CopyRh(ight)adamantys Campaign: Rhadamantys Exploits Intellectual Property Infringement Baits

research.checkpoint.com/2024/massive-phishing-campaign-deploys-latest-rhadamanthys-version/

November 6, 2024

Key findings

- Check Point Research is tracking an ongoing, large scale and sophisticated phishing campaign deploying the newest version of the Rhadamanthys stealer (0.7). We dubbed this campaign CopyRh(ight)adamantys.
- This campaign utilizes a copyright infringement theme to target various regions, including the United States, Europe, East Asia, and South America.
- The campaign impersonates dozens of companies, while each email is sent to a specific targeted entity from a different Gmail account, adapting the impersonated company and the language per targeted entity. Almost 70% of the impersonated companies are from Entertainment /Media and Technology/Software sectors.
- Analysis of the lures and targets in this campaign suggests the threat actor uses automation for lures distribution. Due to the scale of the campaign and the variety of the lures and sender emails, there is a possibility that the threat actor also utilized AI tools.
- One of the main updates in the Rhadamanthys stealer version according to claims by the author, is AI-powered text recognition. However, we discovered that the component introduced by Rhadamanthys does not incorporate any of the modern AI engines, but instead uses much older classic machine learning, typical for OCR software.

While we finalized this blog post, a technical analysis of this activity was published by fellow researchers from Cisco Talos. While it overlaps with our findings to some extent, our report provides additional extended information about the activity.

Introduction

Since July 2024, Check Point Research (CPR) has been tracking an extensive and ongoing phishing campaign that leads to the deployment of the Rhadamanthys stealer. This campaign masquerades as various companies and falsely claims that victims have committed copyright infringement related on their Facebook pages.

The phishing emails, typically sent from Gmail accounts, prompt recipients to download an archive file, which triggers the infection through DLL side-loading. The vulnerable binary then installs the latest version of the Rhadamanthys stealer (version 0.7), which includes new capabilities such as an alleged AI-powered OCR (optical character recognition) module.

In this report, we share our ongoing efforts to study the use of the Rhadamanthys stealer, which both cybercriminals and state-sponsored actors have adopted. We provide an in-depth examination of the phishing campaign, the tactics used by the attackers, and the updates introduced in this latest version of Rhadamanthys.

Background

Throughout 2024, we have been monitoring threat actors' activities leveraging the Rhadamanthys stealer, including its use by Void Manticore, an Iranian actor operating in Israel and Albania. In one campaign tied to **Handala,** a persona linked to Void Manticore, the Rhadamanthys stealer was distributed under the guise of a F5 update. This marked their first use of the stealer, which they continued to deploy in subsequent campaigns impersonating Israeli and international companies.

Simultaneously, Check Point Software Technologies began receiving reports of phishing lures mimicking Check Point- branded emails leading to the deployment of Rhadamanthys. Given Handala's previous interest in Check Point and threats they published in their Telegram channel, our initial assumption was that they were also behind this campaign. However, further analysis revealed this was merely a coincidence and the Check Point lures were part of a larger, distinct cybercrime-oriented cluster, which we explore in detail.

Figure 1 - Phishing email impersonating Check Point

CopyRh(ight)adamantys Emails

The newly identified cluster is characterized by spear-phishing emails sent from Gmail accounts allegedly from well-known companies claiming supposed copyright violations. These emails, which appear to come from the legal representatives of the impersonated companies, accuse the recipient of misusing their brand on the target's social media page and requesting the removal of specific images and videos.

The removal instructions are said to be in a password-protected file. However, the attached file is a download link to `appspot.com`, linked to the Gmail account, which redirects the user to Dropbox or Discord to download a password-protected archive (with the password provided in the email).

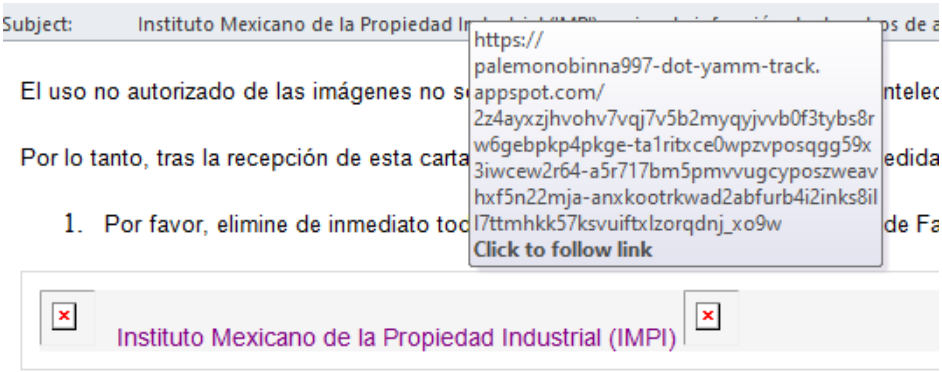

Figure 2 – Malicious ZIP download link.

