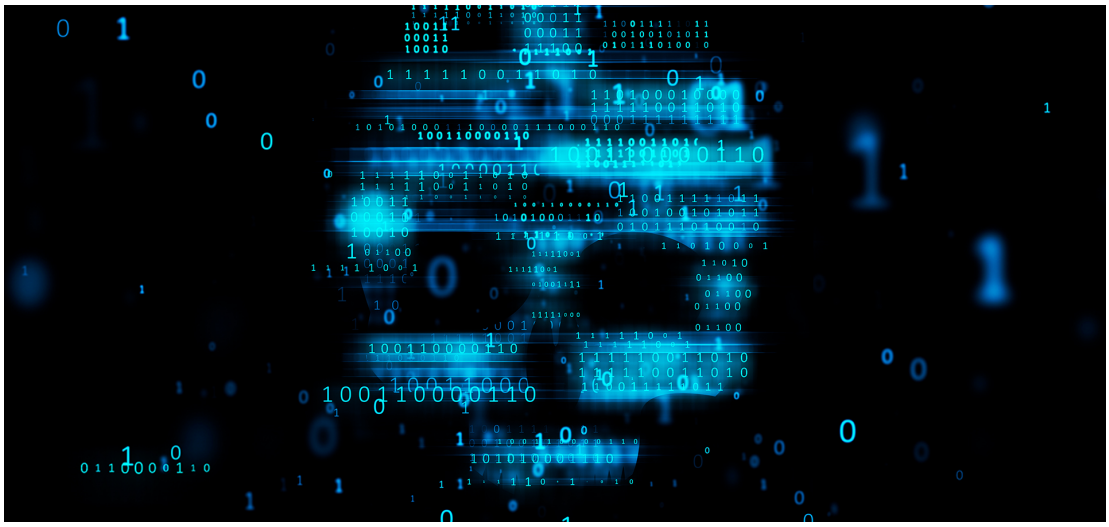
PhantomVAI Loader Delivers a Range of Infostealers

Tom Fakterman ⋮ ⋮ 10/15/2025



Executive Summary

Unit 42 researchers have been tracking phishing campaigns that use PhantomVAI Loader to deliver information-stealing malware through a multi-stage, evasive infection chain. Threat actors wage these campaigns to deliver obfuscated scripts and loaders that use steganography techniques to conceal payloads.

The loader initially used in these campaigns was dubbed Katz Stealer Loader, for the Katz Stealer malware that it delivers. Hackers are selling this new infostealer on underground forums as malware as a service (MaaS). Recently, we observed that the loader now delivers additional infostealers, such as AsyncRAT, XWorm, FormBook and DCRat. Given this unique behavior, we now track the loader under a new name: PhantomVAI Loader. We chose the name because of the loader's stealth and the VAI method it executes.

Threat actors deploy PhantomVAI Loader in attacks worldwide, targeting organizations from a wide spectrum of industries:

- Manufacturing
- Education
- Utilities
- Technology
- Healthcare
- Information
- Government

We explore each stage of the multi-layered infection chain, from the initial phishing email to the final deployment of the infostealer payload. We also outline the functionality of Katz Stealer specifically.

Palo Alto Networks customers are better protected from this activity through the following products and services:

- Advanced WildFire
- Cortex XDR and XSIAM

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

Related Unit 42 Topics **Infostealers**

Background

On April 13, 2025, a user called katzadmin posted about a new infostealer named Katz Stealer. The user uploaded these posts to the BreachForums underground forum, and later to the exploit[.]in and xss[.]is forums as well. Katz Stealer is a type of MaaS that collects sensitive data from a variety of applications hosted on infected machines.

We observed threat actors delivering Katz Stealer through phishing emails containing obfuscated JavaScript or VBS code, PowerShell scripts and a .NET loader. Initially called Katz Stealer Loader — and also known as VMDetectLoader — this loader now delivers infostealers such as AsyncRAT, XWorm, FormBook and DCRat. We track this loader under a new name: PhantomVAI Loader.

Infection Chain Analysis

The PhantomVAI Loader attack chain starts with an initial phishing operation and culminates in the deployment of payloads. Figure 1 summarizes the steps of this process.

