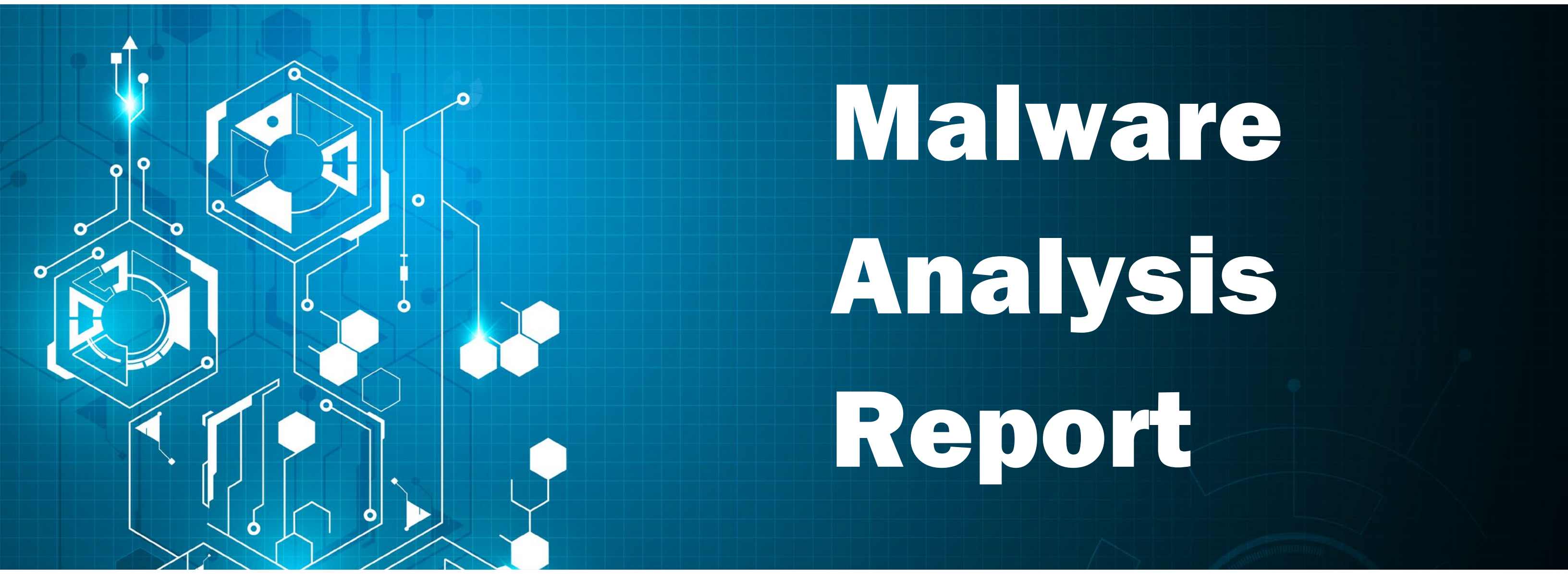
Communications Security Establishment Canada
Centre de la sécurité des télécommunications Canada

Canadian Centre for Cyber Security
Centre canadien pour la cybersécurité



BRICKSTORM Backdoor

Publication: December 4, 2025

U.S. Cybersecurity and Infrastructure Security Agency
U.S. National Security Agency
Canadian Centre for Cyber Security

*This document is marked TLP:CLEAR. Disclosure is not limited. Sources may use TLP:CLEAR when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:CLEAR information may be distributed without restriction. For more information, see [Traffic Light Protocol (TLP) Definitions and Usage](). *

Malware Analysis at a Glance

Executive Summary	The Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and Canadian Centre for Cyber Security (Cyber Centre) assess People's Republic of China (PRC) state-sponsored cyber actors are using BRICKSTORM malware for long-term persistence on victim systems. CISA, NSA, and Cyber Centre are releasing this Malware Analysis Report to share indicators of compromise (IOCs) and detection signatures based off analysis of eight BRICKSTORM samples. CISA, NSA, and Cyber Centre urge organizations to use the IOCs and detection signatures to identify BRICKSTORM malware samples.
Key Actions	▪ **Use the IOCs and detection signatures** to identify BRICKSTORM samples. ▪ If BRICKSTORM, similar malware, or potentially related activity is detected, **report the incident to CISA, Cyber Centre, or required authorities immediately.**
Indicators of Compromise	For a downloadable copy of IOCs associated with this malware, see: MAR-251165.c1.v1.CLEAR.
Detection	This malware analysis report includes YARA and Sigma rules. For a downloadable copy of the Sigma rules associated with this malware, see: AR25-338A Sigma YAML.
Intended Audience	**Organizations:** Government and critical infrastructure organizations. **Roles:** Digital forensics analysts, incident responders, vulnerability analysts, system administrators

Table of Contents

Introduction

The Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and Canadian Centre for Cyber Security (Cyber Centre) assess People's Republic of China (PRC) state-sponsored cyber actors are using BRICKSTORM malware for long-term persistence on victim systems. Victim organizations are primarily in the Government Services and Facilities and Information Technology Sectors. BRICKSTORM is a sophisticated backdoor for VMware vSphere (specifically VMware vCenter servers and VMware ESXI)[1] and Windows environments.[2]

The cyber actors have been observed targeting VMware vSphere platforms. Once compromised, the cyber actors can use their access to the vCenter management console to steal cloned virtual machine (VM) snapshots for credential extraction and create hidden, rogue VMs. See CISA's Alert PRC State-Sponsored APT Actors Employ BRICKSTORM Malware Across Public Sector and Information Technology.

CISA analyzed eight BRICKSTORM samples obtained from victim organizations, including an organization where CISA conducted an incident response engagement.

At the victim organization where CISA conducted an incident response engagement, PRC state-sponsored cyber actors gained long-term persistent access to the organization's internal network in April 2024 and uploaded BRICKSTORM malware to an internal VMware vCenter server. They also gained access to two domain controllers and an Active Directory Federation Services (ADFS) server. They successfully compromised the ADFS server and exported cryptographic keys. The cyber actors used BRICKSTORM for persistent access from at least April 2024 through at least Sept. 3, 2025.

CISA, NSA, and Cyber Centre urge organizations to use the indicators of compromise (IOCs) and detection signatures in this Malware Analysis Report to identify BRICKSTORM malware samples. If identified, follow the guidance in the **Incident Response** section.

For a downloadable copy of IOCs associated with this malware, see: MAR-251165.c1.v1.CLEAR.

For a downloadable copy of the SIGMA rule associated with this malware, see: AR25-338A Sigma YAML.

For more information on PRC state-sponsored cyber activity, see CISA's People's Republic of China Threat Overview and Advisories webpage.

Malware Summary

BRICKSTORM is a custom Executable and Linkable Format (ELF) Go-based backdoor. The analyzed samples differ in function, but all enable cyber actors to maintain stealthy access and provide capabilities for initiation, persistence, and secure command and control (C2). Even though the analyzed samples were for VMware vSphere environments, there is reporting about Windows versions.

BRICKSTORM initiates by running checks and maintains persistence by using a self-watching function and automatically reinstalls or restarts if disrupted.

For C2, BRICKSTORM uses multiple layers of encryption (HTTPS, WebSockets, nested Transport Layer Security [TLS]) to hide its communications with the cyber actors' C2 server. It also uses DNS-over-HTTPS (DoH) and mimics web server functionality to blend its communications with legitimate traffic. For remote system control, BRICKSTORM gives cyber actors interactive shell access on the system and allows them to browse, upload, download, create, delete, and manipulate files. In addition, some samples act as a SOCKS proxy, facilitating lateral movement and allowing cyber actors to compromise additional systems.

Malware Delivery

Note: This advisory uses the MITRE ATT&CK® Matrix for Enterprise framework, version 18. See **Appendix A: MITRE ATT&CK Techniques** for tables mapping the cyber actors' activity to MITRE ATT&CK tactics and techniques.

At the victim organization where CISA conducted an incident response engagement, PRC state-sponsored cyber actors accessed a web server on April 11, 2024. The web server was inside the organization's demilitarized zone (DMZ), and cyber actors accessed it through a web shell [T1505.003] present on the server. Incident data does not indicate how they obtained initial access to the web server or when the web shell was implanted. On the same day, the cyber actors used service account credentials [T1078] to move laterally using Remote Desktop Protocol (RDP) [T1021.001] from the web server to a domain controller in the DMZ, from which they copied the Active Directory (AD) database (`ntds.dit`) [T1003.003].

On April 12, 2024, the cyber actors moved laterally from the web server to a domain controller within the internal network using RDP and credentials associated with a second service account. It is unknown how they obtained the credentials. Subsequently, they copied the AD database, obtaining credentials for a managed service provider (MSP) account. Using the MSP credentials, the cyber actors proceeded to move from the internal domain controller to the VMware vCenter server.

From the web server, the actors also moved laterally using Server Message Block (SMB) to two jump servers and an ADFS server, from which they exfiltrated cryptographic keys. See **Figure 1** for a diagram of the cyber actors' movement.

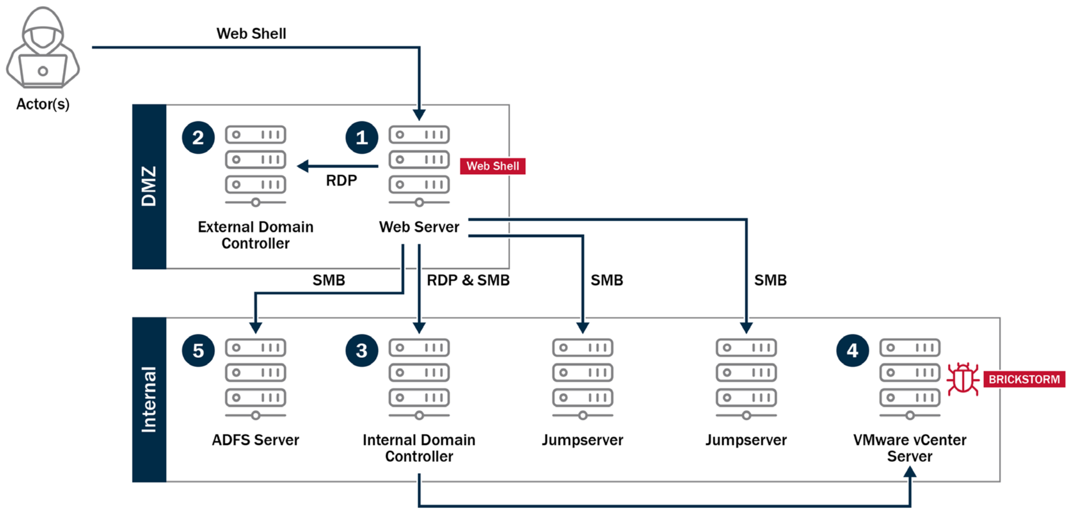


Figure 1. PRC State-Sponsored Cyber Actors' Lateral Movement

After gaining access to vCenter, the cyber actors elevated privileges using the `sudo` command [T1548.003], dropped BRICKSTORM malware in the server's `/etc/sysconfig/` directory [T1105], and modified the system's `init` file in `/etc/sysconfig/` to run BRICKSTORM.

