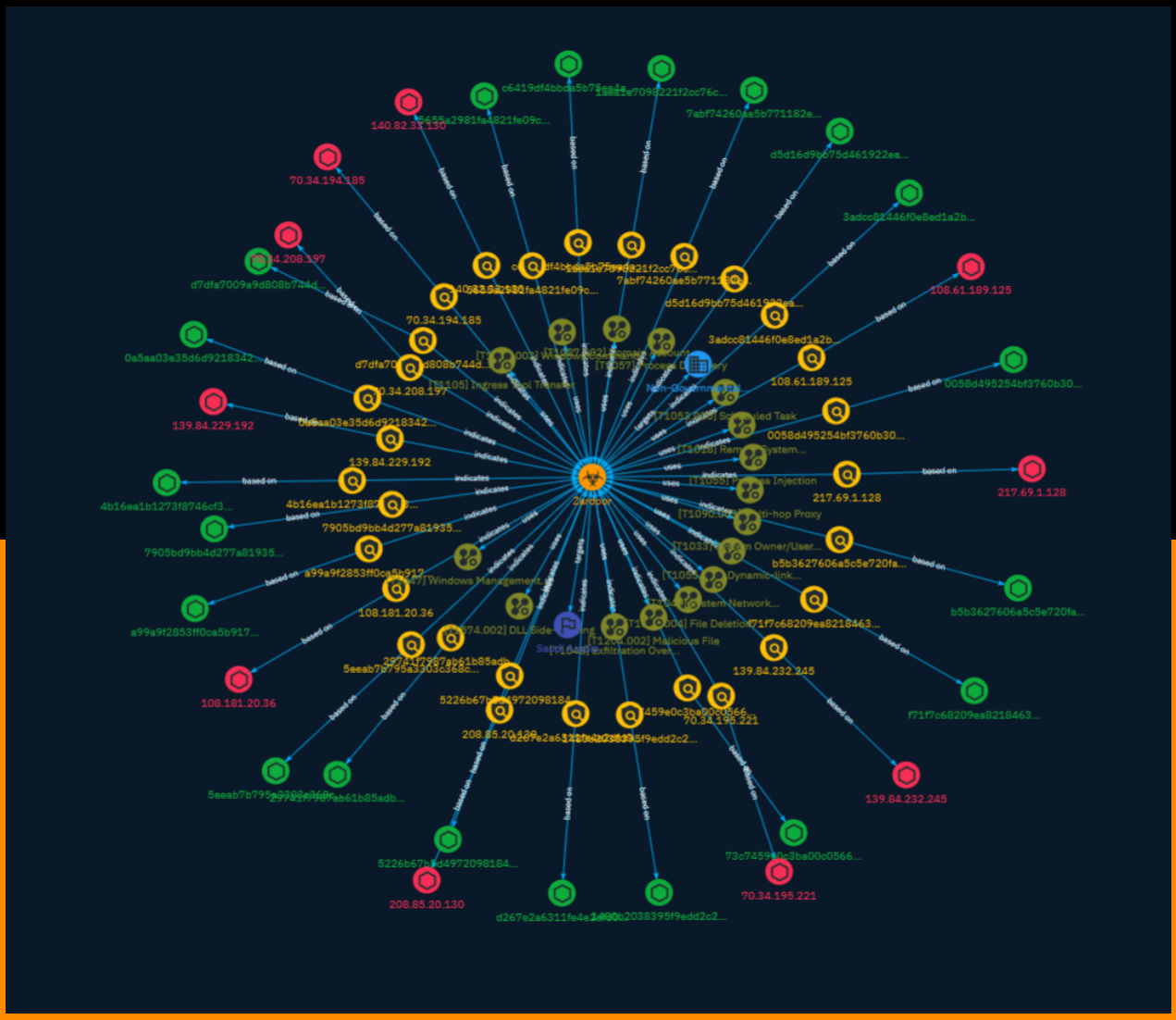
Feb 08 2024

NETMANAGEIT

# Intelligence Report

# New backdoor used in long-term cyber espionage operation targeting an Islamic organization



# Table of contents

## Overview


- Description 4
- Confidence 4
- Content 5


## Entities


- Indicator 6
- Malware 31
- Attack-Pattern 32
- Country 54
- Sector 55


## Observables


- IPv4-Addr 56
- StixFile 58

# External References


- External References 61

# Overview

## Description

Cisco Talos discovered an ongoing espionage campaign targeting an Islamic charitable organization in Saudi Arabia using a new backdoor malware family named Zardoor. The threat actor has likely been active since at least March 2021 and uses customized reverse proxy tools like Fast Reverse Proxy, sSocks, and Venom to establish command and control. The attacker spreads tools like Zardoor through Windows Management Instrumentation and maintains persistence with scheduled tasks. Talos assesses this is an advanced threat actor based on their ability to create new malware, customize open source tools, and use living-off-the-land techniques to remain undetected.

## Confidence

*This value represents the confidence in the correctness of the data contained within this report.*

15 / 100

# Content

N/A

# Indicator

| Name |
| --- |
| 70.34.195.221 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '70.34.195.221'] |
| **Name** |
| 70.34.194.185 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '70.34.194.185'] |
| **Name** |
| 217.69.1.128 |

| Pattern Type |
| --- |
| stix |
| **Pattern** |
| [ipv4-addr:value = '217.69.1.128'] |
| **Name** |
| 139.84.232.245 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '139.84.232.245'] |
| **Name** |
| 139.84.229.192 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '139.84.229.192'] |
| **Name** |
| 108.61.189.125 |

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '108.61.189.125']

**Name**

f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2']

**Name**

d7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'd7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87']

**Name**

d5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'd5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71']

**Name**

d267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'd267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef']

**Name**

c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74']

**Name**

b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897']

**Name**

a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d']

**Name**

7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3']