We observed hundreds of emails impersonating dozens of companies, each sent to a specific address from a different Gmail account. Almost 70% of the impersonated companies are from Entertainment /Media and Technology/Software sectors. This is possibly due to the fact that those sectors have a high online presence and are more likely to send such requests than other sectors. These high profile sectors also have frequent copyright-related communications, making such phishing attempts appear more credible.

The attackers likely used an automated tool, possibly with AI integration, to generate both the emails and the accounts. While most emails are written in the recipient's local language or English, occasional errors occur. For example, one email intended for an Israeli target was written in Korean instead of Hebrew, with only the target's name correctly localized.

Figure 3 – Phishing email written in Korean mistakenly sent to a target in Israel.

Infection Chain

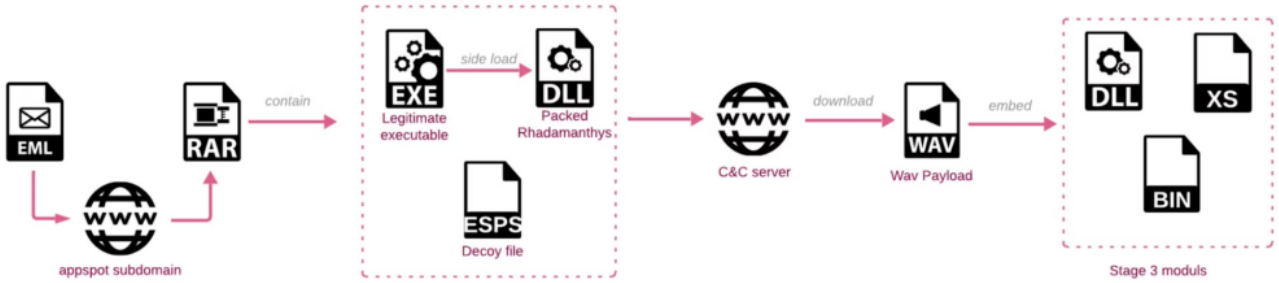

Figure 4 – Copyright campaign infection chain.

As we stated, the infection begins with a spear-phishing email containing a link to download a password-protected archive. This archive typically includes three files: a legitimate executable, a DLL (which contains the packed Rhadamanthys), and a decoy Adobe ESPS or PDF file. When the executable is run, it utilizes DLL sideloading to load the malicious DLL, which subsequently unpacks and loads the Rhadamanthys components.

The legitimate executables and the names given to the DLL for sideloading:

Legitimate Executable (Often renamed)	Sideloaded DLL
Launcher.exe	msimg32.dll
AcroLicApp.exe	msimg32.dll
AdobeARM.exe	SensApi.dll

Once active, the stealer writes a significantly larger copy of the DLL into the `Documents` folder, masquerading as a Firefox-related component (`FirefoxData.dll`). It also creates a registry key for persistence:

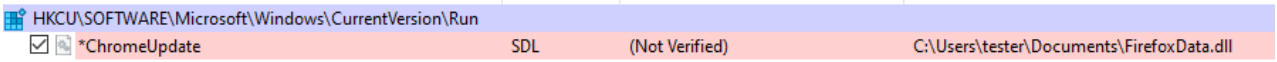

Figure 5 – Registry key added for persistence.

The only difference between the dropped DLL and the one from the initial package is an appended empty overlay. This is a simple trick intended to evade hash-based detection, as the random padding changes the original hash of the executable. Sometimes, the enlarged size of the file may also cross the acceptable file size threshold defined by a particular antivirus engine, and as a result, the file is not scanned.

As the Rhadamanthys modules are loaded, they are injected into one of the following processes from the system32 directory:

- credwiz.exe
- OOBE-Maintenance.exe
- openwith.exe
- dllhost.exe
- rundll32.exe

The process of loading Rhadamanthys modules and their general flow did not change much since the last version, 0.5.0, that we described in detail in our previous report. The initial Rhadamanthys executable has a hardcoded package from which Stage 2 is unpacked.

The role of Stage 2 is to run extensive evasion checks on the compromised machine, connect to the Command-and-Control server (C2), and download the next package which contains Stage 3. Stage 3, shipped steganographically in a WAV file, is a rich set of stealer modules that attack various targets. We described most of these modules in our previous article.

A complete list of Rhadamanthys Stages 2 and 3 is available in Appendix A.

Targets

The campaign's targets are distributed across a wide geographic area, including the US, Europe, the Middle East, East Asia, and South America. However, it's important to note that our target observations are limited by our customers' that were targeted by this campaign. We believe this is part of a much larger campaign, with likely many more countries affected than we've seen.