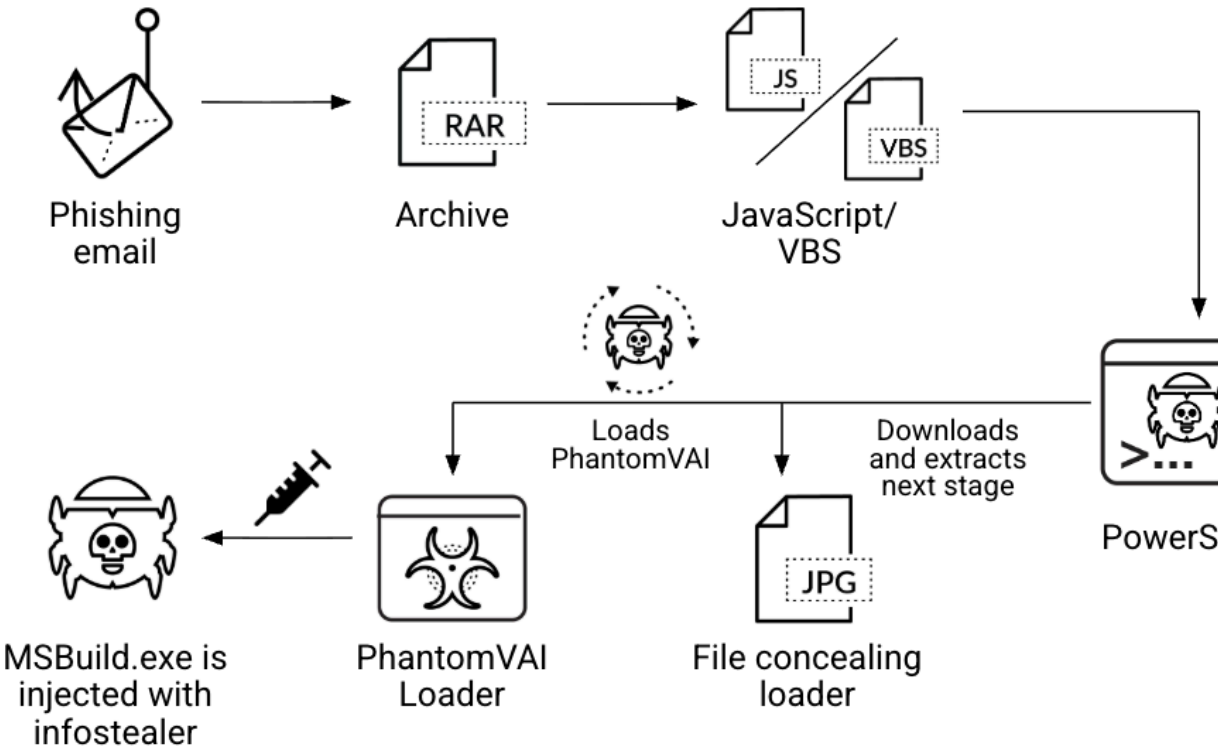


Figure 1. The PhantomVAI Loader attack chain.

Phishing Emails

The infection chain starts with a phishing email that contains a malicious attachment. Figure 2 shows an example of one of the phishing emails.