**Name**

73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095']

**Name**

208.85.20.130

**Description**

**ISP:** The Constant Company, LLC **OS:** - ---------- Services: **500:** ``` VPN (IKE) Initiator SPI: 33626f7234356879 Responder SPI: 6d7836666d74326a Next Payload: RESERVED Version: 2.0 Exchange Type: DOI Specific Use Flags: Encryption: False Commit: False Authentication: False Message ID: 00000000 Length: 36 ``` ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '208.85.20.130']

**Name**

5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852']

**Name**

5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e']

**Name**

5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480']

**Name**

4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3']

**Name**

3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1']

**Name**

29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd']

**Name**

140.82.33.130

**Description**

**ISP:** The Constant Company, LLC **OS:** Linux ---------- Services: **22:** ``` SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u4 Key type: ssh-rsa Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQDdPTzUicoJ111LZeexxmQwphU9WsHE90papaCe/fKFZBO2 whsqRc38/a2f+jB/sRvUQIYf6u+umxVYpdqrYQUNdNo87UOUGTUNE5ZI2EGpkzMAgv1Wtzrk8uI0Eos1tsN9JWYw1KG+5Sq7eF/GSUOjOYbrXRTHx9+oYU1wGpCPDyCzCCaPjj69x3cnyokwkvL/2SCBwweZ2XpWtEin5mfdOm7YXsPuWREcEuhvvfWLJKv4wCIk0lDWopeQxoQIitT1RXPVO6A9DrDfS3kZmRar+gIO7DNP65zfREP5zvYEEnrvF0pp+vKPD+C6EZT6Ozb7HY39AbO+jLksGmas69Ef Fingerprint: cb:e4:57:4d:3d:26:54:a3:7c:f1:a9:26:08:d3:8c:79 Kex Algorithms: curve25519-sha256 curve25519-sha256@libssh.org ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 diffie-hellman-group-exchange-sha256 diffie-hellman-group16-sha512 diffie-hellman-group18-sha512 diffie-hellman-group14-sha256 diffie-hellman-group14-sha1 kex-strict-s-v00@openssh.com Server Host Key Algorithms: rsa-sha2-512 rsa-sha2-256 ssh-rsa Encryption Algorithms: chacha20-poly1305@openssh.com aes128-ctr aes192-ctr aes256-ctr aes128-gcm@openssh.com aes256-gcm@openssh.com MAC Algorithms: umac-64-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512-etm@openssh.com hmac-sha1-etm@openssh.com umac-64@openssh.com umac-128@openssh.com hmac-sha2-256 hmac-sha2-512 hmac-sha1 Compression Algorithms: none zlib@openssh.com ``` ---------- **80:** ``` HTTP/1.1 403 Forbidden Server: nginx Date: Sun, 21 Jan 2024 17:56:38 GMT Content-Type: text/html Content-Length: 342 Connection: keep-alive Vary: Accept-Encoding ETag: "6530f430-156" ``` ---------- **443:** ``` HTTP/1.1 403 Forbidden Server: nginx Date: Fri, 26 Jan 2024 12:51:47 GMT Content-