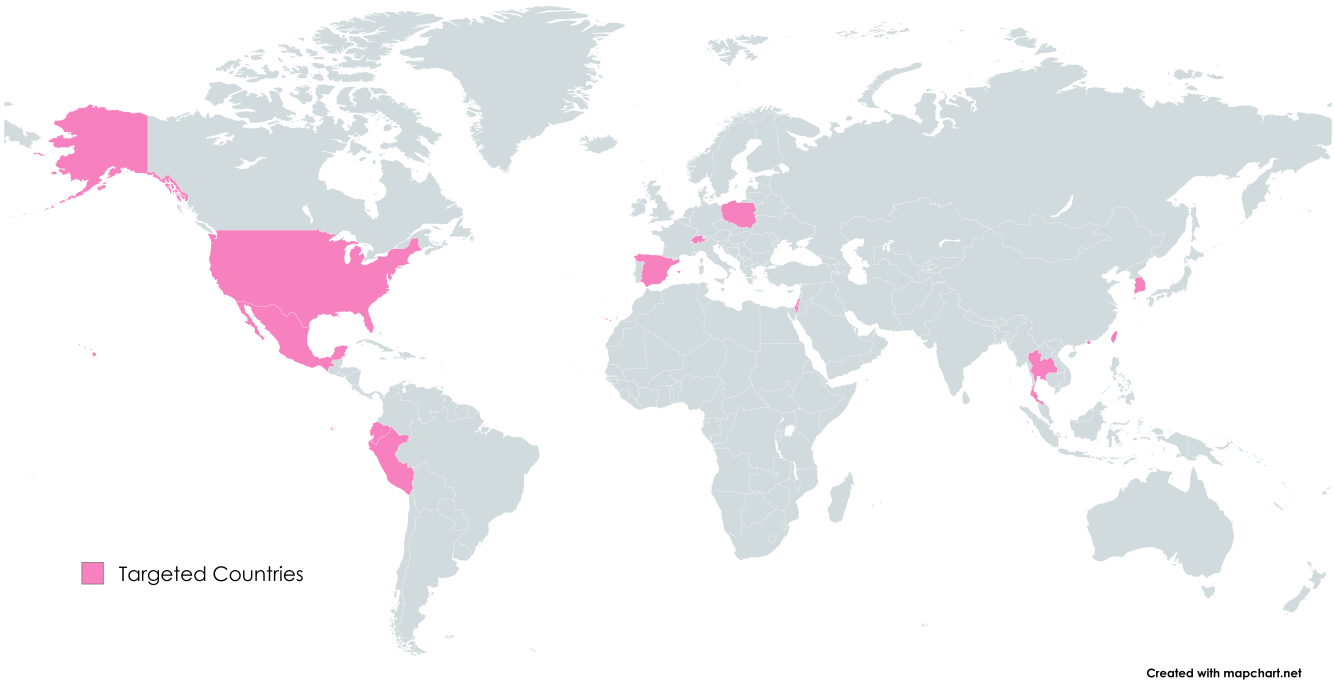

Figure 6 – Map of targeted countries according to Check Point's telemetry.

Attribution

Although Rhadamanthys was previously linked to nation-state threat actors like those from Russia or Iran, we assess that this campaign is more likely the work of a cybercrime group rather than a state-sponsored operation for the following reasons:

- Unlike nation-state actors, who typically target high-value assets such as government agencies or critical infrastructure, this campaign displays no such selectivity. Instead, it targets a diverse range of organizations with no clear strategic connections, reinforcing the conclusion that financial motives drive the attackers.
- The infrastructure used, such as creating different Gmail accounts for each phishing attempt, indicates the possible use of automation tools possibly powered by AI. This level of operational efficiency, along with the indiscriminate targeting of multiple regions and sectors, points to a cybercrime group seeking to maximize financial returns by casting a wide net.

Rhadamanthys 0.7

*While working on this report, Recorded Future, a cyber security company,released a comprehensive analysis of Rhadamantys 0.7.*The Rhadamanthys version used in this campaign, identified as 0.7, is the latest release at the time of this writing. It was introduced by the developer of the malware a few months ago in the following announcement:

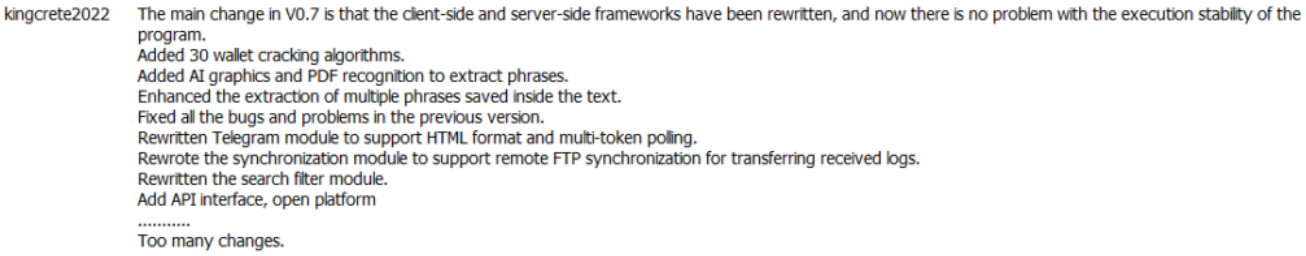

Figure 7 – Source: https://x.com/g0njxa/status/1812902577530454023

The author stated that text recognition is implemented with AI. As AI is currently a hot topic, this may help promote the product and show that it uses cutting-edge technology. However, as we found out, the component introduced by Rhadamanthys does not incorporate any of the modern AI engines but instead uses much older classic machine learning, typical for OCR software.

In this latest release, the author announced many improvements. Some of the existing components were polished and modified. Only one new executable, the OCR component, was added.

