DPRK Adopts EtherHiding: Nation-State Malware Hiding on Blockchains

By Mandiant

Published: 2025-10-16 · Archived: 2026-04-02 11:14:13 UTC

Written by: Blas Kojusner, Robert Wallace, Joseph Dobson

Google Threat Intelligence Group (GTIG) has observed the North Korea (DPRK) threat actor UNC5342 using ‘EtherHiding’ to deliver malware and facilitate cryptocurrency theft, the first time GTIG has observed a nation-state actor adopting this method. This post is part of a two-part blog series on adversaries using EtherHiding, a technique that leverages transactions on public blockchains to store and retrieve malicious payloads—notable for its resilience against conventional takedown and blocklisting efforts. Read about UNC5142 campaign leveraging EtherHiding to distribute malware.

Since February 2025, GTIG has tracked UNC5342 incorporating EtherHiding into an ongoing social engineering campaign, dubbed Contagious Interview by Palo Alto Networks. In this campaign, the actor uses JADESNOW malware to deploy a JavaScript variant of INVISIBLEFERRET, which has led to numerous cryptocurrency heists.

How EtherHiding Works

EtherHiding emerged in September 2023 as a key component in the financially motivated CLEARFAKE campaign (UNC5142), which uses deceptive overlays, like fake browser update prompts, to manipulate users into executing malicious code.

EtherHiding involves embedding malicious code, often in the form of JavaScript payloads, within a smart contract on a public blockchain like BNB Smart Chain or Ethereum. This approach essentially turns the blockchain into a decentralized and highly resilient command-and-control (C2) server.

The typical attack chain unfolds as follows:

1. **Initial Compromise:** DPRK threat actors typically utilize social engineering for their initial compromise (e.g., fake job interviews, crypto games, etc.). Additionally, in the CLEARFAKE campaign, the attacker first gains access to a legitimate website, commonly a WordPress site, through vulnerabilities or stolen credentials.
2. **Injection of a Loader Script:** The attacker injects a small piece of JavaScript code, often referred to as a "loader," into the compromised website.
3. **Fetching the Malicious Payload:** When a user visits the compromised website, the loader script executes in their browser. This script then communicates with the blockchain to retrieve the main malicious payload stored in a remote server. A key aspect of this step is the use of a read-only function call (such as `eth_call`), which does not create a transaction on the blockchain. This ensures the retrieval of the malware is stealthy and avoids transaction fees (i.e. gas fees).
4. **Payload Execution:** Once fetched, the malicious payload is executed on the victim's computer. This can lead to various malicious activities, such as displaying fake login pages, installing information-stealing malware, or deploying ransomware.

Advantages for Attackers

EtherHiding offers several significant advantages to attackers, positioning it as a particularly challenging threat to mitigate:

- **Decentralization and Resilience:** Because malicious code is stored on a decentralized and permissionless blockchain, there is no central server that law enforcement or cybersecurity firms can take down. The malicious code remains accessible as long as the blockchain itself is operational.
- **Anonymity:** The pseudonymous nature of blockchain transactions makes it difficult to trace the identity of the attackers who deployed the smart contract.

- **Immutability:** Once a smart contract is deployed, the malicious code within it typically cannot be easily removed or altered by anyone other than the contract owner.
- **Stealth:** Attackers can retrieve the malicious payload using read-only calls that do not leave a visible transaction history on the blockchain, making their activities harder to track.
- **Flexibility:** The attacker who controls the smart contract can update the malicious payload at any time. This allows them to change their attack methods, update domains, or deploy different types of malware to compromised websites simultaneously by simply updating the smart contract.

In essence, EtherHiding represents a shift toward next-generation bulletproof hosting, where the inherent features of blockchain technology are repurposed for malicious ends. This technique underscores the continuous evolution of cyber threats as attackers adapt and leverage new technologies to their advantage.

DPRK Social Engineering Campaign

North Korea's social engineering campaign is a sophisticated and ongoing cyber espionage and financially motivated operation that cleverly exploits the job application and interview process. This campaign targets developers, particularly in the cryptocurrency and technology sectors, to steal sensitive data, cryptocurrency, and gain persistent access to corporate networks.