The modified `init` file controls the bootup process [T1037] on VMware vSphere systems and executes BRICKSTORM. Typically, this file is used to define certain visual variables for the bootup process. After the setting for visual variables, an additional line was added to the script to execute BRICKSTORM from the hard-coded file path `/etc/sysconfig/`.

Note: CISA is still completing analysis to understand the malicious activity and full impact of the compromise.

Malware Metadata

See **Table 1** through **Table 8** for metadata of the analyzed malware.

Table 1. BRICKSTORM Sample 1

File Name	vmsrc
Size	7692288 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	8e4c88d00b6eb46229a1ed7001451320
SHA1	9bf4c786ebd68c0181cfe3eb85d2fd202ed12c54
SHA256	aaf5569c8e349c15028bc3fac09eb982efb06eabac955b705a6d447263658e38
SHA512	5e654776e9c419e11e6f93a452415a601bd9a2079710f1074608570e498a9af37b81bb57c98cb8bb626c5ee4b3e35757d3ae8c1c3717f28d9f3fe7a4cebe0608
ssdeep	49152:9IDeYNeYunc1S3/U05q+CIKUbwgBfd1Vww/uUJSZina/TokDDko0n8oQhEoAgsUJ:O3IcE380sIDbdB11p3i/TokEIowIb/r
Entropy	5.993799

Table 2. BRICKSTORM Sample 2

File Name	vnetd
Size	26603668 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped

MD5	39111508bfde89ce6e0fe6abe0365552
SHA1	f639d9404c03af86ce452db5c5e0c528b81dc0d7
SHA256	013211c56caaa697914b5b5871e4998d0298902e336e373ebb27b7db30917eaf
SHA512	74b4c6f7c7cae07c6f8edf3f2fb1e9206d4f1f9734e8e4784b15d192eec8cd8a4f59078fc0c56dc4ad0856cdd792337b5c92ffd3d2240c8a287a776df4363bba
ssdeep	196608:GbkKsdDjru3WUIOsW5SYVRk/Qvk1LzK3RMxy2wBW:GwKMjr3Os4k/QiLzERMMdW
Entropy	6.211446

Table 3. BRICKSTORM Sample 3

File Name	if-up
Size	15511700 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	dbca28ad420408850a94d5c325183b28
SHA1	fb11c6caa4ea844942fe97f46d7eb42bc76911ab
SHA256	57bd98dbb5a00e54f07ffacda1fea91451a0c0b532cd7d570e98ce2ff741c21d
SHA512	659205fa2cfa85e484c091cc2e85a7ec4e332b196e423b1f39bafdc8fca33e3db712bbe07afcc091ff26d9b4f641fa9a73f2a66dce9a0ced54ebeb8c2be82a7f
ssdeep	98304:dzB06b0KX4Mnb+sJf+AjBzH3MF4m1d4U2TuAJ5VGY3gIknTSk2nH:dFQKIsJBBzXMum83RJ5VGY3gS2nH
Entropy	6.102490

Table 4. BRICKSTORM Sample 4

File Name	viocli
Size	6311936 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	0a4fa52803a389311a9ddc49b7b19138

SHA1	97001baaa379bcd83677dca7bc5b8048fdfaaddc
SHA256	b3b6a992540da96375e4781afd3052118ad97cfe60ccf004d732f76678f6820a
SHA512	65ebf5dfafb8972ffead44271436ec842517cfaaf3d1f1f1237a32d66e1d280943bd3a69f1d539a1b7aca6152e96b29bc822e1047e2243f6aec8959595560147
ssdeep	49152:BgClz8/9cMSThwhWyh/zypzOzRzqm9hRp6FY+fAn/bkNqr+HfHF2xkdpb3gAiDIi:W08/9I6WMzUcRz9zvn//Z5D
Entropy	6.005898

Table 5. BRICKSTORM Sample 5

File Name	vts
Size	6303744 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	82bf31e7d768e6d4d3bc7c8c8ef2b358
SHA1	de28546ec356c566cd8bca205101a733e9a4a22d
SHA256	22c15a32b69116a46eb5d0f2b228cc37cd1b5915a91ec8f38df79d3eed1da26b
SHA512	4c52caf2e5f114103ed5f60c6add3aa26c741b07869bb66e3c25a1dc290d4a8bf87c42c336e8ac8ebf82d9a9b23eaa18c31f7051a5970a8fe1125a2da890340f
ssdeep	49152:uP9kPWdmrJI+9zxKsSJ32ssUZGHZ9ECKDfvCb3XKRbaYJcRHMH9xkdgY3gqF2HxR:yqWdmd4x5SgssUZOOCKDfvChYrRq
Entropy	6.005438

Table 6. BRICKSTORM Sample 6

File Name	vmckd
Size	6311936 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	18f895e24fe1181bb559215ff9cf6ce3
SHA1	c3549d4e5e39a11f609fc6fbf5cc1f2c0ec272b4

SHA256	f7cda90174b806a34381d5043e89b23ba826abcc89f7abd520060a64475ed506
SHA512	79276523a6a507e3fa1b12b96e09b10a01c783a53d58b9ae7f5780a379431639a8016 5e81154522649b8e2098e86d1a310efffebe32faafc7b3bc093eec60a64
ssdeep	49152:6XUQ9anktEg7z/QbPB83A+FQGQzqufqCjt2F81jh+eS530OwJylHJHuxkdqz3gHG:mVa nkxn2Pe3JQGQz57t2Y4f3TwrQHAz
Entropy	6.005752

Table 7. BRICKSTORM Sample 7

Size	8332689 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	a52e36a70b5e0307cbcaa5fd7c97882c
SHA1	44a3d3f15ef75d9294345462e1b82272b0d11985
SHA256	39b3d8a8aedffc1b40820f205f6a4dc041cd37262880e5030b008175c45b0c46
SHA512	bbe18d32bef66ccfa931468511e8ba55b32943e47a1df1e68bb5c8f8ae97a5bf99120185 8ae9632fa24df5f6c674b6cb260297a1c11889ca61bda68513f440ce
ssdeep	98304:78Se5IqfYMKDdopPx0E4j+dM/GLaCXNwqYL6wt/5APUnb:78Se54fYMUaiE4j+dM/GL aCXNmLP+
Entropy	6.063930

Table 8. BRICKSTORM Sample 8

Size	8332689 bytes
Type	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, stripped
MD5	a02469742f7b0bc9a8ab5e26822b3fa8
SHA1	10d811029f6e5f58cd06143d6353d3b05bc06d0f
SHA256	73fe8b8fb4bd7776362fd356fdc189c93cf5d9f6724f6237d829024c10263fe5
SHA512	8e29aeb3603ffe307b2d60f7401bd9978bebe8883235eb88052ebf6b9e04ce6bf356674 80cedea5712c1e13e8c6dcfb34d5fde0ddca6ca31328de0152509bf8f

ssdeep	98304:78Se5IqfYMKDdopPx0E4j+dM/GLaCXNwqYL6wt/5APUnU:78Se54fYMUaiE4j+dM/GLaCXNmLP+
Entropy	6.063928

Malware Functionality

All analyzed samples enable cyber actors to maintain stealthy access and provide capabilities for environment configuration (initiation), persistence, and secure C2. While initiation and persistence functions are similar across the samples, the secure C2 function varies. BRICKSTORM uses custom handlers to set up a SOCKS proxy, create a web server on the compromised system, and execute commands on the compromised system.

Samples 7 and 8 were designed to work in virtualized environments, using a virtual socket (VSOCK) interface to enable inter-VM communication, facilitate data exfiltration, and maintain persistence.

Most samples used Exclusive OR (XOR) cipher encryption to hide key strings, such as the Internet Protocol version 4 (IPv4) addresses of public DoH servers, within their code.

Initiation Capabilities

Upon execution, BRICKSTORM runs checks and can reinstall and restart itself to maintain persistence. BRICKSTORM initiates a function (referred to as `main_startNew` in some samples) to configure environment variables specific to the compromised environment, enabling it to operate effectively. Following this, BRICKSTORM identifies if it is already in its intended state and proceeds to continue running, copy itself for execution, or terminate based on the following logic:

1. **Environment Variable Check:** BRICKSTORM checks a specified environment variable (differs by sample; see **Table 9**) to determine if it is running as a child process (to identify if it is running in its intended state).
    a. If the specified variable is set, indicating it is running as a child process, BRICKSTORM continues its code execution.
    b. If the specified variable is not set (indicating it is not running as a child process), BRICKSTORM checks whether it is executing from `/etc/sysconfig/` (Samples 1 through 2 and 4 through 7) or `/etc/sysconfig/network/` (Sample 3) by attempting to load file contents from that path.
2. **File Path Validation and Copying:** If BRICKSTORM is running from the validated path, it copies itself to a specific location with a specific file name.
    a. Next, the parent BRICKSTORM instance modifies the `PATH` environment variable by appending the copied location's path [T1574.007]. This ensures the newly copied version of BRICKSTORM will be executed first if any commands or processes attempt to run VMware vSphere.
    b. The parent instance subsequently executes the copied instance of BRICKSTORM with the specified variable set in the context of the child process and terminates its own execution.

3. **Termination:** If BRICKSTORM is not running from the validated path, it terminates its own execution.

See **Figure 2** for the operational flow of the malware.