Type: text/html Content-Length: 342 Connection: keep-alive Vary: Accept-Encoding ETag: "6530f430-156" ``` HEARTBLEED: 2024/01/26 12:51:59 140.82.33.130:443 - SAFE ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '140.82.33.130']

**Name**

1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778']

**Name**

1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071']

**Name**

0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d']

**Name**

0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf']

**Name**

70.34.208.197

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '70.34.208.197']

**Name**

108.181.20.36

**Description**

**ISP:** Psychz Networks **OS:** Ubuntu ---------- Services: **80:** ``` HTTP/1.1 404 Not Found Server: nginx/1.18.0 (Ubuntu) Date: Sat, 03 Feb 2024 20:07:48 GMT Content-Type: text/html Content-Length: 564 Connection: keep-alive ``` ---------- **443:** ``` HTTP/1.1 200 OK Server: nginx/1.18.0 (Ubuntu) Date: Tue, 06 Feb 2024 12:11:12 GMT Content-Type: text/html; charset=UTF-8 Transfer-Encoding: chunked Connection: keep-alive X-Content-Type-Options: nosniff Access-Control-Allow-Origin: * Access-Control-Allow-Methods: GET, HEAD, POST ``` HEARTBLEED: 2024/02/06 12:11:31 108.181.20.36:443 - SAFE ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '108.181.20.36']

**Name**

7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea

**Description**

GoLandBuildPE SHA256 of e0f4afe374d75608d604fbf108eac64f

**Pattern Type**

stix

| Pattern |
| --- |
| [file:hashes.'SHA-256' = '7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea'] |
| **Name** |
| 70.34.195.221 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '70.34.195.221'] |
| **Name** |
| 70.34.194.185 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [ipv4-addr:value = '70.34.194.185'] |
| **Name** |
| 217.69.1.128 |
| **Pattern Type** |

stix

**Pattern**

[ipv4-addr:value = '217.69.1.128']

**Name**

139.84.232.245

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '139.84.232.245']

**Name**

139.84.229.192

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '139.84.229.192']

**Name**

108.61.189.125

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '108.61.189.125']

**Name**

f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2']

**Name**

d7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'd7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87']

**Name**

d5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71

| Pattern Type |
| --- |
| stix |
| **Pattern** |
| [file:hashes.'SHA-256' = 'd5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71'] |
| **Name** |
| d267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef |
| **Pattern Type** |
| stix |
| **Pattern** |
| [file:hashes.'SHA-256' = 'd267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef'] |
| **Name** |
| c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [file:hashes.'SHA-256' = 'c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74'] |

**Name**

b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897']

**Name**

a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = 'a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d']

**Name**

7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3']

**Name**

73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095']

**Name**

208.85.20.130

**Description**

**ISP:** The Constant Company, LLC **OS:** - ---------- Services: **500:** ``` VPN (IKE) Initiator SPI: 33626f7234356879 Responder SPI: 6d7836666d74326a Next Payload: RESERVED Version: 2.0 Exchange Type: DOI Specific Use Flags: Encryption: False Commit: False Authentication: False Message ID: 00000000 Length: 36 ``` ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '208.85.20.130']

| Name |
| --- |
| 5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852 |
| **Pattern Type** |
| stix |
| **Pattern** |
| [file:hashes.'SHA-256' = '5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852'] |
| **Name** |
| 5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e |
| **Pattern Type** |
| stix |
| **Pattern** |
| [file:hashes.'SHA-256' = '5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e'] |
| **Name** |
| 5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480 |
| **Pattern Type** |
| stix |
| **Pattern** |

[file:hashes.'SHA-256' = '5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480']

**Name**

4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3']

**Name**

3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1']

**Name**

29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd']