The campaign has a dual purpose that aligns with North Korea's strategic goals:

- **Financial Gain:** A primary objective is the theft of cryptocurrency and other financial assets to generate revenue for the regime, helping it bypass international sanctions.
- **Espionage:** By compromising developers, the campaign aims to gather valuable intelligence and potentially gain a foothold in technology companies for future operations.

The campaign is characterized by its elaborate social engineering tactics that mimic legitimate recruitment processes.

1. The Phishing Lure:

- **Fake Recruiters and Companies:** The threat actors create convincing but fraudulent profiles on professional networking sites like LinkedIn and job boards. They often impersonate recruiters from well-known tech or cryptocurrency firms.
- **Fabricated Companies:** In some instances, they have gone as far as setting up fake company websites and social media presences for entities like "BlockNovas LLC," "Angeloper Agency," and "SoftGlideLLC" to appear legitimate.
- **Targeted Outreach:** They aggressively contact potential victims, such as software and web developers, with attractive job offers.

2. The Interview Process:

- **Initial Engagement:** The fake recruiters engage with candidates, often moving the conversation to platforms like Telegram or Discord.
- **The Malicious Task:** The core of the attack occurs during a technical assessment phase. Candidates are asked to perform a coding test or review a project, which requires them to download files from repositories like GitHub. These files contain malicious code.
- **Deceptive Tools:** In other variations, candidates are invited to a video interview and are prompted with a fake error message (a technique called ClickFix) that requires them to download a supposed "fix" or a specific software to proceed, which is actually the malware.

3. The Infection Chain:

The campaign employs a multi-stage malware infection process to compromise the victim's system, often affecting Windows, macOS, and Linux systems.

- **Initial Downloader (**e.g., JADESNOW**):** The malicious packages downloaded by the victim are often hosted on the npm (Node Package Manager) registry. These loaders may collect initial system information and download the next

stage of malware.

- **Second-Stage Malware (**e.g., BEAVERTAIL, JADESNOW**):** The JavaScript-based malware is designed to scan for and exfiltrate sensitive data, with a particular focus on cryptocurrency wallets, browser extension data, and credentials. The addition of JADESNOW to the attack chain marks UNC5342's shift towards EtherHiding to serve up the third-stage backdoor INVISIBLEFERRET.
- **Third-Stage Backdoor (**e.g., INVISIBLEFERRET**):** For high-value targets, a more persistent backdoor is deployed. INVISIBLEFERRET, a Python-based backdoor, provides the attackers remote control over the compromised system, allowing for long-term espionage, data theft, and lateral movement within a network.

JADESNOW

JADESNOW is a JavaScript-based downloader malware family associated with the threat cluster UNC5342. JADESNOW utilizes EtherHiding to fetch, decrypt, and execute malicious payloads from smart contracts on the BNB Smart Chain and Ethereum. The input data stored in the smart contract may be Base64-encoded and XOR-encrypted. The final payload in the JADESNOW infection chain is usually a more persistent backdoor like INVISIBLEFERRET.JAVASCRIPT.

The deployment and management of JADESNOW differs from that of similar campaigns that implement EtherHiding, such as CLEARFAKE**.** The CLEARFAKE campaign, associated with the threat cluster UNC5142, functions as a malicious JavaScript framework and often masquerades as a Google Chrome browser update pop-up on compromised websites. The primary function of the embedded JavaScript is to download a payload after a user clicks the "Update Chrome" button. The second-stage payload is another Base64-encoded JavaScript stored on the BNB Smart Chain. The final payload may be bundled with other files that form part of a legitimate update, like images or configuration files, but the malware itself is usually an infostealer like LUMASTEALER.

Figure 1 presents a general overview of the social engineering attack chain. The victim receives a malicious interview question, deceiving the victim into running code that executes the initial JavaScript downloader that interacts with a malicious smart contract and downloads the second-stage payload. The smart contract hosts the JADESNOW downloader that interacts with Ethereum to fetch the third-stage payload, in this case INVISIBLEFERRET.JAVASCRIPT. The payload is run in memory and may query Ethereum for an additional credential stealer component. It is unusual to see a threat actor make use of multiple blockchains for EtherHiding activity; this may indicate operational compartmentalization between teams of North Korean cyber operators. Lastly, campaigns frequently leverage EtherHiding's flexible nature to update the infection chain and shift payload delivery locations. In one transaction, the JADESNOW downloader can switch from fetching a payload on Ethereum to fetching it on the BNB Smart Chain. This switch not only complicates analysis but also leverages lower transaction fees offered by alternate networks.