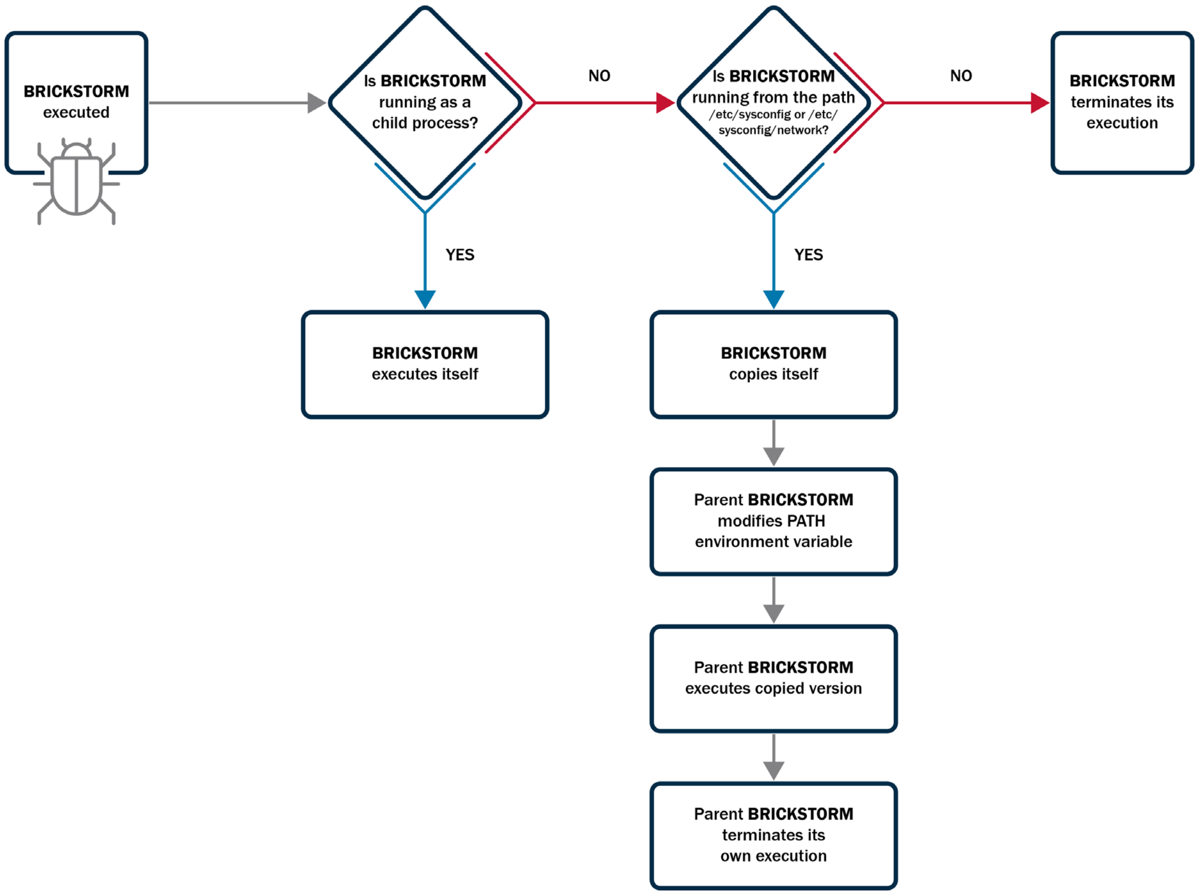

BRICKSTORM executed
Is BRICKSTORM running as a child process?
NO
Is BRICKSTORM running from the path /etc/sysconfig or /etc/sysconfig/network?
NO
BRICKSTORM terminates its execution
YES
YES
BRICKSTORM executes itself
BRICKSTORM copies itself
Parent BRICKSTORM modifies PATH environment variable
Parent BRICKSTORM executes copied version
Parent BRICKSTORM terminates its own execution

Figure 2. BRICKSTORM Operational Flow, Malware Initiation

See **Table 9** for checked variables, copied locations, and copied file names of the analyzed samples.

Table 9. BRICKSTORM Initiation Checks and Copied File Information

Sample	Checked Environment Variable To Determine if Running as a Child Process	Copied Location	Copied File Name
Sample 1	VMware [T1036]	/opt/vmware/sbin	vmware-sphere
Sample 2	[redacted]ET4	/usr/java/jre-vmware/bin/	updatemgr
Sample 3	CZePMeGj	etc/applmgmt/appliance/	vami

Sample	Checked Environment Variable To Determine if Running as a Child Process	Copied Location	Copied File Name
Sample 4	[redacted]NET6	/usr/java/jre-vmware/bin/	updatemgr
Sample 5	FIOON	/usr/java/jre-vmware/bin/	updatemgr
Sample 6	[redacted]NET4	/usr/java/jre-vmware/bin/	updatemgr
Sample 7	VREG		
Sample 8	VARGS		

Persistence Capabilities

To ensure its continued operations, BRICKSTORM uses built-in self-monitoring and persistence capabilities while running. Specifically, it has a built-in self-watching function (referred to as `main_selfWatcher` in some samples) to maintain persistence. This function monitors if BRICKSTORM is running correctly and, if not, BRICKSTORM reinstalls and executes itself, mirroring its initiation capabilities.

The self-watching function begins by checking a specific environment variable (see **Table 10**) to confirm whether BRICKSTORM is running as an active process. If the check returns a `false` value (indicating the variable is not set), BRICKSTORM assumes it is not running properly. In response, BRICKSTORM re-installs itself from predefined file path—`/etc/sysconfig/` (Samples 1 through 2 and 4 through 8) or `/etc/sysconfig/network/` (Sample 3)—to a new location (the file name of the new BRICKSTORM instance and location copied varies by sample; see **Table 10**). BRICKSTORM then updates the `PATH` environment variable to include the new file location, ensuring the newly copied backdoor file is executed first. Subsequently, the parent instance terminates its own execution, allowing the new process to take over.

If the initial checks confirm that BRICKSTORM is running as intended (the variable is set), the self-watcher function allows the code to continue its operations.

See **Table 10** for details on checked variables, processes, copied locations, and file names associated with the analyzed samples.

Table 10. BRICKSTORM Checked Variables, Processes, and Copied Names and Locations

Sample	Checked Environment Variable	Checked Process Existence	Copies To	Newly Copied File Name
Sample 1	Sphere	vmware-sphere	/opt/vmware/sbin/	vmware-sphere
Sample 2	[redacted]NET3	vnetd	/usr/java/jre-vmware/bin/	updatemgr

Sample	Checked Environment Variable	Checked Process Existence	Copies To	Newly Copied File Name
Sample 3	rcMJVF	vami	/etc/applmgmt/appliance/	vami
Sample 4	[redacted]NET5	updatemgr	/usr/java/jre-vmware/bin/	updatemgr
Sample 5	DIGNN	updatemgr	/usr/java/jre-vmware/bin/	updatemgr
Sample 6	[redacted]NET3	updatemgr	/usr/java/jre-vmware/	updatemgr
Sample 7	VREG			
Sample 8	VARGS			

Secure Command and Control

After passing initiation checks, BRICKSTORM establishes a connection to a C2 server, secures communications with the server, and enables cyber actors' full control over the compromised system. This control includes capabilities such as file system management and interactive shell access. In most samples, BRICKSTORM also provides a SOCKS proxy to facilitate tunneling and lateral movement.

The implementation of these capabilities varies across samples, with notable differences in Samples 7 and 8, which specifically target virtualized environments.

Sample 1

Initial Connection to the C2 Server: Sample 1 first creates an encrypted Domain Name System (DNS) query for a hard-coded C2 domain (the domain has been redacted from this report because according to public reporting, the cyber threat actors are not reusing C2 domains).[3] The sample uses DoH to resolve the address of its C2 servers by sending an encrypted HTTPS request to one of the following legitimate public DoH resolvers [T1071.001]:

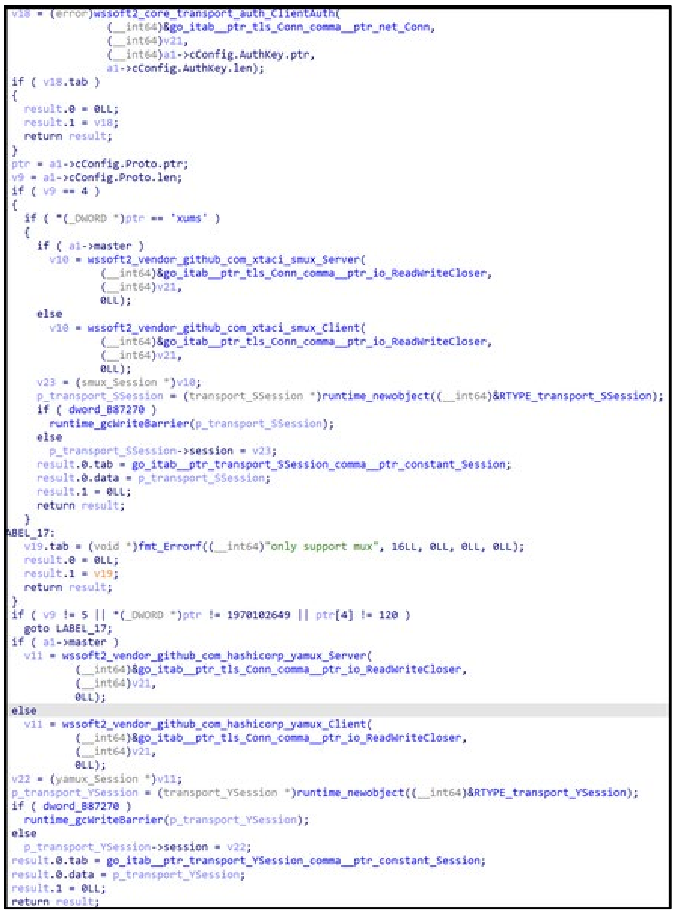
- https[:]//1.0.0[.]1/dns-query (Cloudflare)
- https[:]//1.1.1[.]1/dns-query (Cloudflare)
- https[:]//8.8.4[.]4/dns-query (Google)
- https[:]//8.8.8[.]8/dns-query (Google)
- https[:]//9.9.9[.]9/dns-query (Quad9)

If the C2 domain is not found in the public DoH resolver cache, the legitimate resolver forwards the request to the next server in the DNS hierarchy, ultimately reaching the threat actors' DNS server. The DNS server responds with the correct IP address for the domain. The response is sent back through the legitimate DoH resolver to BRICKSTORM, which receives the encrypted response, decrypts it to get the C2 server's IP address, and establishes a connection.

Establishing Secure Communications: Sample 1 establishes an encrypted connection to the C2 server using HTTPS, then upgrades the session to WebSockets with an additional layer of TLS encryption. To do this, Sample 1 first communicates over HTTPS with a specific legitimate cloud platform (redacted). The sample then sends an HTTP upgrade request to convert the initial encrypted HTTPS connection into a persistent WebSocket connection: `wss://[REDACTED].com/api`. Sample 1 nests additional layers of TLS encryption within the WebSocket session and performs a series of nested TLS handshakes within the established WebSocket tunnel. The first handshake is the standard TLS handshake for the initial HTTPS request to the cloud platform. The second TLS handshake occurs within the WebSocket tunnel, during which BRICKSTORM authenticates itself to the C2 server using a hard-coded key.

Upon successful authentication, BRICKSTORM establishes a multiplexing layer, which allows it to send multiple commands and data streams over the same connection. It does this using both Simple Multiplexing (`smux`) and Yet Another Multiplexer (`Yamux`) libraries to create virtual streams over a single underlying TLS-secured connection based on client configuration or handshake data. Multiplexing conceals threat actor activity by embedding multiple commands and network tunnels within a single encrypted stream.

See **Figure 3** for the applicable decompiler output.