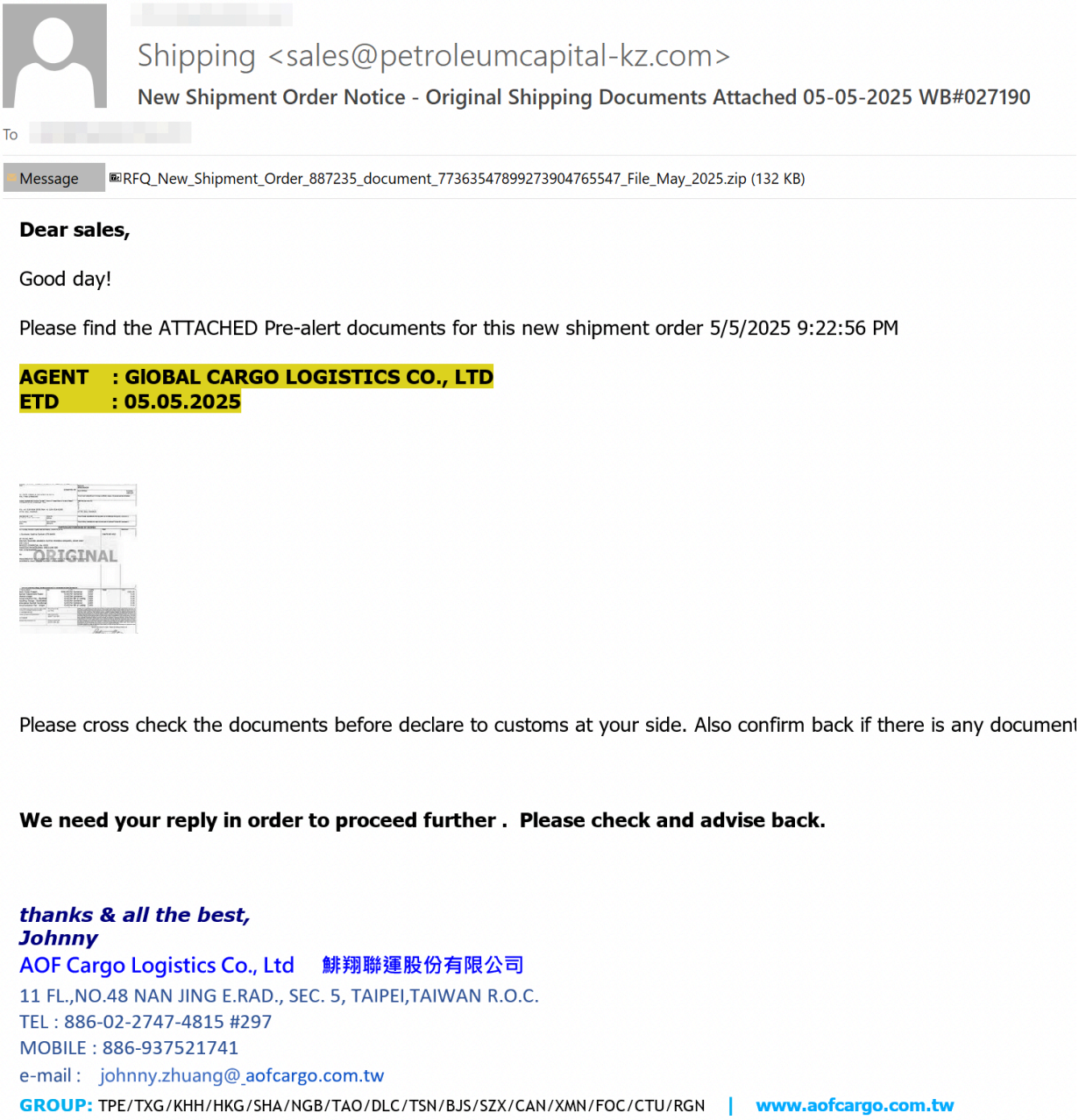


Figure 2. Phishing email. Source: VirusTotal.

The emails contain themes like sales, payments and legal actions to trick the targeted users into opening the malicious attachment. Some of these emails incorporate homograph attacks, which involve replacing Latin characters in the email with other Unicode or math characters. Attackers use this technique to bypass email defenses by disguising terms that email security mechanisms usually flag as suspicious.

Stage 1: JavaScript and VBS Scripts

The phishing email attachments are archived JavaScript or VBS files. Threat actors obfuscate these scripts in an attempt to bypass detections. Figure 3 shows an example of obfuscated JavaScript from one of these files.

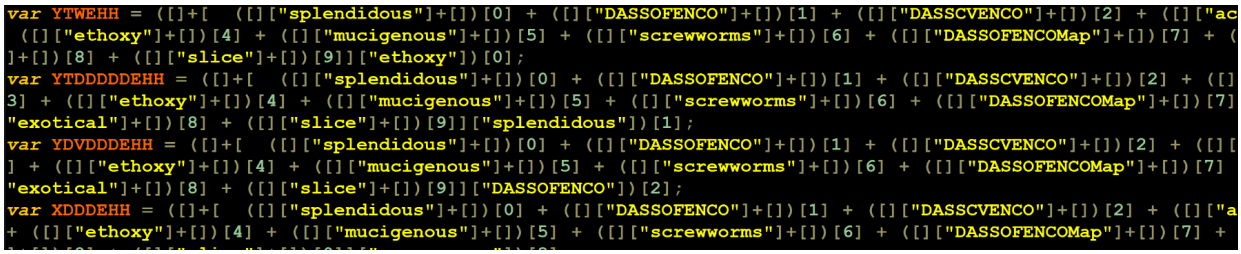


Figure 3. Obfuscated JavaScript.