**Name**

140.82.33.130

**Description**

**ISP:** The Constant Company, LLC **OS:** Linux ---------- Services: **22:** ``` SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u4 Key type: ssh-rsa Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQDdPTzUicoJ111LZeexxmQwphU9WsHE90papaCe/fKFZBO2 whsqRc38/a2f+jB/sRvUQIYf6u+umxVYpdqrYQUNdNo87UOUGTUNE5ZI2EGpkzMAgv1Wtzrk8uI0Eos1tsN9JWYw1KG+5Sq7eF/GSUOjOYbrXRTHx9+oYU1wGpCPDyCzCCaPjj69x3cnyokwkvL/2SCBwweZ2XpWtEin5mfdOm7YXsPuWREcEuhvvfWLJKv4wCIk0lDWopeQxoQIitT1RXPVO6A9DrDfS3kZmRar+gIO7DNP65zfREP5zvYEEnrvF0pp+vKPD+C6EZT6Ozb7HY39AbO+jLksGmas69Ef Fingerprint: cb:e4:57:4d:3d:26:54:a3:7c:f1:a9:26:08:d3:8c:79 Kex Algorithms: curve25519-sha256 curve25519-sha256@libssh.org ecdh-sha2-nistp256 ecdh-sha2-nistp384 ecdh-sha2-nistp521 diffie-hellman-group-exchange-sha256 diffie-hellman-group16-sha512 diffie-hellman-group18-sha512 diffie-hellman-group14-sha256 diffie-hellman-group14-sha1 kex-strict-s-v00@openssh.com Server Host Key Algorithms: rsa-sha2-512 rsa-sha2-256 ssh-rsa Encryption Algorithms: chacha20-poly1305@openssh.com aes128-ctr aes192-ctr aes256-ctr aes128-gcm@openssh.com aes256-gcm@openssh.com MAC Algorithms: umac-64-etm@openssh.com umac-128-etm@openssh.com hmac-sha2-256-etm@openssh.com hmac-sha2-512-etm@openssh.com hmac-sha1-etm@openssh.com umac-64@openssh.com umac-128@openssh.com hmac-sha2-256 hmac-sha2-512 hmac-sha1 Compression Algorithms: none zlib@openssh.com ``` ---------- **80:** ``` HTTP/1.1 403 Forbidden Server: nginx Date: Sun, 21 Jan 2024 17:56:38 GMT Content-Type: text/html Content-Length: 342 Connection: keep-alive Vary: Accept-Encoding ETag: "6530f430-156" ``` ---------- **443:** ``` HTTP/1.1 403 Forbidden Server: nginx Date: Fri, 26 Jan 2024 12:51:47 GMT Content-Type: text/html Content-Length: 342 Connection: keep-alive Vary: Accept-Encoding ETag: "6530f430-156" ``` HEARTBLEED: 2024/01/26 12:51:59 140.82.33.130:443 - SAFE ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '140.82.33.130']

**Name**

1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778']

**Name**

1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071']

**Name**

0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d']

**Name**

0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf

**Pattern Type**

stix

**Pattern**

[file:hashes.'SHA-256' = '0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf']

**Name**

70.34.208.197

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '70.34.208.197']

**Name**

108.181.20.36

**Description**

**ISP:** Psychz Networks **OS:** Ubuntu ---------- Services: **80:** ``` HTTP/1.1 404 Not Found Server: nginx/1.18.0 (Ubuntu) Date: Sat, 03 Feb 2024 20:07:48 GMT Content-Type: text/html Content-Length: 564 Connection: keep-alive ``` ---------- **443:** ``` HTTP/1.1 200 OK Server: nginx/1.18.0 (Ubuntu) Date: Tue, 06 Feb 2024 12:11:12 GMT Content-Type: text/html; charset=UTF-8 Transfer-Encoding: chunked Connection: keep-alive X-Content-Type-Options: nosniff Access-Control-Allow-Origin: * Access-Control-Allow-Methods: GET, HEAD, POST ``` HEARTBLEED: 2024/02/06 12:11:31 108.181.20.36:443 - SAFE ----------

**Pattern Type**

stix

**Pattern**

[ipv4-addr:value = '108.181.20.36']