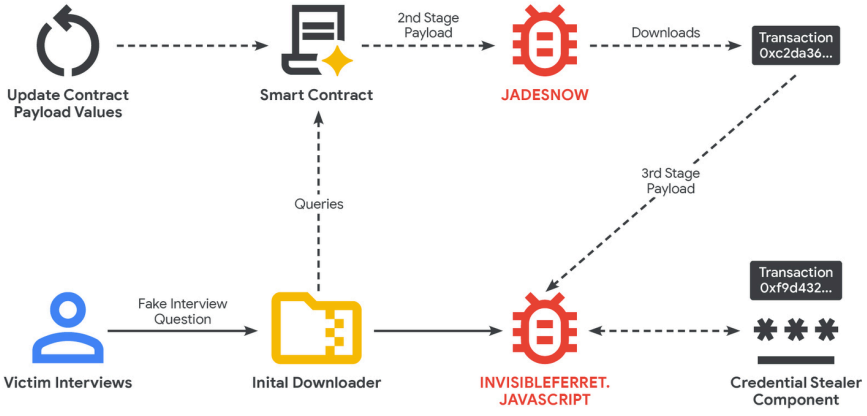

Figure 1: UNC5342 EtherHiding on BNB Smart Chain and Ethereum

Malicious Smart Contracts

BNB Smart Chain and Ethereum are both designed to run decentralized applications (dApps) and smart contracts. A smart contract is code on a blockchain that automatically executes actions when certain conditions or agreements are met, enabling secure, transparent, and automated agreements without intermediaries. Smart contracts are compiled into bytecode and uploaded to the blockchain, making them publicly available to be disassembled for analysis.

BNB Smart Chain, like Ethereum, is a decentralized and permissionless blockchain network that supports smart contracts programmed for the Ethereum Virtual Machine (EVM). Although smart contracts offer innovative ways to build decentralized applications, their unchangeable nature is leveraged in EtherHiding to host and serve malicious code in a manner that cannot be easily blocked.

Making use of Ethereum and BNB Smart Chain for the purpose of EtherHiding is straightforward since it simply involves calling a custom smart contract on the blockchain. UNC5342's interactions with the blockchain networks are done through centralized API service providers rather than Remote Procedure Call (RPC) endpoints, as seen with CLEARFAKE. When contacted by GTIG, responsible API service providers were quick to take action against this malicious activity; however, several other platforms have remained unresponsive. This indifference and lack of collaboration is a significant concern, as it increases the risk of this technique proliferating among threat actors.

JADESNOW On-Chain Analysis

The initial downloader queries the BNB Smart Chain through a variety of API providers, including Binplorer, to read the JADESNOW payload stored at the smart contract at address `0x8eac3198dd72f3e07108c4c7cff43108ad48a71c`.

Figure 2 is an example of an API call to read data stored in the smart contract from the transaction history. The transaction details show that the contract has been updated over 20 times within the first four months, with each update costing an average of $1.37 USD in gas fees. The low cost and frequency of these updates illustrate the attacker's ability to easily change the campaign's configuration. This smart contract has also been linked to a software supply chain attack that impacted React Native Aria and GlueStack via compromised npm packages in June 2025

```
{
   timestamp: 1738949853,
   transactionHash: "0x5c77567fcf00c317b8156df8e00838105f16fdd4fbbc6cd83d624225397d8856",
   tokenInfo: {
       address: "0x8eac3198dd72f3e07108c4c7cff43108ad48a71c",
       (...)
       owner: "0x9bc1355344b54dedf3e44296916ed15653844509",
       (...)
       txsCount: 22,
       (...)
   },
   type: "issuance",
   value: "1",
   priority: 127,
   address: "0x9bc1355344b54dedf3e44296916ed15653844509"
}
```

Figure 2: ABI call for transaction history

Blockchain explorers like BscScan (for BNB Smart Chain) and Etherscan (for Ethereum) are essential tools for reviewing on-chain information like smart contract code and historic transactions to and from the contract. These transactions may include input data such as a variable `Name`, its `Type`, and the `Data` stored in that variable. Figure 3 shows on-chain activity at the transaction address `0x5c77567fcf00c317b8156df8e00838105f16fdd4fbbc6cd83d624225397d8856`, where the `Data` field contains a Base64-encoded and XOR-encrypted message. This message decrypts to a heavily obfuscated JavaScript payload that GTIG assesses as the second-stage downloader, JADESNOW.