The script embeds a Base64-encoded PowerShell script and executes it to download and deliver the next stage of the infection.

Stage 2: PowerShell Script

The decoded PowerShell script downloads and loads the next stage of the infection. Figure 4 shows an example of a decoded PowerShell script.

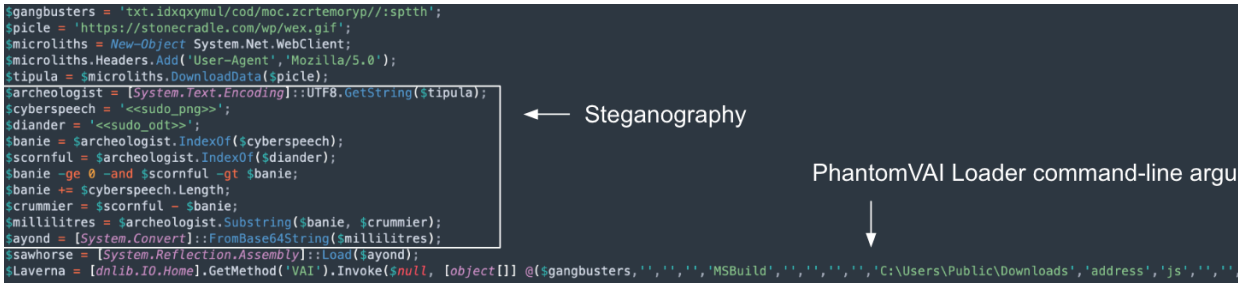


Figure 4. PowerShell script used to download the next stages of the attack.

The PowerShell script downloads a GIF or other image file that conceals the loader payload. This technique is known as steganography. In the infections that we observed, threat actors used this technique to embed text within the image. The text is a Base64-encoded DLL file.

Next, the script extracts the Base64 data by searching for specific strings that represent the start and end of the encoded text. In this case, the PowerShell script searches for all text between <<sudo_png>> and <<sudo_odt>>. This text is an encoded DLL. In other cases, threat actors inserted the encoded text between different headers. Figure 5 shows an example of encoded text embedded in a GIF file using steganography.

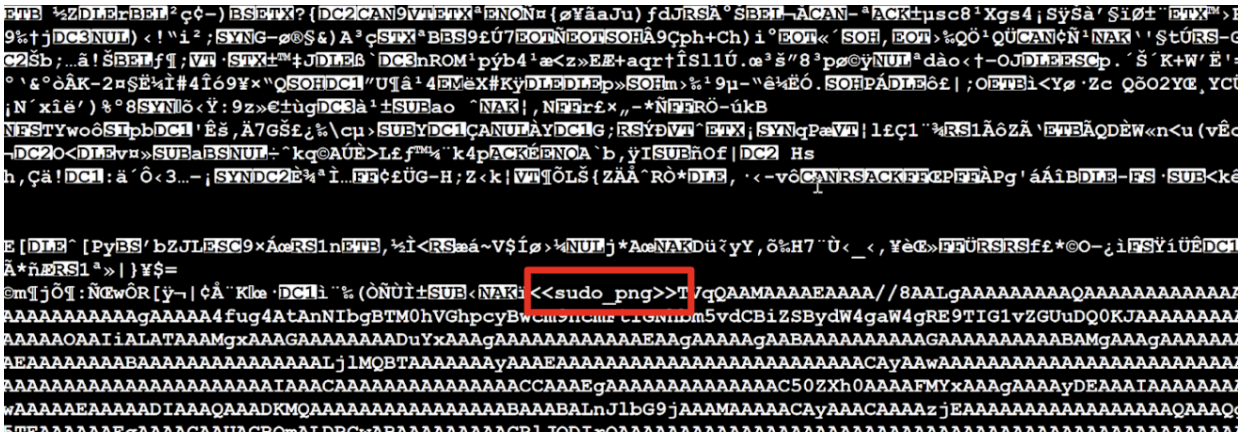


Figure 5. The start of encoded Base64 text embedded in a GIF file.