The OCR module

In the package downloaded from the C2, we find the following:

- ImgDat (full path in Rhadamanthys Filesystem: `/bin/amd64/imgdat.bin`) – An executable in XS2 format.
- bip39.txt (`/etc/bip39.txt`) – A small dictionary.

The ImgDat is the OCR module that the author mentioned in the announcement: *"Added AI graphics and PDF recognition to extract phases"*. The newly added text file `bip39.txt` is its configuration and contains a dictionary of search phrases that will be checked against the extracted text.

The name Bip39 suggests that it is related to the Bitcoin Improvement Proposal 39, which states how to create phrases out of numbers to make wallet protection codes easier for humans to remember. According to the specification, Bip39 contains a dictionary of 2048 words – just like the file we found. We compared it with the official Bip39 wordlist and confirmed that the content is the same: https://github.com/bitcoin/bips/blob/master/bip-0039/english.txt (https://www.blockplate.com/pages/bip-39-wordlist).

Based on this information, we can easily guess that the OCR module is applied to search for documents where such phrases may be stored, and the retrieved information will be further used in attacks on Bitcoin wallets. This set of phrases used for text recognition suggests that the campaign is motivated by financial gain rather than espionage purposes.

How ImgDat is deployed

The ImgDat executable is deployed by the main module of Stage 3 (coredll.bin), which is responsible for coordinating the work of all the stage's components. In the ImgDat module, calling the Entry Point retrieves a list of its exported functions that are used later.

The exported functions:

- `init` – Initializes the OCR component with a given configuration and returns the context structure.
- `delete` – Destroys the initialized structure.

- `process` – Implements the main operations – image processing and text extraction.

> Workflow:

1. The function `init` is called and the content of “bip39.txt” is passed to it to initialize the component with the given list of searched phrases. They are stored in the dedicated linked list that is a part of the context structure.
2. The function `process` is fetched from ImgDat. The core module walks through the disks of the infected machine and then calls this function on every retrieved path.
3. The function “process” from ImgDat is called a callback from within a filtering function. It first checks if the retrieved path contains the extension from the hardcoded list. The supported formats from which the module can retrieve the text:

- BMP
- JPEG
- PNG
- TIFF
- WMF

If the extension matches, the file content is read and passed to the `process` function from the ImgDat module.

The OCR implementation

The image is loaded via the GDI+ interface and then preprocessed to facilitate text recognition. First, all pixels from the image are loaded into the dedicated buffer:

```
to_GdipDrawImageRectRectI(&v39, gdip_img, &img_start_ptr2, 0, 0, img_w1, _img_h1, 2, &imgAttr, 0LL, 0LL);
img_buf = malloc(img_w1 * _img_h1 + 1);
img_start_ptr1 = img_buf;
img_start_ptr2 = img_buf;
if ( img_buf )
{
  memset(img_buf, 0xFF, img_w1 * _img_h1);
  pos_Y = 0;
  if ( _img_h1 )
  {
    dst_indx = 0LL;
    do
    {
      pos_X = 0;
      if ( img_w1 )
      {
        ptr = img_start_ptr2;
        do
        {
          pixel_data = 0xFF000000;
          if ( !GdipBitmapGetPixel(bitmap, pos_X, pos_Y, &val) )
            pixel_data = val;
          ++pos_X;
          ptr[dst_indx++] = (BYTE1(pixel_data) * *&qword_14004D298
                           + BYTE2(pixel_data) * *&qword_14004D290
                           + pixel_data * *&qword_14004D288);
        }
        while ( pos_X < img_w1 );
        _img_h1 = img_h1;
      }
      ++pos_Y;
    }
    while ( pos_Y < _img_h1 );
    _stc = stc;
    img_start_ptr1 = img_start_ptr2;
  }
  LODWORD(_stc->img0.is_loaded) = 1;
  _stc->img0.pixels_buf = img_start_ptr1;
  _stc->img0.img_width = img_w1;
  _stc->img0.img_hight = _img_h1;
  is_ok = 1;
```

Figure 8 – Fragment of a function denoted as “read_bitmap_from_image” within the “process” API.

The RGB components of the picture are compressed into a single byte.

The OCR functionality is implemented within the function denoted as `extract_text` that is a part of the `process` API. When we look inside, we can find a reference to thresholding, a well known technique commonly applied in OCR software. Its role is to enhance the contrast to be able to distinguish the text from the background of the image.

```
45  v33 = a4;
46  v31 = 0;
47  sprintf(Buffer, "[thresholding] image = %p , (%d , %d) (%d , %d)\n", a1, a4, a4 + a6, a5, a5 + a7);
48  OutputDebugStringA(Buffer);
49  if ( a5 < (int)(a5 + a7) )
```

Figure 9 – Rhadamanthys OCR code.

```
[thresholding] image = %p , (%d , %d) (%d , %d)\n
```

The image is then processed using a trained local machine learning model. Extracted sentences are stored in the dedicated structure.