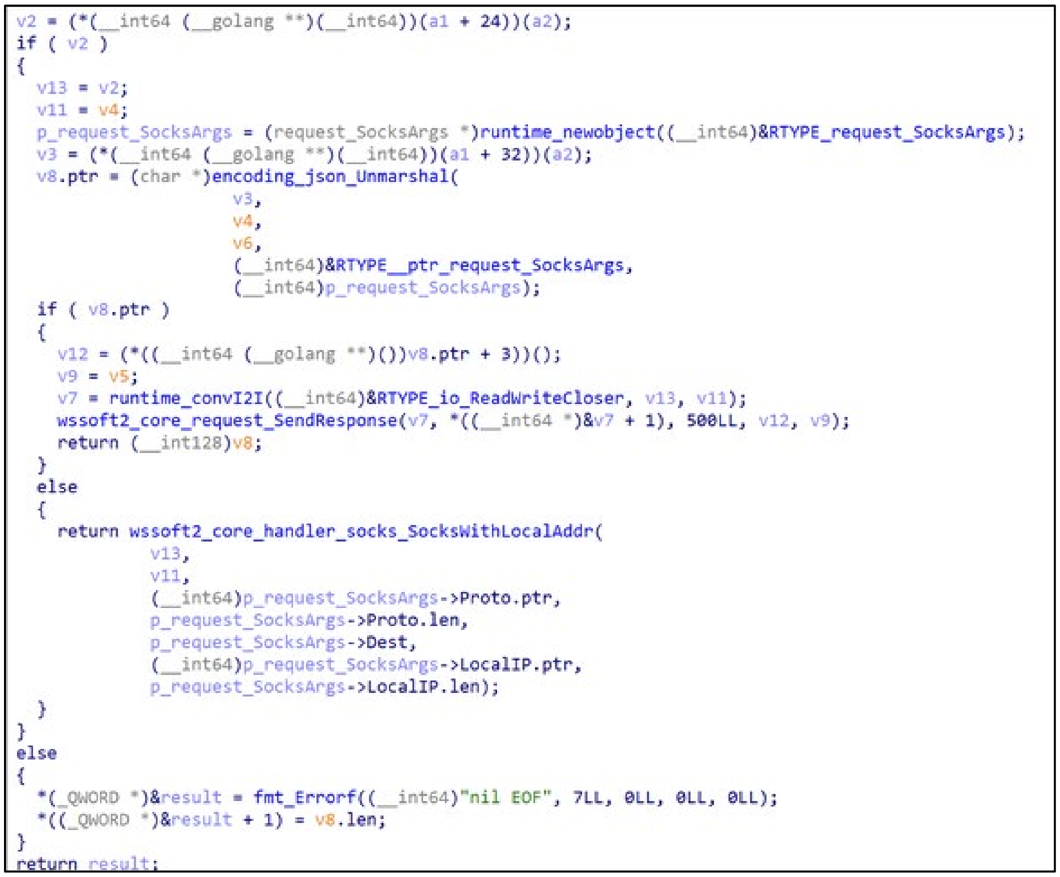
```
v18 = (error)wssoft2_core_transport_auth_ClientAuth(
               (__int64)&go_itab__ptr_tls_Conn_comma__ptr_net_Conn,
               (__int64)v21,
               (__int64)a1->cConfig.AuthKey.ptr,
               a1->cConfig.AuthKey.len);
if ( v18.tab )
{
  result.0 = 0LL;
  result.1 = v18;
  return result;
}
ptr = a1->cConfig.Proto.ptr;
v9 = a1->cConfig.Proto.len;
if ( v9 == 4 )
{
  if ( *(_DWORD *)ptr == 'xums' )
  {
    if ( a1->master )
      v10 = wssoft2_vendor_github_com_xtaci_smux_Server(
              (__int64)&go_itab__ptr_tls_Conn_comma__ptr_io_ReadWriteCloser,
              (__int64)v21,
              0LL);
    else
      v10 = wssoft2_vendor_github_com_xtaci_smux_Client(
              (__int64)&go_itab__ptr_tls_Conn_comma__ptr_io_ReadWriteCloser,
              (__int64)v21,
              0LL);
    v23 = (smux_Session *)v10;
    p_transport_SSession = (transport_SSession *)runtime_newobject((__int64)&RTYPE_transport_SSession);
    if ( dword_B87270 )
      runtime_gcWriteBarrier(p_transport_SSession);
    else
      p_transport_SSession->session = v23;
    result.0.tab = go_itab__ptr_transport_SSession_comma__ptr_constant_Session;
    result.0.data = p_transport_SSession;
    result.1 = 0LL;
    return result;
  }
ABEL_17:
  v19.tab = (void *)fmt_Errorf((__int64)"only support mux", 16LL, 0LL, 0LL, 0LL);
  result.0 = 0LL;
  result.1 = v19;
  return result;
}
if ( v9 != 5 || *(_DWORD *)ptr != 1970102649 || ptr[4] != 120 )
  goto LABEL_17;
if ( a1->master )
  v11 = wssoft2_vendor_github_com_hashicorp_yamux_Server(
          (__int64)&go_itab__ptr_tls_Conn_comma__ptr_io_ReadWriteCloser,
          (__int64)v21,
          0LL);
else
  v11 = wssoft2_vendor_github_com_hashicorp_yamux_Client(
          (__int64)&go_itab__ptr_tls_Conn_comma__ptr_io_ReadWriteCloser,
          (__int64)v21,
          0LL);
v22 = (yamux_Session *)v11;
p_transport_YSession = (transport_YSession *)runtime_newobject((__int64)&RTYPE_transport_YSession);
if ( dword_B87270 )
  runtime_gcWriteBarrier(p_transport_YSession);
else
  p_transport_YSession->session = v22;
result.0.tab = go_itab__ptr_transport_YSession_comma__ptr_constant_Session;
result.0.data = p_transport_YSession;
result.1 = 0LL;
return result;
```

Figure 3. BRICKSTORM Decompiler Output for Establishing Secure Connections

Full System Control: Once the secure connection to the C2 domain is established, Sample 1 uses a custom Go package `wssoft2` to manage incoming network connections and to process commands it receives. Commands are directed to one of three handlers based on the function it needs: SOCKS Handler, Web Service Handler, and Command Handler.

The SOCKS Handler sets up a SOCKS proxy [T1090.001] to route C2 traffic and facilitate lateral movement within the victim network. To set up the proxy, the handler parses JSON requests from the C2 server. If the request is valid, the handler delegates request handling to `wssoft2/core/handler/socks.SocksWithLocalAddr`, which performs SOCKS relaying and network tunneling over Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

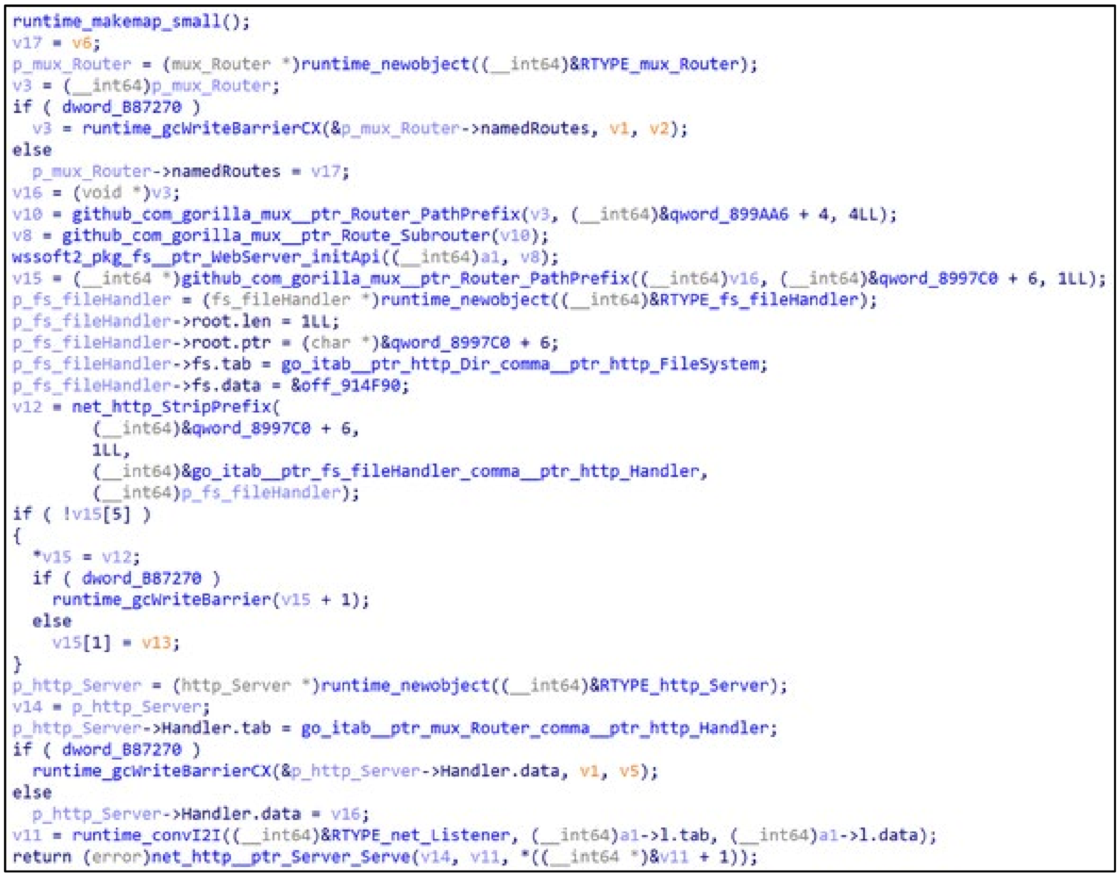
See **Figure 4** for the handler's decompiler output.



```
v2 = (*(__int64 (__golang **)(__int64))(a1 + 24))(a2);
if ( v2 )
{
  v13 = v2;
  v11 = v4;
  p_request_SocksArgs = (request_SocksArgs *)runtime_newobject((__int64)&RTYPE_request_SocksArgs);
  v3 = (*(__int64 (__golang **)(__int64))(a1 + 32))(a2);
  v8.ptr = (char *)encoding_json_Unmarshal(
                     v3,
                     v4,
                     v6,
                     (__int64)&RTYPE__ptr_request_SocksArgs,
                     (__int64)p_request_SocksArgs);
  if ( v8.ptr )
  {
    v12 = (*((__int64 (__golang **)())v8.ptr + 3))();
    v9 = v5;
    v7 = runtime_convI2I((__int64)&RTYPE_io_ReadWriteCloser, v13, v11);
    wssoft2_core_request_SendResponse(v7, *((__int64 *)&v7 + 1), 500LL, v12, v9);
    return (__int128)v8;
  }
  else
  {
    return wssoft2_core_handler_socks_SocksWithLocalAddr(
             v13,
             v11,
             (__int64)p_request_SocksArgs->Proto.ptr,
             p_request_SocksArgs->Proto.len,
             p_request_SocksArgs->Dest,
             (__int64)p_request_SocksArgs->LocalIP.ptr,
             p_request_SocksArgs->LocalIP.len);
  }
}
else
{
  *(_QWORD *)&result = fmt_Errorf((__int64)"nil EOF", 7LL, 0LL, 0LL, 0LL);
  *((_QWORD *)&result + 1) = v8.len;
}
return result;
```