After extracting the encoded text from the image or GIF file, the PowerShell script decodes the text and loads the DLL. The loaded DLL is the .NET loader payload that we call PhantomVAI Loader.

The PowerShell script invokes a method called VAI within PhantomVAI Loader and provides it with several parameters. The first parameter is a URL for the command and control (C2) server that hosts the final payload.

Stage 3: Executing PhantomVAI Loader

PhantomVAI Loader is written in C#, and the VAI method has three main functionalities:

- Running virtual machine checks
- Establishing persistence
- Retrieving the final payload

Virtual Machine Detection

When PhantomVAI Loader is executed, it performs checks to determine whether it is running on a virtual machine, as the code below shows. The VM detection portion of the code appears to be based on a GitHub project named VMDetector. If any of the checks return a true response, PhantomVAI Loader exits and stops executing.

```
1 Detected as a virtual machine given key computer information.
2
3 Detected as a virtual machine given bios information.
4
5 Detected as a virtual machine given hard disk information.
6
7 Detected as a virtual machine given PnP devices information.
8
9 Detected as a virtual machine given Windows services information.
```

Establishing Persistence

PhantomVAI Loader uses one or all of the following methods to create persistence:

- A scheduled task executes PowerShell commands to download a file from an attacker-controlled URL. The task saves the file with a specific name and extension and then executes it.
- A scheduled task executes a script using wscript.exe. The path to this script is supplied as a command-line parameter.
- A Run registry key to execute a specific file. The file's path is also provided as a command-line argument.

Retrieving Payload and Injection

PhantomVAI Loader downloads the payload from the URL specified as a command-line parameter in the Stage 2 PowerShell script. It then injects this payload into a target process that is also defined by a command-line parameter, using the process hollowing technique. The loader injects the payload into a process located in one of these four paths, depending on the command-line argument and the payload architecture:

- C:\Windows\Microsoft.NET\Framework\v4.0.30319\
- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\
- C:\Windows\System32\
- C:\Windows\SysWOW64\

In most of the cases observed at the time of writing this article, PhantomVAI Loader injected the payload into the Microsoft Build Engine executable, MSBuild.exe. Figure 6 shows an example of such an injection, in the context of the infection chain.

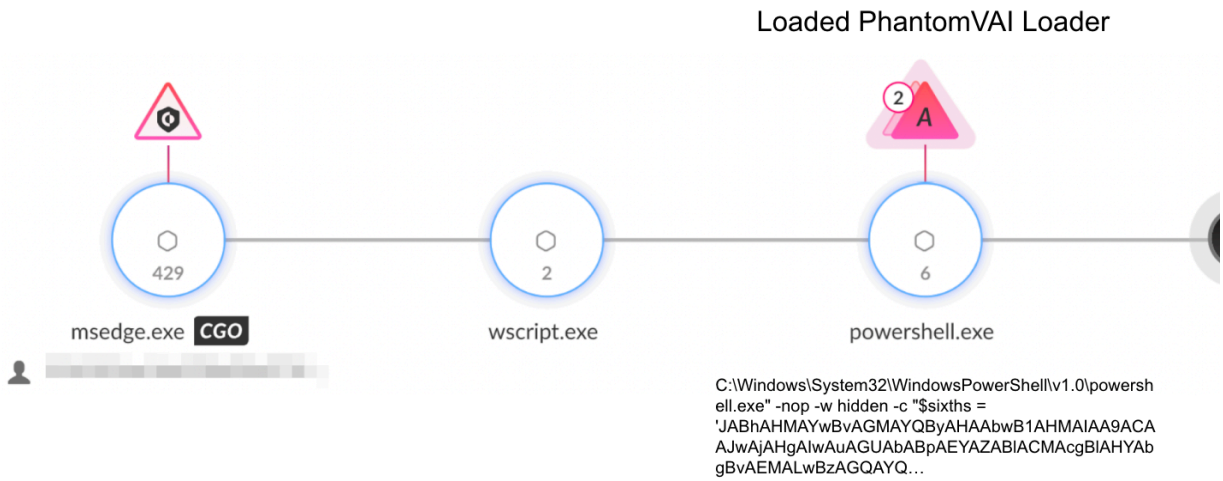


Figure 6. Infection chain that starts with the user opening an email using msedge.exe (Microsoft Edge browser) and ends with Phant injecting the payload to MSBuild.exe.