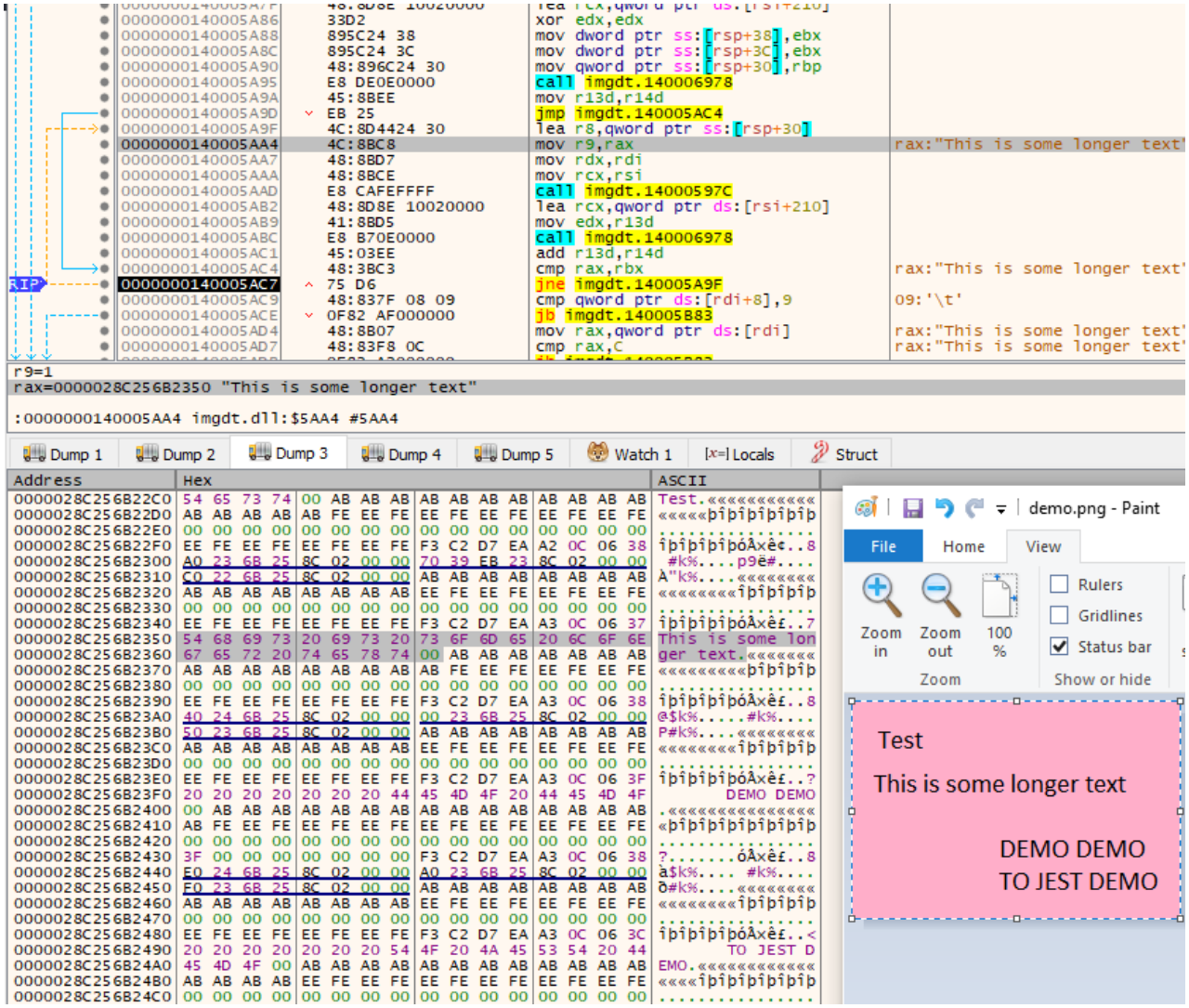

Figure 10 – Example of an input image (PNG) and the phrases extracted with the help of ImgDat.

Finally, the extracted phrases are separated into words, which are then compared with the previously initialized token list. If there are enough matches (at least 9 strings from the list and at least 12 strings processed), the `process` function returns `true`.

The model has limited precision. For example, it handles only the most popular fonts and cannot recognize handwritten text. In addition, it doesn't do well with text in mixed colors (especially if one line of the text is darker than the background and another is lighter).

Conclusion

In this article, we analyzed a large-scale phishing campaign discovered in July 2024. This campaign used a copyright infringement theme to spread the Rhadamanthys info stealer. This campaign employed tactics including DLL sideloading and anti-detection techniques, making the latest version of Rhadamanthys (0.7) more potent and more challenging to detect. We also examined Rhadamanthys' new OCR features.

The campaign's widespread and indiscriminate targeting of organizations across multiple regions suggests it was orchestrated by a financially motivated cybercrime group rather than a nation-state actor. Its global reach, automated phishing tactics, and diverse lures demonstrate how attackers continuously evolve to improve their success rates.

Check Point Customers Remain Protected Against the Threats Described in this Report.