Figure 4. SOCKS Handler Decompiler Output

The Web Service Handler establishes covert C2 communication by creating a legitimate-appearing web server on the compromised system. It uses the `net/http package` and `gorilla/mux` library to create the web server, which includes a hidden Application Programming Interface (API) endpoint for receiving and executing commands from the C2 server. See **Figure 5** for the Web Service Handler decompiler output that sets up specific API endpoints.



```
runtime_makemap_small();
v17 = v6;
p_mux_Router = (mux_Router *)runtime_newobject((__int64)&RTYPE_mux_Router);
v3 = (__int64)p_mux_Router;
if ( dword_B87270 )
  v3 = runtime_gcWriteBarrierCX(&p_mux_Router->namedRoutes, v1, v2);
else
  p_mux_Router->namedRoutes = v17;
v16 = (void *)v3;
v10 = github_com_gorilla_mux__ptr_Router_PathPrefix(v3, (__int64)&qword_899AA6 + 4, 4LL);
v8 = github_com_gorilla_mux__ptr_Route_Subrouter(v10);
wssoft2_pkg_fs__ptr_WebServer_initApi((__int64)a1, v8);
v15 = (__int64 *)github_com_gorilla_mux__ptr_Router_PathPrefix((__int64)v16, (__int64)&qword_8997C0 + 6, 1LL);
p_fs_fileHandler = (fs_fileHandler *)runtime_newobject((__int64)&RTYPE_fs_fileHandler);
p_fs_fileHandler->root.len = 1LL;
p_fs_fileHandler->root.ptr = (char *)&qword_8997C0 + 6;
p_fs_fileHandler->fs.tab = go_itab__ptr_http_Dir_comma__ptr_http_FileSystem;
p_fs_fileHandler->fs.data = &off_914F90;
v12 = net_http_StripPrefix(
        (__int64)&qword_8997C0 + 6,
        1LL,
        (__int64)&go_itab__ptr_fs_fileHandler_comma__ptr_http_Handler,
        (__int64)p_fs_fileHandler);
if ( !v15[5] )
{
  *v15 = v12;
  if ( dword_B87270 )
    runtime_gcWriteBarrier(v15 + 1);
  else
    v15[1] = v13;
}
p_http_Server = (http_Server *)runtime_newobject((__int64)&RTYPE_http_Server);
v14 = p_http_Server;
p_http_Server->Handler.tab = go_itab__ptr_mux_Router_comma__ptr_http_Handler;
if ( dword_B87270 )
  runtime_gcWriteBarrierCX(&p_http_Server->Handler.data, v1, v5);
else
  p_http_Server->Handler.data = v16;
v11 = runtime_convI2I((__int64)&RTYPE_net_Listener, (__int64)a1->l.tab, (__int64)a1->l.data);
return (error)net_http__ptr_Server_Serve(v14, v11, *((__int64 *)&v11 + 1));
```

Figure 5. Web Service Handler Decompiler Output Setting Up the Web Server With Specific API Endpoints

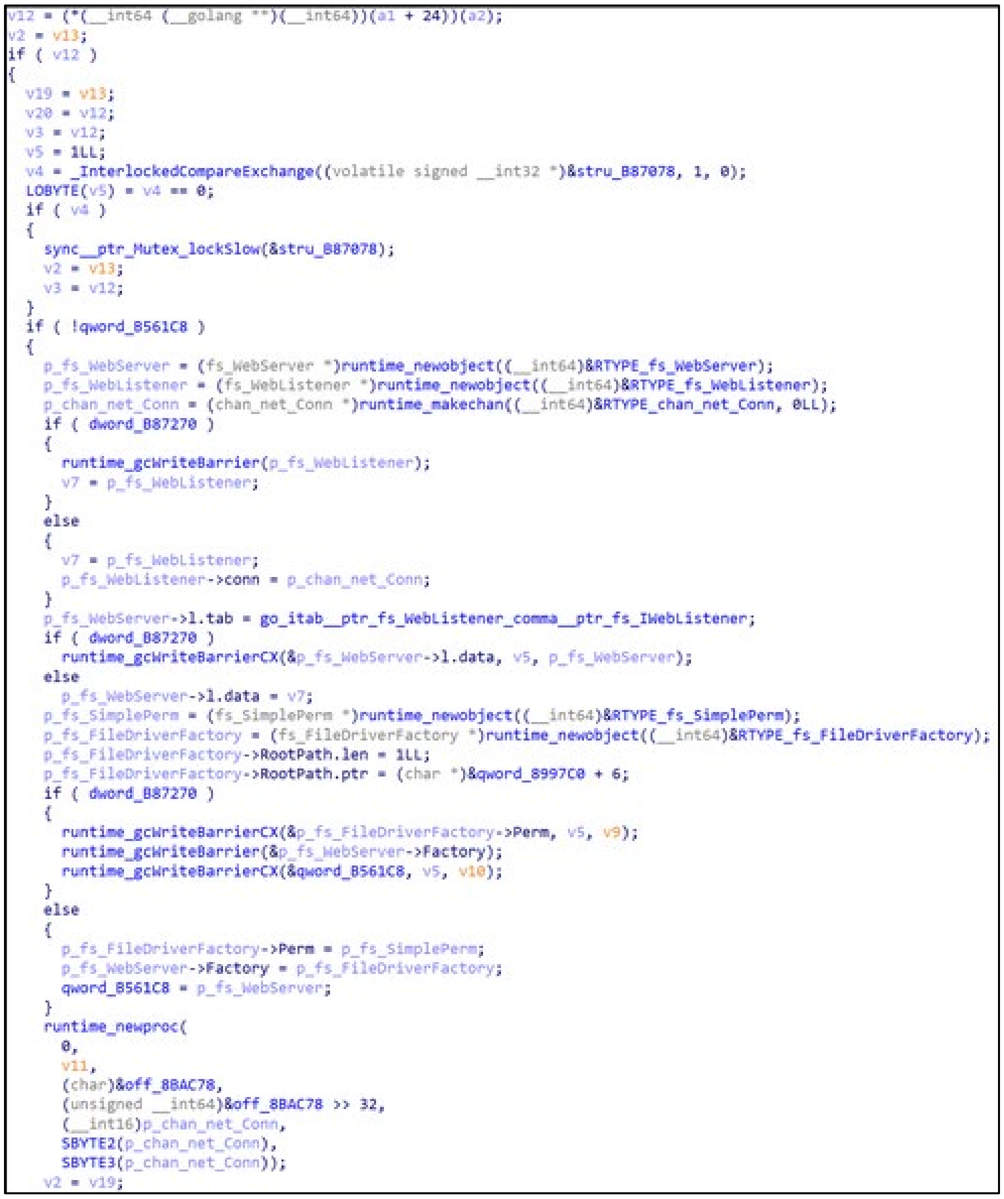
Through the API, the cyber actors can browse, upload, download, create, delete, and manipulate files and folders on the victim's system. See **Table 11** for file management commands contained in BRICKSTORM.

Table 11. BRICKSTORM File Management Commands

Command	Function
file-md5	Calculates the MD5 checksum of a specified file to verify file integrity.
get-file	Downloads a file from the compromised system to the C2 server [T1041].
list-dir	Lists the contents of a directory on the compromised system (e.g., browses the file system) [T1083].
put-file	Uploads a file from the C2 server to the compromised system.
slice-up	Reads and downloads specific, partial sections of a file.

To evade detection, BRICKSTORM serves seemingly legitimate web file types, such as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript, from a designated directory.

See **Figure 6** for the Web Service Handler decompiler output.