Katz Stealer: A New Malware-as-a-Service Stealer

PhantomVAI Loader has evolved to deliver a number of infostealers. As Katz Stealer is the least well known and documented, we cover it in additional detail here.

Threat actors use Katz Stealer to steal data from infected machines, such as:

- Browser credentials
- Browser data (such as cookies, history, login data)
- Cryptocurrency wallets
- Telegram data
- Discord data
- Operating system information
- Steam and game data
- VPN data
- FTP clients data
- Communication and messaging applications data
- Email clients data
- Screenshots
- Clipboard data

Katz Stealer also checks the machine's language and compares it to a hardcoded list of country codes by using the following APIs:

- GetKeyboardLayout
- GetLocaleInfoA
- GetSystemDefaultLangID

The country codes that Katz Stealer checks are all part of the Commonwealth of Independent States (CIS), as Figure 7 shows. If it finds a match, Katz Stealer stops executing. This language check and subsequent behavior could provide a clue to the origin of the author of the malware.

```
country_codes   dq offset aRu          ; DATA XREF: sub_140005366+3B↑o
                                       ; "RU" - Russia
                dq offset aBy          ; "BY" - Belarus
                dq offset aKz          ; "KZ" - Kazakhstan
                dq offset aKg          ; "KG" - Kyrgyzstan
                dq offset aTj          ; "TJ" - Tajikistan
                dq offset aUz          ; "UZ" - Uzbekistan
                dq offset aAm          ; "AM" - Armenia
                dq offset aAz          ; "AZ" - Azerbaijan
                dq offset aMd          ; "MD" - Moldova
```

Figure 7. Code snippet showing the country codes that Katz Stealer checks.

Conclusion

This article highlights phishing campaigns that deliver PhantomVAI Loader, also known as Katz Stealer Loader. Combining social engineering via phishing emails, obfuscated scripts, steganography and a .NET loader, this multi-stage infection chain demonstrates the lengths attackers go to in attempts to evade detection and bypass defenses.

Our research highlights how this loader has evolved in the cybercrime ecosystem. While initially, threat actors used the loader solely to deliver Katz Stealer, recent observations show that the loader now distributes additional malware strains, including AsyncRAT, XWorm, FormBook and DCRat.

MaaS offerings like Katz Stealer are a pervasive threat that can significantly impact security and privacy by exposing sensitive data such as passwords, networking data, emails and files. Understanding the attack chains and techniques that threat actors use to deliver these malicious payloads is vital to ensuring organization security.

Palo Alto Networks Protection and Mitigation

Palo Alto Networks customers are better protected from the threats discussed above through the following products and services:

- The Advanced WildFire machine-learning models and analysis techniques have been reviewed and updated in light of the indicators shared in this research.
- Cortex XDR and XSIAM help prevent all the threats described above by employing the Malware Prevention Engine. This approach combines several layers of protection, including Advanced WildFire, Behavioral Threat Protection and the Local Analysis module, to prevent both known and unknown malware from causing harm to endpoints.

Figure 8 shows two examples of detection alerts that the emails in this campaign trigger in Cortex XDR.