Check Point's Threat Emulation provides comprehensive coverage of attack tactics, file types, and operating systems:

- InfoStealer.Wins.Rhadamanthys.ta.V
- InfoStealer.Wins.Rhadamanthys.*

Harmony Endpoint provides comprehensive endpoint protection at the highest security level, crucial to avoid security breaches and data compromise :

InfoStealer.Wins.Rhadamanthys.*

Harmony Email and Collaboration provides comprehensive inline protection at the highest security level.

Appendix A

Stage 2 – Unpacked from the hardcoded package:

Name	Format	SHA256
dt.x86	XS1	bea558e8129fcb647e6f42c8beda4464e109dd3cd546342c0337dbd50616f991
early.x86	XS1	4fd469d08c051d6997f0471d91ccf96c173d27c8cff5bd70c3f2c5008faa786f
early.x64	XS1	633b0fe4f3d2bfb18d4ad648ff223fe6763397daa033e9c5d79f2cae89a6c3b2
netclient.x86	XS1	b97dd0279e112e0591b38064f59077102ab188b07a069cb104e66e4756e2570a
phexec.bin	XS1	13872271ee511aa83f3f27d5db248516652b10a079ad01f78ed734cd2a87ec77
prepare.bin	shellcode	d96ec4b08c08b81ba9075423d5e83bf330de09866066b4bdb459bcbac389a350
proto.x86	shellcode	a905226a2486ccc158d44cf4c1728e103472825fb189e05c17d998b9f5534d63
stage.x86	shellcode	44f3936ee158d2846664bf5cd795fd90a99441186b20b90ff241ba1b38a6a3e9
strategy.x86	XS1	219a6387d91c4b2c8e91c8613192af950bd9c790114a238eb0e1e7c878f6e728
unhook.bin	XS1	37438095a5e7be0ce12997dc23d1ff117912989d2f24beab95284f9380f65834
ua.txt	plain text	aeba4ece8c4bf51d9761e49fad983967e76c705a06999c556c099f39853f737c
processex.x	plain text	3ca87045da78292a6bba017138ff9ee42b4e626b64d0fee6d86a16cc3258c8c3

Stage 3 – Downloaded from the C2:

Name	Format	SHA256
coredll.bin (32)	XS2	3737501bbd4abd0844da016c0263399e3c670ae52952b30ca46c6c96cf4e318d
coredll.bin (64)	XS2	6012386eab453f4fb1cfb88fb5b05ba9ec71a838029ea51bcff4c0b5a2fbfad2
taskcore.bin (32)	XS2	c0b319bb19092fe3c193e5139fcdf599502b669143b06c676e81f46ab50fb4ed
taskcore.bin (64)	XS2	18273fa35c54332d8763cb17a5ae92de5636f3a05c507ce18d9d6a77c3139deb
stubmod.bin (32)	XS2	d97aa65123c26509e3fc1a9963962b7f707a50ddca44a9a12fd03e654ab5aa66
stubmod.bin (64)	XS2	fd9fbfa809450415e8d0d79199ec8686cb7071d6e13a5b76f0ce1b03a2a61302
runtime.dll	PE, .NET	a87032195e38892b351641e08c81b92a1ea888c3c74a0c7464160e86613c4476
loader.dll	PE, .NET	3d010e3fce1b2c9ab5b8cc125be812e63b661ddcbde40509a49118c2330ef9d0
KeePassHax.dll	PE, .NET	fcb00beaa88f7827999856ba12302086cadbc1252261d64379172f2927a6760e
imgdat.bin (32)	XS2	2625d99af56c79de32f9fba2332f63eb9c88707e9ea83985bce5df9022ced99a
imgdat.bin (64)	XS2	ffb264a19af7c8a8dd5357b62c45fcd3063ca946aa2710740c4e8b21f8e697d9
bip39.txt	plain text	24ce42c2fd4a95c1b86bbee9bce1e1cf255bd0022e19bab6bd591afd68b7efdb
Lua extensions	LUA code	–

IOCs

C2:

- 198.135.48.191
- 139.99.17.158
- 103.68.109.208
- 95.169.204.214
- 15.235.138.155
- 15.235.176.166

Archives:

- d285677cba6acf848aa4869df74af959f60ef1bc1271b4032000fcdd44f407f2
- 2be6ad454fa9e87f78dea80d2855f1c14df81a881093a1a0d57f348377f477a8
- 9ef9c88cef51ee0fb77ea9a78dbe60651603ef807ddb6c44d5bda95cc9026527
- e8aa9a061c6ea803faaf4c8d7a80c6886b4ee73d9a89a9dc6e87e3fecf7a6851
- b1ac4ad92045e935c132214015188d27ec4382f930d0152dfb303695b708b38d
- 00086cf4f35b6fb7f897cfa2f0d5ad9876aa9819cdc87416c798005ce901d3a1