#	Name	Type	Data
0	to	address	0x9BC1355344B54DEDf3E44296916eD15653844509
1	amount	uint256	1
2	text	string	..QxpLV1oHNkRWSzlQWFAiNE0EQQ5cBwIBRQA/SnVYPCVRGkESZlYADEAZCAgRQzIzBBcUVxcKXQIfRAIYRFYnKEsYDBxAHQwRBDgeNRcSdG1 1F0gIDVtZThFVV1ZMQD0iUB9LWxsvLhEMAE0WXFQhYUFLEQMNX1pWHFcSDFgVMHxIWEhQVQwZCx8aBR8KTW4aeU1CWklDGwJWTAtQGg48fUdN Sx4QQhYbfwM5UEYbMClFBGVBEwREHh8KCx8CQzIzBBkZBQAEUQAfA09GA04lIFZWQAheQUUMD15RXwMeeyQBRRAFDlJEWFINFk1eCDZrDBkPA w9ZREgMCUFeHwVgcQ1NUlRJSwZeQEkHVlxUIWFVS1AQWFAbAlZMClBSbjgcHw5/XmZWFThVMV8VcUQOfEpNQQgTD0YXDUlSVRkAY3kQTVkOSQ

Figure 3: UNC5342 on-chain activity

When comparing transactions, the launcher-related code remains intact, but the next stage payload is frequently updated with a new obfuscated payload. In this case, the obfuscated payload is run in memory and decrypts an array of strings that combine to form API calls to different transaction hashes on Ethereum. This pivot to a different network is notable. The attackers are not using an Ethereum smart contract to store the payload; instead, they perform a `GET` request to query the transaction history of their attacker-controlled address and read the `calldata` stored from transactions made to the well-known “burn” address `0x00⋯dEaD`.

Transaction Hash	Method	Block	Age	From		To	Amount	Txn Fee
0xcdb95d62b1...	Transfer	21795916	210 days ago	0x0f2746be...40eC7E5c3	IN	0x9BC13553...653844509	0.005 ETH	0.00003545
0xc2da361c40...	Transfer*	21795760	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.0005091
0x0493601b42...	Transfer*	21795460	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00085906
0xf93c0b6873f...	Transfer*	21795302	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00089686
0xe2a54e1d10...	Transfer*	21795284	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00093591
0x0f705fd19dd...	Transfer*	21795280	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00107926
0x1f62d5c5a93...	Transfer*	21795239	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00116115
0x8d31cc22ce...	Transfer*	21795119	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.0010676
0xf9d432745ea...	Transfer*	21795000	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.0006824
0x24622144ea...	Transfer*	21794921	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00049611
0x86d1a21fd15...	Transfer*	21794376	210 days ago	0x9BC13553...653844509	OUT	Null: 0x00...dEaD	0 ETH	0.00013408
0xdf5cd655ab5...	Transfer	21794361	210 days ago	0x0f2746be...40eC7E5c3	IN	0x9BC13553...653844509	0.01 ETH	0.00003211

Figure 4: On-chain transactions

The final address of these transactions is inconsequential since the malware only reads the data stored in the details of a transaction, effectively using the blockchain transaction as a Dead Drop Resolver. These transactions are generated frequently, showing how easily the campaign can be updated with a simple blockchain transaction, including changing the C2 server.

The in-memory payload fetches and evaluates the information stored on-chain by querying Ethereum via different blockchain explorer APIs. Multiple explorers are queried simultaneously (including Blockchair, Blockcypher, and Ethplorer), likely as a fail-safe way to ensure payload retrieval. The use of a free API key, such as `apiKey=freekey` offered by Ethplorer for development, is sufficient for the JADESNOW operation despite strict usage limits.