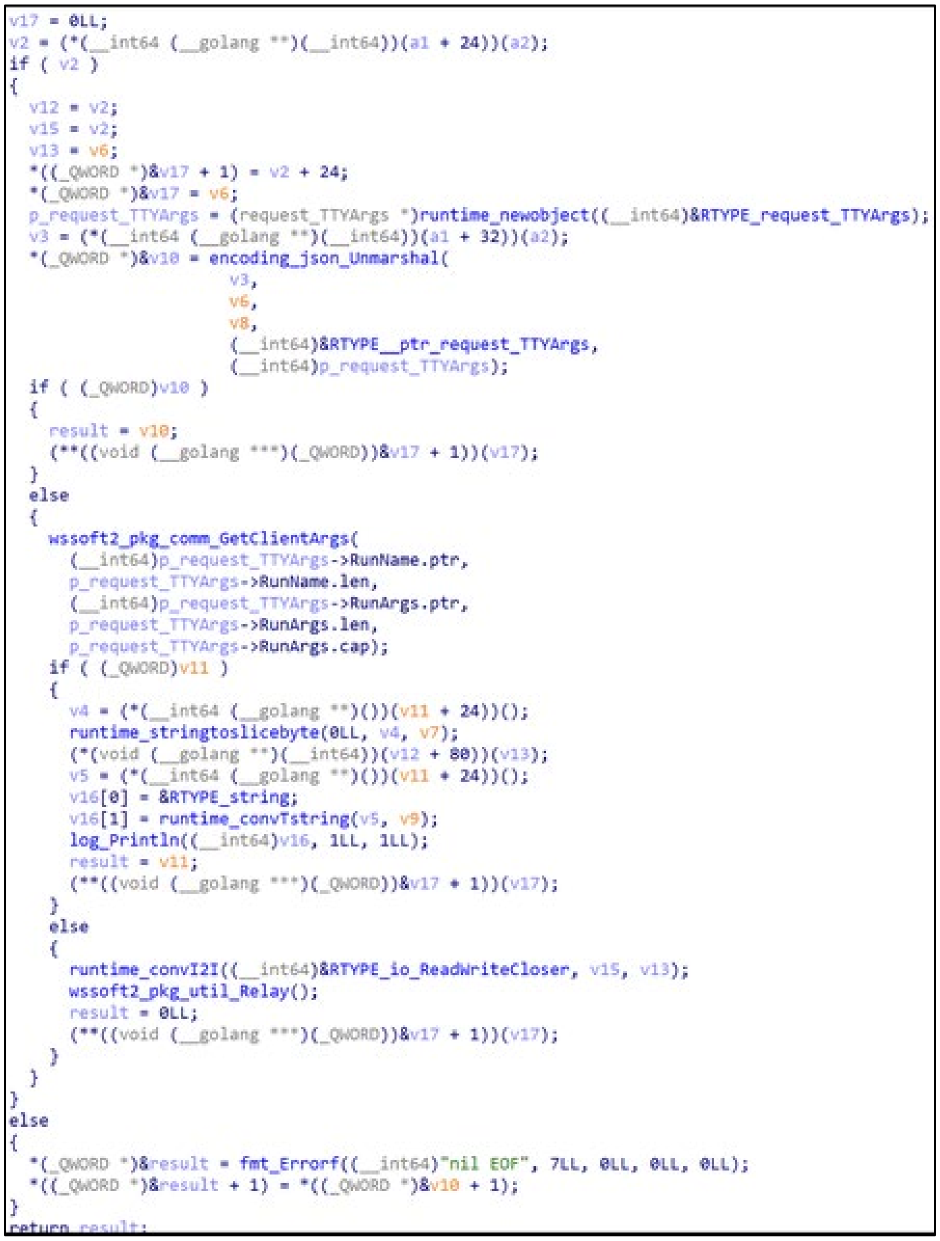


```
v12 = (*(__int64 (__golang **)(__int64))(a1 + 24))(a2);
v2 = v13;
if ( v12 )
{
  v19 = v13;
  v20 = v12;
  v3 = v12;
  v5 = 1LL;
  v4 = _InterlockedCompareExchange((volatile signed __int32 *)&stru_B87078, 1, 0);
  LOBYTE(v5) = v4 == 0;
  if ( v4 )
  {
    sync__ptr_Mutex_lockSlow(&stru_B87078);
    v2 = v13;
    v3 = v12;
  }
  if ( !qword_B561C8 )
  {
    p_fs_WebServer = (fs_WebServer *)runtime_newobject((__int64)&RTYPE_fs_WebServer);
    p_fs_WebListener = (fs_WebListener *)runtime_newobject((__int64)&RTYPE_fs_WebListener);
    p_chan_net_Conn = (chan_net_Conn *)runtime_makechan((__int64)&RTYPE_chan_net_Conn, 0LL);
    if ( dword_B87270 )
    {
      runtime_gcWriteBarrier(p_fs_WebListener);
      v7 = p_fs_WebListener;
    }
    else
    {
      v7 = p_fs_WebListener;
      p_fs_WebListener->conn = p_chan_net_Conn;
    }
    p_fs_WebServer->l.tab = go_itab__ptr_fs_WebListener_comma__ptr_fs_IWebListener;
    if ( dword_B87270 )
      runtime_gcWriteBarrierCX(&p_fs_WebServer->l.data, v5, p_fs_WebServer);
    else
      p_fs_WebServer->l.data = v7;
    p_fs_SimplePerm = (fs_SimplePerm *)runtime_newobject((__int64)&RTYPE_fs_SimplePerm);
    p_fs_FileDriverFactory = (fs_FileDriverFactory *)runtime_newobject((__int64)&RTYPE_fs_FileDriverFactory);
    p_fs_FileDriverFactory->RootPath.len = 1LL;
    p_fs_FileDriverFactory->RootPath.ptr = (char *)&qword_8997C0 + 6;
    if ( dword_B87270 )
    {
      runtime_gcWriteBarrierCX(&p_fs_FileDriverFactory->Perm, v5, v9);
      runtime_gcWriteBarrier(&p_fs_WebServer->Factory);
      runtime_gcWriteBarrierCX(&qword_B561C8, v5, v10);
    }
    else
    {
      p_fs_FileDriverFactory->Perm = p_fs_SimplePerm;
      p_fs_WebServer->Factory = p_fs_FileDriverFactory;
      qword_B561C8 = p_fs_WebServer;
    }
    runtime_newproc(
      0,
      v11,
      (char)&off_8BAC78,
      (unsigned __int64)&off_8BAC78 >> 32,
      (__int16)p_chan_net_Conn,
      SBYTE2(p_chan_net_Conn),
      SBYTE3(p_chan_net_Conn));
    v2 = v19;
```

Figure 6. Web Service Handler Decompiler Output

The Command Handler executes shell commands on the compromised system, giving the cyber threat actors full control over the compromised system through interactive command-line access. The handler receives a JSON request from the C2 server, parses it, and extracts it. The handler then sets up a `pseudo-terminal` (a virtual command-line interface) and runs the command on the victim system.

See **Figure 7** for the Command Handler decompiler output.



```
v17 = 0LL;
v2 = (*(__int64 (__golang **)(__int64))(a1 + 24))(a2);
if ( v2 )
{
  v12 = v2;
  v15 = v2;
  v13 = v6;
  *((_QWORD *)&v17 + 1) = v2 + 24;
  *(_QWORD *)&v17 = v6;
  p_request_TTYArgs = (request_TTYArgs *)runtime_newobject((__int64)&RTYPE_request_TTYArgs);
  v3 = (*(__int64 (__golang **)(__int64))(a1 + 32))(a2);
  *(_QWORD *)&v10 = encoding_json_Unmarshal(
                      v3,
                      v6,
                      v8,
                      (__int64)&RTYPE__ptr_request_TTYArgs,
                      (__int64)p_request_TTYArgs);
  if ( (_QWORD)v10 )
  {
    result = v10;
    (**((void (__golang ***)(_QWORD))&v17 + 1))(v17);
  }
  else
  {
    wssoft2_pkg_comm_GetClientArgs(
      (__int64)p_request_TTYArgs->RunName.ptr,
      p_request_TTYArgs->RunName.len,
      (__int64)p_request_TTYArgs->RunArgs.ptr,
      p_request_TTYArgs->RunArgs.len,
      p_request_TTYArgs->RunArgs.cap);
    if ( (_QWORD)v11 )
    {
      v4 = (*(__int64 (__golang **)())(v11 + 24))();
      runtime_stringtoslicebyte(0LL, v4, v7);
      (*(void (__golang **)(__int64))(v12 + 80))(v13);
      v5 = (*(__int64 (__golang **)())(v11 + 24))();
      v16[0] = &RTYPE_string;
      v16[1] = runtime_convTstring(v5, v9);
      log_Println((__int64)v16, 1LL, 1LL);
      result = v11;
      (**((void (__golang ***)(_QWORD))&v17 + 1))(v17);
    }
    else
    {
      runtime_convI2I((__int64)&RTYPE_io_ReadWriteCloser, v15, v13);
      wssoft2_pkg_util_Relay();
      result = 0LL;
      (**((void (__golang ***)(_QWORD))&v17 + 1))(v17);
    }
  }
}
else
{
  *(_QWORD *)&result = fmt_Errorf((__int64)"nil EOF", 7LL, 0LL, 0LL, 0LL);
  *((_QWORD *)&result + 1) = *((_QWORD *)&v10 + 1);
}
return result;
```

Figure 7. Command Handler Decompiler Output

Samples 2 Through 6

Initial Connection to the C2 Server: Like Sample 1, these samples create an encrypted DNS query for hard-coded C2 domains (redacted) and use DoH to resolve the addresses of their C2 servers by sending an encrypted HTTPS request to one of the following legitimate public DoH resolvers:

- https[:]//1.0.0[.]1/dns-query (Cloudflare)
- https[:]//1.1.1[.]1/dns-query (Cloudflare)
- https[:]//149.112.112[.]11/dns-query (Quad9)
- https[:]//45.90.28.160/dns-query (NextDNS)

- https[:]//8.8.4[.]4/dns-query (Google)
- https[:]//8.8.8[.]8/dns-query (Google)
- https[:]//9.9.9[.]11/dns-query (Quad9)
- https[:]//9.9.9[.]9/dns-query (Quad9)

Note: Some of these samples use XOR encryption to decrypt IPv4 addresses for DoH servers.

Establishing Secure Communications: Like Sample 1, these samples establish WebSocket Secure (WSS) connections with the C2 server and set up a multiplexing layer.

Full System Control: Once the connection is established with the C2 server via WebSockets, these BRICKSTORM samples receive commands that are directed to one of four specific handlers to perform tasks on the compromised system: SOCKS Handler, Web Service Handler, Command Handler, or CommandNoContext Handler. The SOCKS, Web Service, and Command Handlers function similar to the Sample 1 handlers. The CommandNoContext Handler executes shell commands on the compromised system without using an explicit security context.

Sample 7

Initial Connection to the C2 Server: Sample 7 retrieves configuration parameters from environment variables, performs checks, generates a TLS configuration used for secure communication to BRICKSTORM's client, and starts a network communications routine. This sample also uses a VSOCK interface to enable inter-VM communication, support data exfiltration, and maintain persistence in virtualized environments.

Upon execution, Sample 7 retrieves the following three configuration values from environment variables using the `os_Getenv` function:

- `listenAddr` (listen address and port)
- `listenPath` (listen path to route requests to the WSS connection)
- `password` (authentication key)

Establishing Secure Communications: Sample 7 establishes a secure WebSocket server with minimal external dependencies; specifically, all communication is encrypted using in-memory self-signed certificates. This enables encrypted communication without relying on publicly trusted Certificate Authorities (CAs) or storing certificate files on disk. It dynamically generates a self-signed X.509 certificate and a corresponding 2048-bit Rivest–Shamir–Adleman (RSA) private key in memory, which are loaded into a `tls.Certificate` struct and assigned to the certificate field's `tls.Config` object. This allows the server to handle HTTPS/WSS connections using the in-memory self-signed certificate, as standard NET/HTTP servers are configured to use `tls.Config`.

Sample 7 uses a single, multiplexed connection over secure WebSockets to communicate with a specified C2 address (retrieved from the `listenAddr` value) and path (retrieved from the `listenPath` value). During or before the WSS handshake, Sample 7 implements a custom authentication check, involving the specific pre-shared authentication key (retrieved from `password` value).

Full System Control: Once the WSS connection with the C2 server is established, Sample 7 processes incoming commands through one of four handlers: Web Service Handler, Command Handler, VSOCK-proxy handler, or VSOCK handler.

The Web Service Handler functions similar to Sample 1's Web Service Handler.

The Command Handler functions similar to Sample 1's Command Handler.

The VSOCK-proxy Handler performs VSOCK relaying and network tunneling. It implements a proxy with specific configuration arguments to establish a tunneled connection to process JSON payloads. First, the handler unmarshals the payload data and extracts and validates the `TunnelAddr`, `Context ID (CID)`, `Port`, and `Family` configuration arguments. Based on the validated arguments, the handler binds to a specific VSOCK address (defined by the CID and port) and establishes a connection to the destination specified by `TunnelAddr`. When the connection is completed or terminated, the handler sends an appropriate success or error response back to the client. This functionality enables cyber actors to maintain covert communication channels, evade detection, and pivot within virtualized environments.

The VSOCK Connection Handler creates and connects to VSOCK endpoints to maintain covert connections within the virtual environment. It processes incoming network requests containing a JSON payload with specific configuration arguments for connecting to a VSOCK endpoint. The handler extracts the JSON payload from the request and uses a JSON parser to unmarshal the data into a structured object with fields for `Context (CID)`, `Port`, and `Family`. The handler checks the unmarshalled data for validity and, if the configuration is valid, the handler establishes a connection to a VSOCK endpoint using a specified CID and port number. If the virtual socket creation is successful, the handler allocates a new runtime object to hold the CID and port information. If unmarshalling fails, validation fails, or the destination connection cannot be established, the handler returns an appropriate error to the client.

