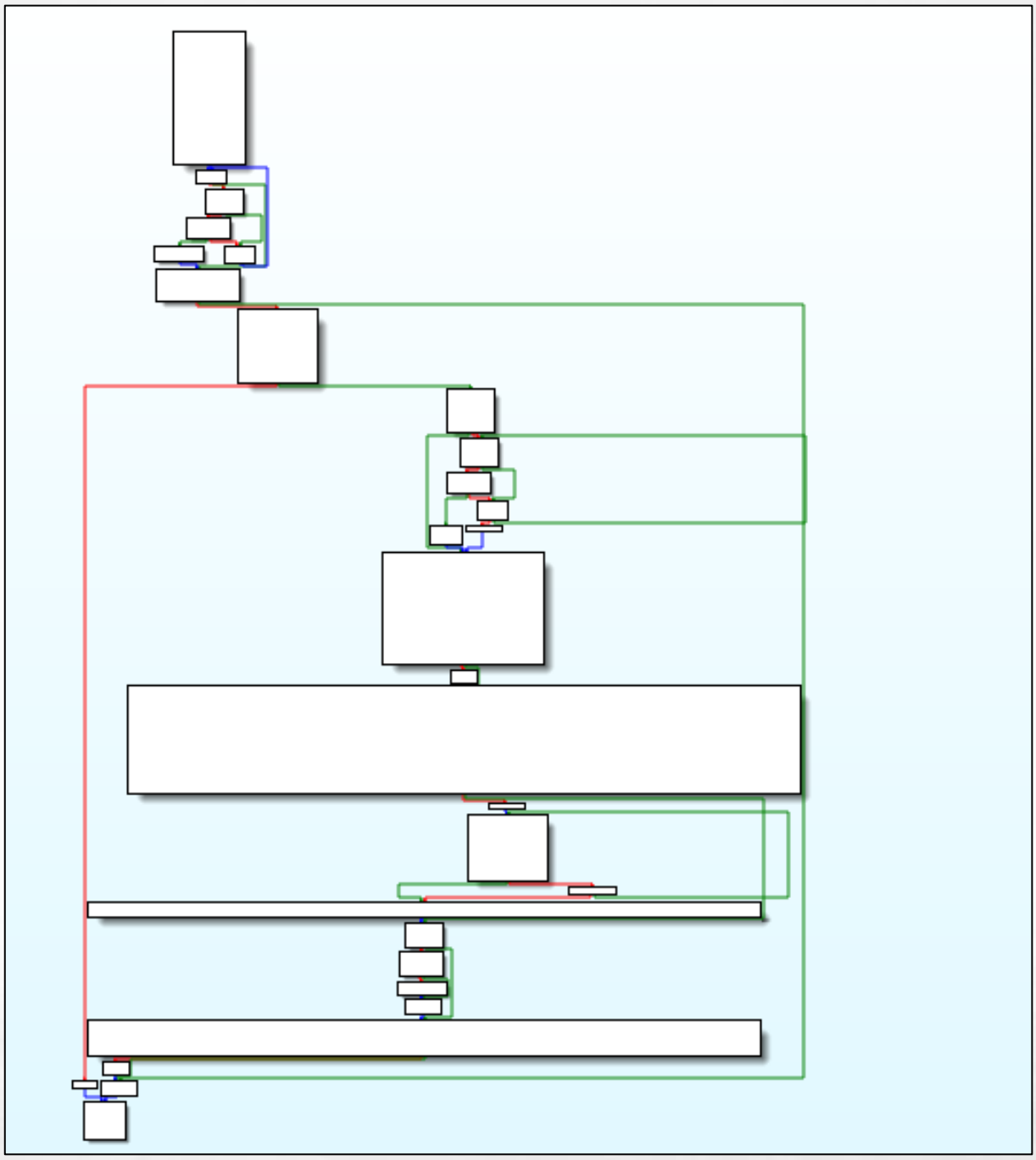
```
; Attributes: bp-based frame

; DWORD __stdcall sub_403E88(LPVOID lpThreadParameter)
sub_403E88 proc near

CmdLine= byte ptr -28h
var_18= xmmword ptr -18h
var_8= dword ptr -8
var_4= dword ptr -4
lpThreadParameter= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 28h
mov     eax, ___security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
call    sub_4030F4
movaps  xmm0, ds:xmmword_468A20
xor     ecx, ecx
movups  xmmword ptr [ebp+CmdLine], xmm0
mov     [ebp+var_8], 9B52161Fh
movaps  xmm0, ds:xmmword_468A30
movups  [ebp+var_18], xmm0

loc_403EBC:
mov     al, [ebp+ecx+CmdLine]
add     al, cl
xor     al, cl
add     al, cl
xor     al, 0A7h
sub     al, 1Dh
xor     al, cl
add     al, cl
xor     al, 86h
add     al, cl
xor     al, cl
add     al, cl
xor     al, cl
sub     al, 14h
xor     al, cl
add     al, cl
mov     [ebp+ecx+CmdLine], al
inc     ecx
cmp     ecx, 24h ; '$'