- 05e02f0f9b8625fe3959ae1219f31b0167d787fefc0a9d152edf6524d6859590
- 0a3dfe260dd7b038ddb8911689c899541391c188aff966261e7bd9d0280d153d
- 0b9bd95d815af9ea4a59840ef6fcdc7ccfd0e239c40974334cb4cfb41df530db
- 0de8d2d3217cebd37a2fe488713d1c288ae5a63d3d3b2a3495e2e636ba6a1f89
- 10eafd75429ffadee2384acd37b0d4e7ca26b83666e6786f2acaf1b1c29c3f17
- 12b7390835f30c1bcdeddd258e49684c98133cee4a6a2ccab869785567deae4f
- 2a276ca5b2e095cdac7b24e58b3f7a67cee7db2fb5c1568e4775909265c7e914
- 2aa58fa8d71bd2b4fd1ffac16a6461191bbf6f4b2c97455ae52800cce929a0f2
- 2e0c99758432a3759b5af6f190ec5cb72a5a84c977d8883dcf041c4de003f3d3
- 324dfc7bb75f27e6fba8d67dea67a63525efbe947bf8e29ef39980c6efc1c3f6
- 3448005600ccb0ae52443a4c227a657de9cd767b389e9a1ed75ef074709981bd
- 3de252c9023bc8920d77570acdfe21813532727af3f91d59af35fa8abcd3700f
- 3ecf2838b2e07e6d329d45cde7d0162ba47fea4b94bacb24838358314daed756
- 415ee9b12002f17ca4f36bef794fdb19884e22980e21bf8a15043258624c439b
- 416f3fa48b75ab168e3373dae77cab7f4702de5158835d23a02629e8c1d20156
- 41a3edb3a8e8d5cf093cbd02791911f6ee26df39a377fceb6b101d66a7b7aff2
- 4b33219c5cadb4d741044874f6f0184d45f43891d28ad5b489716d4da21310fd
- 4bbe0f6b5488a51295b15d8144d0a1c9b41bb86384299b88ea48e88c76704f52
- 4cbcfa2a8d56976eff1e8ac0ef4d7703d0b802f227975a0cc36f3dcd3a90e73e
- 5cec33e8f47855da3c4ce1f3953d750275864714b16e08a94605bc3889867caf
- 6044e08402d1abd52991f5c6a4749ba6aa29a0587ff196edf60b38862392e855
- 623bb3f1f476c37afc309d6c0ab89e216aaedc03b8a7ec1aaec5fb5085d78a97
- 741dfdae8948f3e430a5b7b66c8fb4b8a750695b67a84a12abc0b6089e8fba31
- 7990765022c4400a45f996046971b9e6b69cca5b06f8d2adb61bc267fd362197
- 7d7a3e254b7968400a301d83fcd44a69f655386b9b95998a36113cfb2e542720
- 7dc07b8aa268485e40ab78bfbb03a367d80ebd7b2c6c74961dc6842cae7086e1
- 7e270a80cd0f04f245309e8c75cfc2cb46dc075ba01a00b30f66cb8b5deaaf3f
- 865a4f2583679f7a40357b61301d75567cf516a5b8295dc8155e6d4aa2ce244a
- 878917b6a8d241031fc330eff771f416a9fffaecab42c39d57e58ac2d8f38f11
- 970e199e40511e90d6dd5d6f3c9f3701215fd881b1273fe2617bd44444b0bee9
- 9a249dfdc2c16700bc5add2455f2ed00e47a2610b7779cc33e40aac576a2a74d
- 9abf9fb94e2529d8819a3873f2025bdd90d14e75fe4af81e489f6d0560809f9c
- 9d10835f7717c89d17886b7e59cc2dfc9133bfaa044bad5f070e1c8e1212e257
- a03d2956ff8d0ae4d96c9e6cced79b335b70eef10feb0f7202609cb8652179f6
- a064bbc4b58642ab4d7118abc55fb81db6584cbc633800ad14048e8370a95ef2
- a15d0aedc8b4e54a170b6ecc3d9a06835cc499f07b05c6ca261081ace505debf
- a72083974e886856b7d985bdc79888234c8cd9012ed39b2566851fb0d86cca50
- a8729621ca4310e8e1a7ad3e1426708f1e1954a16af420cd3ce46c501e9692ab
- a9896a8f96407a5eedda08a63dd40967f0fe0b3926e7002b6e1abc11f6ab81cc
- aa04c9307a9087455d21dfac02d7f322ab337cd5978f9161285a9c79379efecc
- b36205464ead176a473ab43ea7b5e0c2b8749b3eb9549d65609be2337dce25db
- b529c6df6164ff8badf30f942220a3126f99e3fc2c2ea1494aa3e305b3b53c1f
- b9c4c8343ba75081954b2db54940585c6c0c9bb47e053ac1b9229b4fa8fc9293
- be9c3feed5f6e81ccd375902c8c92616f77694b6cd14f69896d44dd4b1ea4990
- c5bb808a88f9e729484c05a1bc3097157bbfbd28469e502f2ebc4c6e6135df42
- c622c0f67eb5d9a90008e5e120065cd5a1a6e25c6e758e8205d377596059b8fe
- ccb539bf17d479d9707ee717d0afb03cd57e9b6f023becf1abf9cdbd88e1b06c
- cd3040c88a6fd71ed1ce8c2a5d0b13ed8e25e49835932a39891c514ef946dd29