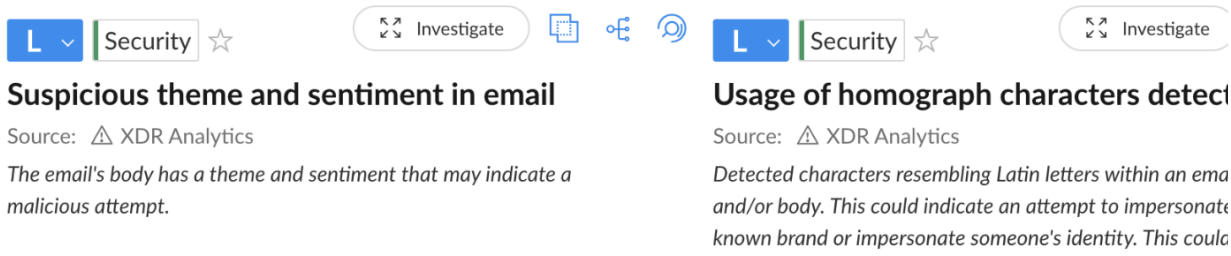


Figure 8. Detection of phishing emails that contain suspicious themes and homograph characters.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

Indicators of Compromise

SHA256 Hash for Archive Example

- 02aa167e4bb41e3e40a75954f5a0bd5915f9a16fd6c21b544a557f2a7df3c89b

SHA256 Hashes for JavaScript Examples

- e663916cc91b4285a1ee762716ff7ce4537153c7893e2d88c13c7e57bbb646a9
- 45fddf55acb50df5b027701073dee604b4135f750c585b29d6dcac824f26ae00
- 9f28f82d21fe99d0efdcab403f73870d68fd94e6d0f762e658d923ccd1e7424c
- 05d66568017f2c2e417fa6680f9b4fa4a8a9bc1b7256fe46fbf3e71956b99773
- 4346c3c08df612b8bcd23a3b57845755bafb0efc57ff77203f8da3b46628a008
- 0c0dae4d7da069c928f06addb1c5c824e820e4556a1244142f56227954bf9c7d
- 3a039ce210a0b5ff65f57d304519b885bae91d1bec345c54e59e07bc39fca97e

SHA256 Hashes for PhantomVAI Loader

- 4ab4a37db01eba53ee47b31cba60c7a3771b759633717e2c7b9c75310f57f429
- 9ae50e74303cb3392a5f5221815cd210af6f4ebf9632ed8c4007a12defdfa50d
- 893ee952fa11f4bdc71aee3d828332f939f93722f2ec4ae6c1edc47bed598345
- b60ee1cd3a2c0ffadaad24a992c1699bcc29e2d2c73107f605264dbf5a10d9b6
- 0df13fd42fb4a4374981474ea87895a3830eddcc7f3bd494e76acd604c4004f7
- 6051384898e7c2e48a2ffb170d71dbf87e6410206614989a037dac7c11b8d346
- 01222c6c2dbb021275688b0965e72183876b7adb5363342d7ac49df6c3e36ebe
- 6f7c5bad09698592411560a236e87acae3195031646ff06a24f1cfada6774ba6
- 6aa2989ebb38e77a247318b5a3410b5d4f72b283c7833a0b800ea7d1de84ccc6
- 4c5d7e437f59b41f9f321be8c17ae1f128c04628107a36f83df21b33d12ff8db
- 639eb0d2c2da5487412e7891638b334927232ff270781fad81dc5371f44f7c8e
- 553d76d0c449377be550570e65e2bcae4371964fc3b539a1e1022d80699da5db
- a7993775f4518c6c68db08e226c11e51f9bc53314e4ff9385269baac582e2528
- 7ddce5be3642b66c7559821e26877c9f0242c748da64b2e68a81844bb1a6b148
- 84e0a543df302b18f1188139160fc5a8bd669da071e492453d5d6756064ee568
- 97b76d61941b790deff9f025dec55484e32ebff32b1b6e173d6fbf42cd8996ef
- bf6a5e37097330d7d68b6ac3deb6a10a1d3269be575fd51315774d1e7e1eca34
- a62a81785714844a099a918c66df9367b5eb14df06e589d59bc81f392358c5cc
- 920309f3822f993afeaa8ec70b4ef6b43dd2562be85cc2985efedc6cda2e7578
- 421c4b4b53d291da2b53c068a491b3913d92fe0eb6f330861e7b60f3d9f8eee7
- 87fae395c0e9ce3631dece94971befa578623ff0540d06539f583df921568814
- 4b8bde867c06b617d731ea9e965bf64800330701942324e475b8119352122e7c
- 3c6a8132df3351e2b7d186d0b3f41847e6920ebcb940548e3c9ed274901104c2
- 76cbb0abd9511aab2cc9dda993e3b9ab77afb09d2959f143647065ca47e725cc
- ed1b4a03595c59e5a90dd4f02f1993a2c5a43ca46a33aab0d15a1bbb1f8b3d30
- c44bac8b66ad11756b4c5ff3b1cd7e1187c634088f9e7aa2250067033df24e8d
- 63dfdb4927c0bca64f8952904f463330360eb052f2a2a749bf91a851a2be89b4
- 373c820cc395ea5b9c6f38b9470913e6684e8afea59e9dfeb3da490014074bf1
- b263df6b58c9259000e45a238327de8c07e79f2e7462c2b687c1c5771bac1dd5
- f05bc36211301087e403df09daa014ea8f04f5bdae5cef75eb866b56b82af2d6
- c45d3b6d2237fc500688a73d3ba18335d0002917f1a1f09df6934c87deaa097f
- fcad234dc2ad5e2d8215bcf6caac29aef62666c34564e723fa6d2eee8b6468ed
- e05b7f44ef8d0b58cfc2f407b84dcff1cb24e0ec392f792a49ad71e7eab39143
- 87c9bede1feac2e3810f3d269b4492fe0902e6303020171e561face400e9bdb4
- c3de728850dc1e777ad50a211a4be212ca6c4ac9d94bf7bb6d5f7fe5f4574021
- e5daa86418ac444d590a2c693cd7749d87134c47d8e0dbac30c69f23a8e8131f