jb      short loc_403EBC
```

```
; DWORD __stdcall delete_vsshadow(LPVOID lpThreadParameter)
delete_vsshadow proc near

CmdLine= byte ptr -38h
var_28= xmmword ptr -28h
var_18= dword ptr -18h
var_10= qword ptr -10h

; __unwind { // __GSHandlerCheck
sub     rsp, 58h
mov     rax, cs:__security_cookie
xor     rax, rsp
mov     [rsp+58h+var_10], rax
movdqa  xmm0, cs:xmmword_140075900
lea     rdx, [rsp+58h+CmdLine]
movdqa  xmm1, cs:xmmword_140075910
xor     ecx, ecx
movdqu  xmmword ptr [rsp+58h+CmdLine], xmm0
mov     [rsp+58h+var_18], 9B52161Fh
movdqu  [rsp+58h+var_28], xmm1

loc_140004952:
mov     al, cl
add     al, [rdx]
xor     al, cl
add     al, cl
xor     al, 0A7h ; '§'
sub     al, 1Dh
xor     al, cl
add     al, cl
xor     al, 86h ; '†'
add     al, cl
xor     al, cl
add     al, cl
xor     al, cl
sub     al, 14h
xor     al, cl
add     al, cl
inc     ecx
mov     [rdx], al
inc     rdx
cmp     ecx, 36
jb      short loc_140004952
```