- ce2f00f1d0e71287e746d5a3507547f355297a3e45a7c2cc0322015916a0137c
- d00d3adf81bf95ff4994dcbd2ae1305a6ee6b0edfad6eb55b87217f85645651a
- d0e3f547e3efcc9d9794774a765b9c3950955e7ad752f3e630ebd5ab9425bcdc
- d452461f3527d674de3e9b680026ceb2b02c56d6d3f7c94da3aab65c05f52c03
- d57f45096e646837dec51129222fcbe79981c595721164009aec68be09bf5dcf
- dbb4f7e6354621c316fbba7e7a15f59cf229684e16ab6d21027f310beecaf49b
- dbdeede6f39936305c4c5bd8e4f7bfccb0b823c025130e7f8fa285e80383be0f
- dc3d72f72247141efeba3c2ffd498025f68e0c4b34c9a4dc2686ffec09b6d401
- de933f7b47707f4bf8d5a4aaef8b31f5059d3b8f465bcaae3e22438466e8390b
- e6315b24e0311758da1c25daa5f2724da4f534ed7ed644cbf43f3cc64c4676a7
- e9a18755312011e30081e7ce0fcc1db3e3aec3b9f3ed3a776dd38498830a2738
- eb4e39d44ad016b8d6d1dc8dc25a9ea3d3e18df87516922fdbd995de15b68f54
- ebd167ca477af620065548a9e55567682b0750625b3e078fc4498dd5adeabdc6
- f2536e520d37512d868a418797974a5c11e67742824a5477100b7e3f5b2efbc3
- f4fcba1c9d7f4ae8e3868f901035ea1e0e9e1122a362a83afd3d111c17a97d7a
- f7eef906c7dc1ce2ffe586d4b7f316a5f5c6761b5cdbf22d892fbc87a5ee2f6f
- fe55c1d263e0ea356d86afd8b2b1cedff570568e45b8a3810e05ea482b8a9329
- fefba5ce20c71a71cfe35dd8ff06c514bf6ffde60356babf4f4bba66dd904b78

DLL:

- cf9d93951e558ed22815b34446cfa2bd2cf3d1582d8bd97912612f4d4128a64e
- 48aaa2dec95537cdf9fc471dbcbb4ff726be4a0647dbdf6300fa61858c2b0099
- 00fc4b8a4c65c06766608f3ef3f92385c8e147f5991dabe290e33dd14b39ad44
- 0ad65fd0897a6547f6febf398708ab2d423a8f8834b53136219cb490ec3ebd13
- 11ba24d023b544e28c37b6cb8afe27d06638175d7f56c2e4d4ff97bf7bd813b6
- 1a2399ecc38f3288206c75b55762d125d3d75254062a2c0d85c86e7f896736ac
- 258ffcc13dbe110bcce21b91f7f075995719791fdd3c9f55ea5934984fa4373d
- 2cbc1e8a4cb5d18a867666adbd3417bc88d48a74ae6500593959aec1a1c92d2d
- 342a5c7df2bdd040570f4b83c74366d4c96a90d6418149d432cb5e8577f2f6b1
- 3648e89e7449ea433a8b3ef0e5b605b5dc4157048c03b20dedc5e3b920fa8552
- 5418e42706bca4712ff2a3db67853eb42a2310660c51cff2f9020586cffedeb3
- 69573694d16b7ccadfa208ff976bfe1b3e36837aba3e5dc4dfc80e66341ef61e
- 6de4f65b1d738d84f8e825613092bbd360194195fe8a1c986e12a9bb704217c1
- 751f149665f87dd20cc8dff743f28e5da1ff2a5f04874d4b8569b9afceeedfec
- 78200cd816acbd39b6664c6582e06500f6d46085b62b49d2f914bea5a004197a
- 783c7f4bf23072343f6247ee14e54e4af0b147553ad1ef42b4e7fb44386d667c
- 7f99e506c17676b98dcc08e6a19f100ef933cde3e0423c6d4072f6802a9196bb
- 8d0b1174cbda6b102bb98c91ba123e9f404b9fad23b49a4e29f3cfd8d20a577a
- 90c7688e0dc23ba4530bac1d567bad920c4ef1c06cbf4b2d867eeb363271eefe
- 9102e564c3262b2c291e8ca3d67f8a55c06650aa86f617c919916f6053c03c9b
- 9327aa03760431b6d86eeb2f1a3efc36aa443b842b5116fbbe0f2a7794c4e70e
- 97286b6f3a6535ff1172ef65172e6967e3670c6b14a3313c3bf0d6c171b1fc85
- 98e28d3423f5d414effe3c0ed6fd0f1c8154942e5e127ecee5f051e1196ffc75
- 99c0bebdc8cb7b0948000a601f510fc70487f9da532be199b8641512a2db9839
- 9bdf49b27fd4d80ef087f63e0bfa0a0822686814863eca09ac506404ad76dfda
- b2588061ba5ee9948bbccd320b40c6d7b8d6a693d181f3bce61e5e267f53aa7e
- b936853a0c50a0cd0bc8b33103b55bd88e19c6c28768d990b954c11d714286ca

- f2429f4bd09897653d0ffa41206a14cafa55356d5edc04dc0915c116867f8c27

GO UP
BACK TO ALL POSTS