SHA256 Hashes for Katz Stealer

- a6b736988246610da83ce17c2c15af189d3a3a4f82233e4fedfabdcbbde0cff0
- 74052cf53b45399b31743a6c4d3a1643e125a277e4ddcfcad4f2903b32bc7dc4
- 20bde6276d6355d33396d5ebfc523b4f4587f706b599573de78246811aabd33c
- e345d793477abbecc2c455c8c76a925c0dfe99ec4c65b7c353e8a8c8b14da2b6
- 96ada593d54949707437fa39628960b1c5d142a5b1cb371339acc8f86dbc7678
- 925e6375deaa38d978e00a73f9353a9d0df81f023ab85cf9a1dc046e403830a8
- b249814a74dff9316dc29b670e1d8ed80eb941b507e206ca0dfdc4ff033b1c1f
- 9b6fb4c4dd2c0fa86bffb4c64387e5a1a90adb04cb7b5f7e39352f9eae4b93fa
- d5ead682c9bed748fd13e3f9d0b7d7bacaf4af38839f2e4a35dc899ef1e261e2
- ece74382ec6f319890e24abbf8e0a022d0a4bd7e0aeaf13c20bab3a37035dcd1

- 2dba8e38ac557374ae8cbf28f5be0541338afba8977fbff9b732dee7cee7b43e
- 11e90765640cbb12b13afa1bcec31f96f50578a5e65e2aa7be24465001b92e41
- b2245ca7672310681caa52dc72e448983d921463c94cdab0ba9c40ad6b2a58fe
- c929ee54bdd45df0fa26d0e357ba554ef01159533501ec40f003a374e1e36974
- c0e3c93c59b45e47dda93438311f50ddb95808fd615a467285c9c359bce02cf0
- 309da3c8422422089b7f9af3b1b3f89e2d5c36e48e4d9d9faa07affb7d9a7b17
- fdc86a5b3d7df37a72c3272836f743747c47bfbc538f05af9ecf78547fa2e789
- 25b1ec4d62c67bd51b43de181e0f7d1bda389345b8c290e35f93ccb444a2cf7a
- 964ec70fc2fdf23f928f78c8af63ce50aff058b05787e43c034e04ea6cbe30ef
- d92bb6e47cb0a0bdbb51403528ccfe643a9329476af53b5a729f04a4d2139647
- 5dd629b610aee4ed7777e81fc5135d20f59e43b5d9cc55cdad291fcf4b9d20eb
- b912f06cf65233b9767953ccf4e60a1a7c262ae54506b311c65f411db6f70128
- 2852770f459c0c6a0ecfc450b29201bd348a55fb3a7a5ecdcc9986127fdb786b