Left: Paymen45 *encoding algorithm* *Right:* EVER101 *encoding algorithm*

5. Decryptor Analysis

Once the decryptor was received, we focused our efforts on analyzing whether it functioned as the threat actors promised–or whether it contained anything malicious. We discovered that the threat actors, in a surprising move, used multiple RSA public-private key pairs across the infected machines, which we were unable to retrieve based on our investigation of the one recovered sample.



Overview of decryption function present in decryption tool

The decryptor itself is simply a reversed version of the ransomware. It uses the on-board RSA-2048 private key (stored in a section labelled *.INFO*) to decrypt any files that match a hardcoded specification (appended with the *.EVER101* extension), which then decrypt the hexadecimal blob appended to the file, in order to obtain the Salsa20 file key. As Salsa20 is a symmetric algorithm, the same key used for encryption is used for decryption. From there, it will overwrite the encrypted file with the decrypted data, and move onto the next file. Additionally, it deletes all files with the name *!=READMY=!.txt*, thus removing any evidence of the prior ransomware infection.

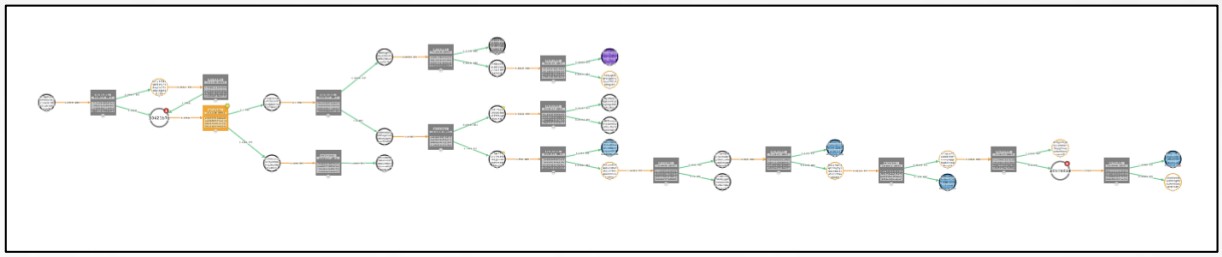
```
; Segment type: Pure data
; Segment permissions: Read
_INFO           segment para public 'DATA' use64
                assume cs:_INFO
                ;org 14005E000h
privateRSAKey   db 'ad1b7d8f4755f386f31c675fe522b239367db6bbaff2a191160f93442db38dfbb'
                db 'c97bc2cbd669f1911b0e0c5683af80c7cb4e35646af29278853d7bdf68eb002a2'
                db 'db5e69486d92f38e8bfddcde32efa88da04b6433bce7204e3aefb386c54bc1eb9'
                db 'c5edbf28c810b63f396469bb5f7f4946b21bc73172c26aaf67f45cfb8c980d5ae'
                db 'af51e9dfc3124da27381d316c9bd70085df5489f4396581cb6072b7b4fbc55c78'
                db '4fbd26d05b29dba8b92f047e14013174ef4722007e5edb66422b1c1d5d80bcd7f'
                db 'de38aea608809e29fb5feba9df6593d6ac8ee4ddc4f34b1f9294c7b344c864272'
                db '9032531cfa7e9faaec5c41fd52bbd7c830d37ff635625429a6cb50536d011b2c9'
                db '53fdd2157b04e9704e97eb6c213897181058d6dfd786cf9298ef9ea7'
aEver101:
                text "UTF-16LE", '.ever101',0
                db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                db 0,0,0,0,0,0,0,0,0,0,0,0,0,0
aReadmyTxt:
                text "UTF-16LE", '!=READMY=!.txt',0
```

Hardcoded RSA Private Key inside .INFO section

Comparison of the received decryption tools indicated that the only change across all binaries was the *.INFO* section, which contains three basic pieces of information: the RSA-2048 private key, the file extension to be identified, and the ransom note file name. This further indicates the likely usage of a ransomware builder by the threat actors, as this information could all be incorporated in the main code base, rather than being appended to the binary—which is highly unusual.