Payload Analysis

The third stage is the INVISIBLEFERRET.JAVASCRIPT payload stored at the Ethereum transaction address `0x86d1a21fd151e344ccc0778fd018c281db9d40b6ccd4bdd3588cb40fade1a33a`. This payload connects to the C2 server via port 3306, the default port for MySQL. It sends an initial beacon with the victim's hostname, username, operating system, and the directory the backdoor is currently running under. The backdoor proceeds to run in the background, listening for incoming commands to the C2. The command handler is capable of processing arbitrary command execution, executing built-in commands to change the directory, and exfiltrating files, directories, and subdirectories from the victim’s system.

The INVISIBLEFERRET.JAVASCRIPT payload may also be split into different components like is done at the transaction address `0xc2da361c40279a4f2f84448791377652f2bf41f06d18f19941a96c720228cd0f`. The split up JavaScript payload executes the INVISIBLEFERRET.JAVASCRIPT backdoor and attempts to install a portable Python interpreter to execute an additional credential stealer component stored at the transaction address `0xf9d432745ea15dbc00ff319417af3763f72fcf8a4debedbfceeef4246847ce41`. This additional credential stealer component targets web browsers like Google Chrome and Microsoft Edge to exfiltrate stored passwords, session cookies, and credit cards. The INVISIBLEFERRET.JAVASCRIPT credential stealer component also targets cryptocurrency wallets like MetaMask and Phantom, as well as credentials from other sensitive applications like password managers (e.g., 1Password). The data is compressed into a ZIP archive and uploaded to an attacker-controlled remote server and a private Telegram chat.

The Centralized Dependencies in EtherHiding

Decentralization is a core tenet of blockchain networks and other Web3 technologies. In practice, however, centralized services are often used, which introduces both opportunities and risks. Though blockchains like BNB Smart Chain are immutable and permissionless and the smart contracts deployed onto such blockchains cannot be removed, operations by threat actors using these blockchains are not unstoppable.

Neither North Korea's UNC5342 nor threat actor UNC5142 are interacting directly with BNB Smart Chain when retrieving information from smart contracts; both threat actors are utilizing centralized services, akin to using traditional Web2 services such as web hosting. This affords astute defenders the opportunity to mitigate such threats. These centralized intermediaries represent points of observation and control, where traffic can be monitored and malicious activity can be addressed through blocking, account suspensions, or other methods. In other words, UNC5142 and UNC5342 are using permissioned services to interact with permissionless blockchains.

These threat actors exhibit two different approaches to utilizing centralized services for interfacing with blockchain networks:

1. An RPC endpoint is used by UNC5142 (CLEARFAKE) in the EtherHiding activity. This allows direct communication with a BNB Smart Chain node hosted by a third party in a manner that is close to a blockchain node's "native tongue."
2. An API service hosted by a central entity is used by UNC5342 (DPRK), acting as a layer of abstraction between the threat actor and the blockchain.

Though the difference is nuanced, these intermediary services are positioned to directly impact threat actor operations. Another approach not observed in these operations is to operate a node that integrates fully with the blockchain network. Running a full node is resource-intensive, slow to sync, and creates a significant hardware and network footprint that can be traced, making it a cumbersome and risky tool for cyber operations.

Recommendations

EtherHiding presents new challenges as traditional campaigns have usually been halted by blocking known domains and IPs. Malware authors may leverage the blockchain to perform further malware propagation stages since smart contracts operate autonomously and cannot be shut down.

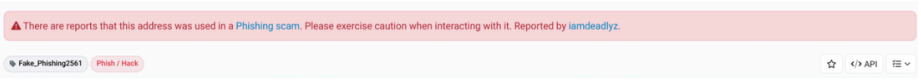

Figure 5: BscScan warning message

While security researchers attempt to warn the community by tagging a contract as malicious on official blockchain scanners (like the warning on BscScan in Figure 5), malicious activity can still be performed.

Chrome Enterprise: Centralized Mitigation

Chrome Enterprise can be a powerful tool to prevent the impact of EtherHiding by using its centralized management capabilities to enforce policies that directly disrupt the attack chain. This approach shifts security away from relying on individual user discretion and into the hands of a centralized, automated system.