Sample 8

Like Sample 7, Sample 8:

- Retrieves C2 parameters (`listenAddr`, `listenPath`, and `password`) from environment variables,
- Uses a self-signed X.509 certificate and a corresponding 2048-bit RSA private key in memory to facilitate encrypted communications without relying on a CA,
- Establishes a secure WebSocket server for encrypted communication, and
- Directs commands to specific handlers.

Sample 8's handlers directing commands differ from Sample 7. In addition to a Web Service Handler, Command Handler, VSOCK-proxy Handler, and VSOCK Connection Handler, Sample 8 also has two additional handlers: The SOCKS Handler (which functions similar to Sample 1's SOCKS Handler) and the CommandNoContext Handler (which functions similar to Samples 2 through 6's CommandNoContext Handler).

Detection

YARA Rules

Deploy the CISA-created YARA rules in **Table 12** to detect malicious activity. See **Appendix B: Scanning Guidance on Remote Hosts** for guidance on how to identify activity with these rules.

Table 12. YARA Rules

BRICKSTORM Rule 1

```
rule CISA_251165_02 : BRICKSTORM backdoor installs_other_components communicates_with_c2
exfiltrates_data

{

meta:

        author = "CISA Code & Media Analysis"

        incident = "251165"

        date = "2025-09-29"

        last_modified = "202051001_1008"

        actor = "n/a"

        family = "BRICKSTORM"

        capabilities = "installs-other-components communicates-with-c2 exfiltrates-data"

        malware_type = "backdoor"

        tool_type = "unknown"

        description = "Detects Go-Based BRICKSTORM backdoor samples"

        sha256_1 = "aaf5569c8e349c15028bc3fac09eb982efb06eabac955b705a6d447263658e38"

strings:

        $s0 = { 6D 61 69 6E 2E 73 74 61 72 74 4E 65 77 }

        $s1 = { 6D 61 69 6E 2E 73 65 6C 66 57 61 74 63 68 65 72 }

        $s2 = { 6D 61 69 6E 2E 73 65 74 53 65 72 76 69 63 65 43 66 67 }

        $s3 = { 73 6F 63 6B 73 2E 48 61 6E 64 6C 65 53 6F 63 6B 73 52 65 71 75 65 73 74 }

        $s4 = { 77 65 62 2E 57 65 62 53 65 72 76 69 63 65 }

        $s5 = { 63 6F 6D 6D 61 6E 64 2E 48 61 6E 64 6C 65 54 54 59 52 65 71 75 65 73 74 }

        $s6 = { 77 65 62 73 6F 63 6B 65 74 2E 28 2A 57 53 43 6F 6E 6E 65 63 74 6F 72 29 2E 43 6F 6E
        6E 65 63 74 }
        $s7 = { 66 73 2E 28 2A 57 65 62 53 65 72 76 65 72 29 2E 52 75 6E 53 65 72 76 65 72 }
        $s8 = { 68 74 74 70 73 3A 2F 2F 31 2E 30 2E 30 2E 31 2F 64 6E 73 2D 71 75 65 72 79 }
        $s9 = { 68 74 74 70 73 3A 2F 2F 31 2E 31 2E 31 2E 31 2F 64 6E 73 2D 71 75 65 72 79 }
        $s10 = { 68 74 74 70 73 3A 2F 2F 38 2E 38 2E 34 2E 34 2F 64 6E 73 2D 71 75 65 72 79 }
        $s11 = { 68 74 74 70 73 3A 2F 2F 38 2E 38 2E 38 2E 38 2F 64 6E 73 2D 71 75 65 72 79 }
        $s12 = { 68 74 74 70 73 3A 2F 2F 39 2E 39 2E 39 2E 39 2F 64 6E 73 2D 71 75 65 72 79 }
condition:
8 of them
}
```

BRICKSTORM Rule 2

```
rule CISA_251155_02 : BRICKSTORM backdoor installs_other_components communicates_with_c2
exfiltrates_data
{
meta:
        author = "CISA Code & Media Analysis"
        incident = "251155"
        date = "2025-09-15"
        last_modified = "20250916_1511"
        actor = "n/a"
        family = "BRICKSTORM"
        capabilities = "installs-other-components communicates-with-c2 exfiltrates-data"
        malware_type = "backdoor"
        tool_type = "unknown"
        description = "Detects Go-Based BRICKSTORM backdoor samples"
        sha256_1 = "320a0b5d4900697e125cebb5ff03dee7368f8f087db1c1570b0b62f5a986d759"
        sha256_1 = "dfac2542a0ee65c474b91d3b352540a24f4e223f1b808b741cfe680263f0ee44"
        sha256_1 = "b91881cb1aa861138f2063ec130b2b01a8aaf0e3f04921e5cbfc61b09024bf12"
        sha256_1 = "bfb3ffd46b21b2281374cd60bc756fe2dcc32486dcc156c9bd98f24101145454"
strings:
        $s0 = { 04 30 0F B6 54 04 2C 31 D1 88 4C 04 34 48 FF C0 }
        $s1 = { 48 83 F8 04 7C E7 48 C7 04 24 }
        $s2 = { 48 8D 44 24 34 48 89 44 24 08 48 C7 44 24 10 04 }
        $s3 = { 48 89 44 24 48 48 89 4C 24 50 48 8B 6C 24 38 48 }
        $s4 = { 48 83 EC 40 48 89 6C 24 38 48 8D 6C 24 38 C7 44 24 }
        $s5 = { 83 EC 38 48 89 6C 24 30 48 8D 6C 24 30 C6 44 24 }
        $s6 = { 4C 24 20 48 89 44 24 40 48 89 4C 24 48 48 8B 6C }
        $s7 = { 64 48 8B 0C 25 F8 FF FF FF 48 3B 61 10 0F 86 81 }
        $s8 = { 64 48 8B 0C 25 F8 FF FF FF 48 3B 61 10 0F 86 91 }
condition:
        all of them
}
```

Sigma Rule

Deploy the CISA-created Sigma rule in Table 13 to detect BRICKSTORM.

Note: This rule can be run in an entity's security information and event management (SIEM) system, but it will only be useful if the SIEM contains the vCenter logs. Additionally, this detection method will not work if run on endpoint detection and response (EDR) logs.

Table 13. Sigma Rule

BRICKSTORM
CISA Code & Media Analysis ## ############ README ############### ## Edit rules and queries as needed for your hunt and based on your environment. ## Ensure your EDR/SIEM instance has enough memory to run these AND/OR condition based queries. May take longer to run than conventional Sigma rule query. ## Do not edit "logsource-product:" unless you are editing this rule to meet specific logsources/fields and know your environment. ## TLP GREEN + Please use local installation of Sigma to convert this rule. ## TLP CLEAR may convert rules using online converter of choice. #################################### title: BRICKSTORM Backdoor Activity r2 incident: 251157.r2 tlp: CLEAR id: 329bec83-54bd-405f-a5ab-ba97ec5e6057

BRICKSTORM

```
status: test
description: BRICKSTORM malware is a backdoor with multiple capabilities that threat actors use to set up persistence on exploited systems.
references:
    - https://cloud.google.com/blog/topics/threat-intelligence/brickstorm-espionage-campaign
    - https://cloud.google.com/blog/topics/threat-intelligence/ivanti-post-exploitation-lateral-movement
    - https://ctid.mitre.org/blog/2024/05/22/infiltrating-defenses-abusing-vmware-in-mitres-cyber-intrusion/
    - https://cybersecuritynews.com/new-brickstorm-stealthy-backdoor/
author: CISA Code & Media Analysis
date: 2025-09-29
modified: 2025-09-29
tags:
    - attack.brickstorm
    - attack.unc5221
logsource:
    product: cma
detection:
    keywords_1:
        - 'vCenter'
    keywords_2:
        - 'inventory object'
        - 'object'
    keywords_3:
        - 'clone'
        - 'destroy'

    keywords_4:
        - 'GET'
        - 'POST'
        - 'PUT'
```

BRICKSTORM

```
keywords_5:
  - 'HTTP/1.1'
keywords_6:
  - '200'
keywords_7:
  - '/rest/com/vmware/cis/session'
  - '/rest/appliance/access/ssh'
keywords_8:
  - 'User Agent'

keywords_9:
  - 'sed -i'
keywords_10:
  - 'export'
  - 'echo'
keywords_11:
  - 'vami-lighttp'
  - '/etc/sysconfig/init'

keywords_12:
  - 'Administrator'
keywords_13:
  - 'Creating local person user'
  - 'Adding users'
  - 'Updating local group'
  - 'Removing principals'
  - 'Deleting principal'
keywords_14:
  - 'PrincipalManagement'
```

BRICKSTORM

```
keywords_15:
  - 'sshd'
keywords_16:
  - 'Postponed keyboard-interactive/pam'


keywords_17:
  - '/bin/vmx'
keywords_18:
  - '-x'
keywords_19:
  - '/vmfs/volumes.vmx'
keywords_20:
  - '2>/dev/null'
keywords_21:
  - '0>/dev/null'


keywords_22:
  - '$parts ='
keywords_23:
  - 'Get-Item -Path'
keywords_24:
  - '"C:\Windows\System32\drivers\etc\hosts":frag*'
keywords_25:
  - '$loader ='
keywords_26:
  - '[IO.File]::ReadAllText'
keywords_27:
  - 'Invoke-Expression $loader'


keywords_28:
```

BRICKSTORM

```
    - 'cp'
    - 'delete'
keywords_29:
    - 'home/vsphere-ui/vcli'
    - '/opt/vmware/sbin'
keywords_30:
    - 'vami-httpd'

keywords_31:
    - 'testComputer$'
keywords_32:
    - 'ldap-ivanti'

keywords_33:
    - 'https://9.9.9.9/dns-query'
    - 'https://45.90.28.160/dns-query'
    - 'https://45.90.30.160/dns-query'
    - 'https://149.112.112.112/dns-query'
    - 'https://9.9.9.11/dns-query'
    - 'https://1.1.1.1/dns-query'
    - 'https://1.0.0.1/dns-query'
    - 'https://8.8.8.8/dns-query'
    - 'https://8.8.4.4/dns-query'
    - '/home/bin/netmon'
    - '/home/bin/logd'
    - '/home/runtime/logd'
    - '/home/config/logd.spec.cfg'
    - '/api/file/change-dir'
    - '/api/file/delete-dir'
    - '/api/file/delete-file'
```

BRICKSTORM
- '/api/file/mkdir' - '/api/file/list-dir' - '/api/file/rename' - '/api/file/put-file' - '/api/file/get-file' - '/api/file/slice-up' - '/api/file/file-md5' - '/api/file/up' - '/api/file/stat' condition: keywords_1 and keywords_2 and keywords_3 or keywords_4 and keywords_5 and keywords_6 and keywords_7 and keywords_8 or keywords_9 and keywords_10 and keywords_11 or keywords_12 and keywords_13 and keywords_14 or keywords_15 and keywords_16 or keywords_17 and keywords_18 and keywords_19 and keywords_20 and keywords_21 or keywords_22 and keywords_23 and keywords_24 and keywords_25 and keywords_26 and keywords_27 or keywords_28 and keywords_29 and keywords_30 or keywords_31 and keywords_32 or keywords_33 falsepositives: - Rate of FP low-moderate with some strings. - Use this rule in an infected environment/logs. - Analyst may need to make adjustments to the query as required. level: high

Additional Detection Resources

See the following resources for detecting BRICKSTORM.