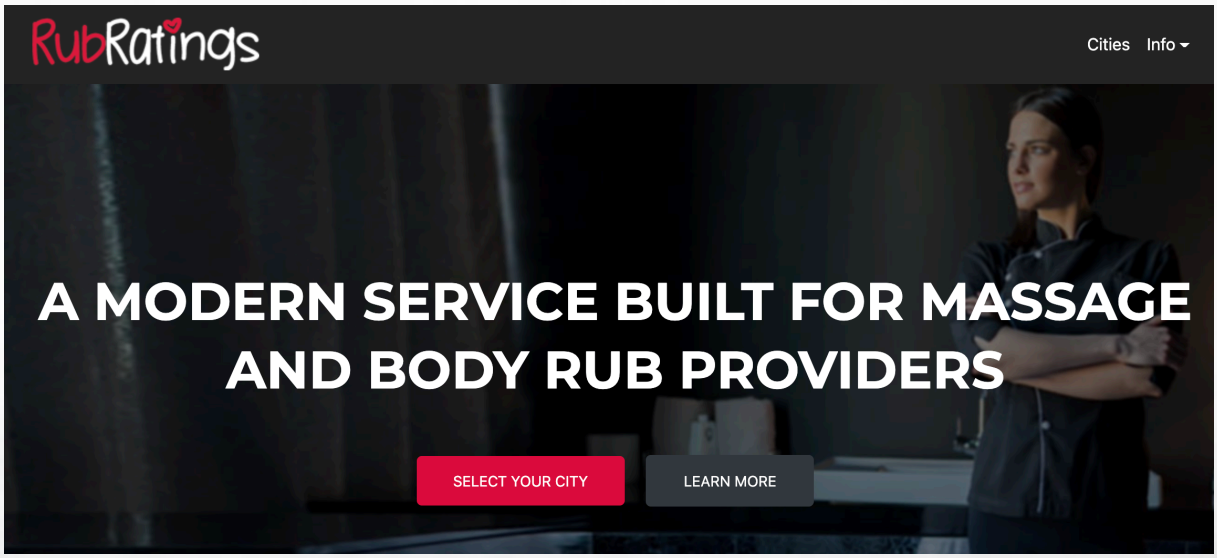
6. Bitcoin Tracing

Once the ransom was paid, we continued tracing the movement of the bitcoin, using CipherTrace, to investigate where the coins were flowing to. In general, the movement did not seem sophisticated, unlike prior cases in which several services were used to convert one currency to another. However, we did make one fascinating discovery.



Bitcoin Movement as shown by CipherTrace

One of the branches in the CipherTrace graph indicate that on May 18, 2020, 0.01378880 BTC was transferred to a wallet (around US$590) linked to a site called RubRatings. The site describes itself as a "platform of providers offering body rubs and sensual massages to clients in major cities across the US." The wallet in question seems to link to a Tip Jar functionality available on the site, which provides a method for clients to send bitcoin to specific individuals.



This discovery suggests of one of two possibilities. The first is that the threat actor(s) are new to the game, and they thought the bitcoin had been hidden successfully. As a result, they decided to pay for services provided through the site, through the Tip Jar functionality or through direct communication with the provider. This would indicate that the attackers are US-based, as the site only provides services to clients in American cities.

The second possibility is that the provider on the site was used as another method of obfuscating the bitcoin movement. It could be that the provider who possesses the bitcoin wallet in question was working with the threat actor(s), but more likely, it is a fake account set up to enable money transfers. The bitcoin in the wallet linked to RubRatings received the payment around 15:48 UTC, and it left the wallet just a few minutes later, at 15:51 UTC.

As of now, we cannot be sure of the processes that individual tips follow, whether the wallets are linked to the site operator and are then distributed to the provider in question, or if it was the provider who moved it out quickly. Regardless, it is an extremely interesting discovery, and we believe this case to be the first of its kind.

7. IOCs & Yara rules

Binary IOCs (MD5):

Ransomware Binary:

RA64.exe: ea504e669073d9e506fb403e633a68c8

Analysed Decryption Tool:

10.10.10.7.exe: 55bd82c389e69098774b5a500aa0b316

Tools:

xDedicLogCleaner.exe: 0f34ab1e2166cada2be7c551e026507c
PH64.exe: b365af317ae730a67c936f21432b9c71

Cobalt Strike:

winhlp_32.exe: d1dd5ffa647734fdcf784dfbc9ffc90d

SystemBC:

VmManagedSetup.exe: 383a80304cc43365619d7e20b9d54d56

SystemBC YARA Rule

```
import "pe"

rule vm_managed_systembc_rule {
  meta:
    description = "exe - file VmManagedSetup.exe"
    author = "Daniel Bunce of Security Joes & Profero"
    hash = "299894D56BE26CA93049278489512 35C61322FEF"
    reference =
"https://www.virustotal.com/gui/file/2f90da6517ba31d42cd907480ded408e711761
fb727c89baef821e040485365a/community"

  strings:

    $str_config_marker = "BEGINDATA" fullword ascii
    $str_hardcoded_ip = "92.53.90.84" fullword ascii
    $str_execute_ps1 = "-WindowStyle Hidden -ep bypass -file" fullword
ascii

    $compare_xordata = { 81 3D ?? ?? ?? ?? 78 6F 72 64 ?? ?? 81 3D ?? ?? ??
?? 61 74 61 00 }

  condition:

    uint16(0) == 0x5a4d and 3 of them

}
```