The core strength of Chrome Enterprise resides in Chrome Browser Cloud Management. This platform allows administrators to configure and enforce security policies across all managed browsers in their organization, ensuring consistent protection regardless of the user's location or device.

For EtherHiding, this means an administrator can deploy a defense strategy that does not rely on individual users making the right security decisions.

Key Prevention Policies and Strategies

An administrator can use specific policies to break the EtherHiding attack at multiple points:

1. Block Malicious Downloads

This is the most direct and effective way to stop the attack. The final step of an EtherHiding campaign requires the user to download and run a malicious file (e.g., from a fake update prompt). Chrome Enterprise can prevent this entirely.

- **DownloadRestrictions Policy:** An admin can configure this policy to block downloads of dangerous file types. By setting this policy to block file types like `.exe`, `.msi`, `.bat`, and `.dll`, the malicious payload can not be saved to the user's computer, effectively stopping the attack.

2. Automate and Manage Browser Updates

EtherHiding heavily relies on social engineering, most notably by using a pop-up that tells the user "Your Chrome is out of date." In a managed enterprise environment, this should be an immediate red flag.

- **Managed Updates:** Administrators use Chrome Enterprise to control and automate browser updates. Updates are pushed silently and automatically in the background.
- **User Training:** Because updates are managed, employees can be trained with a simple, powerful message: "You will never be asked to manually update Chrome." Any prompt to do so is considered a scam and thus undermines the primary social engineering tactic.

3. Control Web Access and Scripts

While attackers constantly change their infrastructure, policies can still reduce the initial attack surface.

- **URLBlocklist** Policy**:** Admins can block access to known malicious websites, domains, or even the URLs of blockchain nodes if they are identified by threat intelligence.
- **Safe Browsing:** Policies can enforce Google's Safe Browsing in its most enhanced mode, which uses real-time threat intelligence to warn users about phishing sites and malicious downloads.

Acknowledgements

This analysis would not have been possible without the assistance from across Google Threat Intelligence Group, including the Koreas Mission, FLARE, and Advanced Practices.

Indicators of Compromise

Type	Indicator	Context
SHA256 Hash (ZIP Archive)	`970307708071c01d32ef542a49099571852846a980d6e8eb164d2578147a1628`	ZIP archive containing downloader, in this ca JADESNOW.
SHA256 Hash (Initial JavaScript Downloader)	`01fd153bfb4be440dd46cea7bebe8eb61b1897596523f6f6d1a507a708b17cc7`	JADESNOW sample infection chain.
BSC Address (Smart Contract)	`0x8eac3198dd72f3e07108c4c7cff43108ad48a71c`	BNB Smart Chain cor UNC5342 to host the JADESNOW payload
BSC Address (Attacker-Controlled)	`0x9bc1355344b54dedf3e44296916ed15653844509`	Owner address of the BNB Smart Chain cor
Ethereum Transaction Hash (INVISIBLEFERRET.JAVASCRIPT Payload)	`0x86d1a21fd151e344ccc0778fd018c281db9d40b6ccd4bdd3588cb40fade1a33a`	Transaction storing th INVISIBLEFERRET. payload.

Type	Indicator	Context
Ethereum Transaction Hash (INVISIBLEFERRET.JAVASCRIPT Split Payload)	`0xc2da361c40279a4f2f84448791377652f2bf41f06d18f19941a96c720228cd0f`	Transaction storing th INVISIBLEFERRET. payload
Ethereum Transaction Hash (INVISIBLEFERRET Credential Stealer Payload)	`0xf9d432745ea15dbc00ff319417af3763f72fcf8a4debedbfceeef4246847ce41`	Transaction storing th INVISIBLEFERRET. credential stealer payl

YARA Detections

```
rule G_Downloader_JADESNOW_1 {
        meta:
                author = "Google Threat Intelligence Group (GTIG)"
        strings:
                $s1 = "global['_V']"
                $s2 = "global['r']"
                $s3 = "umP"
                $s4 = "mergeConfig"
                $s5 = "charAt" nocase
        condition:
                uint16(0) != 0x5A4D and filesize < 10KB and #s3 > 2 and #s5 == 1 and all of them
}
```

Posted in

- Threat Intelligence