Google Mandiant's tactics, techniques, and procedures (TTPs)-based hunt guidance and YARA detections rules provided in Another BRICKSTORM: Stealthy Backdoor Enabling Espionage into Tech and Legal Sectors.

Google Mandiant's BRICKSTORM Espionage Campaign YARA Rules, available at Github.

Google Mandiant's BRICKSTORM Scanner: BRICKSTORM Indicator of Compromise Scanner.

Use the script by first mounting an image followed by the scan.

To mount the image:

- sudo mkdir -p /mnt/image
- sudo mount -o ro,loop image.001 /mnt/image

To unmount the image:

- sudo umount /mnt/image

The script can also be used by mounting a remote server to your local VM to scan its file system:

- sudo apt update
- sudo apt install -y sshfs
- sudo mkdir -p /mnt/remote-server
- sudo chown $(whoami):$(whoami) /mnt/remote-server
- sudo sed -i 's/^# *user_allow_other/user_allow_other/' /etc/fuse.conf || echo 'user_allow_other' | sudo tee -a /etc/fuse.conf
- sudo sshfs root@IPAddress:/ /mnt/remote-server
- sudo ls -la /mnt/remote-server
- sudo yara yara.rule -r /mnt/remote-server
- sudo umount -l /mnt/remote-server
- ls -la /mnt/remote-server

NVISO's analysis of Windows-based variants with IOCs and detection rules contains YARA and other detection and hunting rules. See NVISO Incident Response BRICKSTORM Backdoor Analysis.

CrowdStrike's VirtualGHOST PowerShell Script: CrowdStrike / VirtualGHOST

This script can be used to identify unregistered VMware VMs.

To run in the script PowerShell or pwsh, complete the following steps:

1. Set-ExecutionPolicy RemoteSigned
2. Install-Module -Name VMware.PowerCLI -Scope CurrentUser
3. Import-Module VMware.PowerCLI
4. Get-Module -ListAvailable VMware.PowerCLI

To run the script in Windows, use `.\Detect-VirtualGHOST.ps1`.

To run in the script in Linux, use `sudo apt install -y powershell`.

For vCenter servers, use `username@domain.local` instead of `root`. For ESXi Servers, you may use `root` username.

Incident Response

U.S. organizations: If BRICKSTORM, similar malware, or potentially related activity is detected, CISA and NSA urge organizations to report the activity as required by law and applicable policies. To enable CISA to provide tailored incident response assistance and build a comprehensive picture of this activity, CISA and NSA urge organizations to:

1. Immediately report the findings via CISA's 24/7 Operations Center (contact@cisa.dhs.gov), 1-844-Say-CISA (1-844-729-2472), or CISA's Incident Reporting System. Please identify the activity is related to BRICKSTORM, and CISA will reach out with next steps.
2. Use CISA's Malware Analysis Submission Form to submit a file containing the malicious code. Include the CISA-provided Incident ID number (obtained from reporting the compromise) in the Open Incident ID field.

Canadian organizations: Report incidents by emailing Cyber Centre at contact@cyber.gc.ca or online via the reporting tool Report a Cyber Incident - Canadian Centre for Cyber Security.

Mitigations

CISA, NSA, and Cyber Centre recommend organizations implement the mitigations below to improve organization cybersecurity posture based on the cyber actors' activity. These mitigations align with the Cross-Sector Cybersecurity Performance Goals (CPGs) developed by CISA and the National Institute of Standards and Technology (NIST). The CPGs provide a minimum set of practices and protections that CISA and NIST recommend all organizations implement. CISA and NIST based the CPGs on existing cybersecurity frameworks and guidance to protect against the most common and impactful threats, tactics, techniques, and procedures. Visit CISA's CPGs webpage for more information on the CPGs, including additional recommended baseline protections.

- **Upgrade VMware vSphere servers** to the latest version [CPG 1.E].
- **Harden your VMware vSphere environments** by applying VMware's guidance available at Github. For additional guidance on logging [CPG 2.T] and hardening, see From Help Desk to Hypervisor: Defending Your VMware vSphere Estate from UNC3944.
- **Take inventory of all network edge devices [CPG 1.A] and monitor** for any suspicious network connectivity originating from these devices.
- **Ensure proper network segmentation restricts network traffic from the DMZ to the internal network [CPG 2.F].**
- **Disable RDP and SMB** from the DMZ to the internal network.
- **Apply the principle of least privilege and restrict service accounts** to only needed permissions.
- **Increase monitoring for service accounts,** which are highly privileged and have a predictable pattern of behavior (e.g., scans that reliably run at a certain hour of the day).
- **Block unauthorized DoH providers and external DoH network traffic** to reduce unmonitored communications.

Disclaimer

CISA, NSA, and Cyber Centre do not endorse any commercial entity, product, company, or service, including any entities, products, companies, or services linked within this document. Any reference to specific commercial entities, products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply endorsement, recommendation, or favoring by CISA, NSA, or Cyber Centre.

Acknowledgements

VMware contributed to this advisory.

Version History

December 4, 2025: Initial version.

Appendix A: MITRE ATT&CK Techniques

See **Table 14** through **Table 20** for all referenced threat actor tactics and techniques in this advisory. For assistance with mapping malicious cyber activity to the MITRE ATT&CK framework, see CISA and MITRE ATT&CK's Best Practices for MITRE ATT&CK Mapping and CISA's Decider Tool.

Table 14. Persistence

Technique Title	ID	Use
Boot or Logon Initialization Scripts	T1037	The cyber actors modify the `init` file to execute BRICKSTORM.
Hijack Execution Flow: Path Interception by PATH Environment Variable	T1574.007	BRICKSTORM modifies the `PATH` environment variable so that the copied version of the BRICKSTORM will execute if commands or process reference it.
Server Software Component: Web Shell	T1505.003	The cyber actors accessed a web server inside a victim organization's DMZ using a web shell.

Table 15. Privilege Escalation

Technique Title	ID	Use
Abuse Elevation Control Mechanism: Sudo and Sudo Caching	T1548.003	The cyber actors elevated privileges using the `sudo` command.

Table 16. Defense Evasion

Technique Title	ID	Use
Masquerading	T1036	Some BRICKSTORM samples mimic legitimate names. For example, Sample 1, which was obtained from a VMware vSphere platform, is named `vmsrc` or `vmware-sphere`.
Valid Accounts	T1078	The cyber actors moved laterally using RDP with valid service account credentials.

Table 17. Discovery

Technique Title	ID	Use
File and Directory Discovery	T1083	BRICKSTORM can list directory contents on the compromised system.

Table 18. Credential Access

Technique Title	ID	Use
OS Credential Dumping: NTDS	T1003.003	The cyber actors copied `ntds.dit`.

Table 19. Command and Control

Technique Title	ID	Use
Application Layer Protocol: Web Protocols	T1071.001	BRICKSTORM uses DoH to resolve the address of its C2 servers by sending an encrypted HTTPS request.
Ingress Tool Transfer	T1105	The cyber actors dropped BRICKSTORM malware in the VMware vSphere server's `/etc/sysconfig/` directory. BRICKSTORM can download files from the cyber actors' C2 server to the compromised system.
Proxy: Internal Proxy	T1090.001	BRICKSTORM sets up a SOCKS proxy that routes C2 traffic and allows cyber actors to move laterally throughout the victim network.

Table 20. Exfiltration

Technique Title	ID	Use
Exfiltration Over C2 Channel	T1041	BRICKSTORM can upload files from the victim system to the cyber actors' C2 server.

Appendix B: Scanning Guidance on Remote Hosts

The following tools are designed to support the identification of potentially malicious artifacts and activities but should not be used as standalone detection mechanisms.

Remote YARA Scan Using Nessus

1. Log into Nessus and go to "My Account."
2. Press "About" tab on the left side.
3. Go to Software Update tab and manually update all components.
4. After the update is done, select "Scans" at the top and press the "New Scan" button.
5. Select "Advanced Scan."
6. Give a name and description to your scan.
7. In the "Targets" section, input the IP address of the remote server you want to scan.
8. On the left pane under Settings select "Assessment" then "Malware."
9. Toggle "Scan for malware" on.
10. Scroll down to "Yara Rules" and add your Yara rules file.
11. Select the filesystem and drives to scan.
12. Go back to the top and press "Credentials" tab.
13. Select SSH or Windows and input the credential of the server.
14. Nessus needs credentials to be able to do a Yara scan on the filesystem of the remote server.
15. In the "Plugins" tab, make sure to "Enable All."
16. Launch the scan.

Remote YARA Scan without Nessus

1. Mount Remote Sever to Kali to Scan the Filesystem
    a. sudo apt update
    b. sudo apt install -y sshfs
    c. sudo mkdir -p /mnt/remote-server
    d. sudo chown $(whoami):$(whoami) /mnt/remote-server
    e. sudo sed -i 's/^# *user_allow_other/user_allow_other/' /etc/fuse.conf || echo 'user_allow_other' | sudo tee -a /etc/fuse.conf
    f. sudo sshfs root@IPAddress:/ /mnt/remote-server
    g. sudo ls -la /mnt/remote-server
    h. sudo yara yara.rule -r /mnt/remote-server
    i. sudo umount -l /mnt/remote-server
    j. ls -la /mnt/remote-server

For more information see Tenable's Threat Hunting with YARA and Nessus.

Notes

[1] Matt Lin et al., “Cutting Edge, Part 4: Ivanti Connect Secure VPN Post-Exploitation Lateral Movement Case Studies,” Google Cloud Blog, April 4, 2024, https://cloud.google.com/blog/topics/threat-intelligence/ivanti-post-exploitation-lateral-movement.

[2] Maxime, “NVISO analyzes BRICKSTORM espionage backdoor,” NVISO, April 15, 2025, https://www.nviso.eu/blog/nviso-analyzes-brickstorm-espionage-backdoor.

[3] Sarah Yoder et al., “Another BRICKSTORM: Stealthy Backdoor Enabling Espionage into Tech and Legal Sectors,” Google Cloud Blog, September 24, 2025, https://cloud.google.com/blog/topics/threat-intelligence/brickstorm-espionage-campaign.