Cobalt Strike YARA Rule (may yield some False Positives)

```
import "pe"

rule winhlp_cobaltstrike_rule {
  meta:
    description = "exe - file winhlp_32.exe"
    author = "Daniel Bunce of Security Joes & Profero"
    hash = "DBA5A62139B439E23DEAF854A0FEA9973BEBB33E"
    reference = "Possible weak link in terms of technique:
https://blog.talosintelligence.com/2017/04/threat-roundup-0421-0428.html"
    notes = "Basically a modified WEXTRACT.EXE binary, attackers have
overwritten a function to decrypt and execute their cobalt stager"

  strings:

    $str_rundll_cmd = "rundll32.exe %s,InstallHinfSection %s 128 %s"
fullword ascii
    $str_runonce_reg =
"Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce" fullword ascii
    $str_cmd_line = "Command.com /c %s" fullword ascii
    $str_filename = "msdownld.tmp" fullword ascii
    $str_loadstring_error = "LoadString() Error.  Could not load string
resource." fullword ascii
    $str_pdb = "wextract.pdb" fullword ascii

  condition:

    uint16(0) == 0x5a4d and all of them and
    for any i in (0 .. pe.number_of_signatures) : (
      pe.signatures[i].issuer contains "Microsoft Windows" and
```

```
      pe.signatures[i].serial ==
"33:00:00:00:4e:a1:d8:07:70:a9:bb:e9:44:00:00:00:00:00:4e"
    )

}
```

EVER101 Ransomware YARA Rule

```
import "pe"

rule ever101_paymen45_rule {
  meta:
    description = "exe - file 64RA.exe"
    author = "Daniel Bunce of Security Joes & Profero"
    hash = "32EDA62ED3B0E642072079DE2FFDDF686A5783A0"
    reference = "https://0xsakura.me/paymen45-ransomware-analysis-and-
developing-decryptor/"
  strings:

    $str_extension = ".ever101" fullword wide
    $str_ransom_placeholder = "{KEY11111}" fullword wide
    $str_post_data = {25 00 73 00 7c 00 44 00 45 00 4c 00 49 00 4d 00 49 00
54 00 45 00 52 00 7c 00 4e 00 61 00 6d 00 65 00 28 00 64 00 6f 00 6d 00 61
00 69 00 6e 00 29 00 3a 00 20 00 25 00 73 00 28 00 25 00 73 00 29 00 0d 00
0a 00 43 00 50 00 55 00 3a 00 20 00 25 00 53 00 0d 00 0a 00 52 00 41 00 4d
00 3a 00 20 00 25 00 64 00 0d 00 0a 00 44 00 69 00 73 00 6b 00 73 00 20 00
63 00 6f 00 75 00 6e 00 74 00 3a 00 20 00 25 00 64 00 0d 00 0a 00 46 00 69
00 6c 00 65 00 73 00 20 00 63 00 6f 00 75 00 6e 00 74 00 3a 00 20 00 25 00
64 00 7c 00 44 00 45 00 4c 00 49 00 4d 00 49 00 54 00 45 00 52 00 7c}
    $str_tor_link = "https://dist.torproject.org/torbrowser/8.5.3/tor-
win32-0.3.5.8.zip" fullword wide
    $str_error_msg = "RandomNumberGenerator: IncorporateEntropy not
implemented" fullword ascii

    $string_decode_x64 = { 8A C1 02 02 32 C1 02 C1 34 A7 2C 1D 32 C1 02 C1
34 86 02 C1 32 C1 02 C1 32 C1 2C 14 32 C1 02 C1 FF C1 }
    $encoded_vssadmin = { 8E 84 AF A8 B8 AA AD 49 00 51 3D 3F 27 2D 59 EF }

  condition:

    uint16(0) == 0x5a4d and 3 of ($str_*) and $string_decode_x64 or
$encoded_vssadmin

}
```

8. References

[1] https://app.any.run/tasks/736caba7-a5a5-4cc2-8d7d-216124e1a4df/

[2] https://processhacker.sourceforge.io/

[3] https://www.softperfect.com/products/networkscanner/

[4] https://us-cert.cisa.gov/ncas/analysis-reports/ar21-126a

[5] https://thedfirreport.com/2020/06/16/the-little-ransomware-that-couldnt-dharma/

[6] https://www.cobaltstrike.com/

[7] https://www.proofpoint.com/us/threat-insight/post/systembc-christmas-july-socks5-malware-and-exploit-kits

[8] https://github.com/weidai11/cryptopp

[9] https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/ns-wincrypt-publickeystruc

[10] https://www.pcrisk.com/removal-guides/17725-paymen45-ransomware