**Name**

7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea

**Description**

GoLandBuildPE SHA256 of e0f4afe374d75608d604fbf108eac64f

**Pattern Type**

stix

**Pattern**

```
[file:hashes.'SHA-256' =
'7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea']
```

# Malware

| Name |
| --- |
| Zardoor |

| Name |
| --- |
| Zardoor |

# Attack-Pattern

## Name

Dynamic-link Library Injection

## ID

T1055.001

## Description

Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. DLL injection is a method of executing arbitrary code in the address space of a separate live process. DLL injection is commonly performed by writing the path to a DLL in the virtual address space of the target process before loading the DLL by invoking a new thread. The write can be performed with native Windows API calls such as `VirtualAllocEx` and `WriteProcessMemory`, then invoked with `CreateRemoteThread` (which calls the `LoadLibrary` API responsible for loading the DLL). (Citation: Elastic Process Injection July 2017) Variations of this method such as reflective DLL injection (writing a self-mapping DLL into a process) and memory module (map DLL when writing into process) overcome the address relocation issue as well as the additional APIs to invoke execution (since these methods load and execute the files in memory by manually preforming the function of `LoadLibrary`).(Citation: Elastic HuntingNMemory June 2017)(Citation: Elastic Process Injection July 2017) Another variation of this method, often referred to as Module Stomping/Overloading or DLL Hollowing, may be leveraged to conceal injected code within a process. This method involves loading a legitimate DLL into a remote process then manually overwriting the module's `AddressOfEntryPoint` before starting a new thread in the target process.(Citation: Module Stomping for Shellcode Injection) This variation allows attackers to hide malicious injected code by potentially backing its execution with a legitimate DLL file on disk.(Citation: Hiding Malicious Code with Module Stomping) Running code in the context of another process may allow access to the process's memory, system/

network resources, and possibly elevated privileges. Execution via DLL injection may also evade detection from security products since the execution is masked under a legitimate process.

**Name**

DLL Side-Loading

**ID**

T1574.002

**Description**

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to [DLL Search Order Hijacking](https://attack.mitre.org/techniques/T1574/001), side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s). Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.(Citation: FireEye DLL Side-Loading)

**Name**

Remote System Discovery

**ID**

T1018

**Description**

Adversaries may attempt to get a listing of other systems by IP address, hostname, or other logical identifier on a network that may be used for Lateral Movement from the current system. Functionality could exist within remote access tools to enable this, but utilities available on the operating system could also be used such as [Ping](https://attack.mitre.org/software/S0097) or `net view` using [Net](https://attack.mitre.org/software/S0039). Adversaries may also analyze data from local host files (ex: `C:\Windows\System32\Drivers\etc\hosts` or `/etc/hosts`) or other passive means (such as local [Arp](https://attack.mitre.org/software/S0099) cache entries) in order to discover the presence of remote systems in an environment. Adversaries may also target discovery of network infrastructure as well as leverage [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands on network devices to gather detailed information about systems within a network (e.g. `show cdp neighbors`, `show arp`).(Citation: US-CERT-TA18-106A)(Citation: CISA AR21-126A FIVEHANDS May 2021)

**Name**

File Deletion

**ID**

T1070.004

**Description**

Adversaries may delete files left behind by the actions of their intrusion activity. Malware, tools, or other non-native files dropped or created on a system by an adversary (ex: [Ingress Tool Transfer](https://attack.mitre.org/techniques/T1105)) may leave traces to indicate to what was done within a network and how. Removal of these files can occur during an intrusion, or as part of a post-intrusion process to minimize the adversary's footprint. There are tools available from the host operating system to perform cleanup, but adversaries may use other tools as well.(Citation: Microsoft SDelete July 2016) Examples of built-in [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059) functions include `del` on Windows and `rm` or `unlink` on Linux and macOS.

**Name**

Windows Command Shell

**ID**

T1059.003

**Description**

Adversaries may abuse the Windows command shell for execution. The Windows command shell ([cmd](https://attack.mitre.org/software/S0106)) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels required for different subsets of commands. The command prompt can be invoked remotely via [Remote Services](https://attack.mitre.org/techniques/T1021) such as [SSH](https://attack.mitre.org/techniques/T1021/004).(Citation: SSH in Windows) Batch files (ex: .bat or .cmd) also provide the shell with a list of sequential commands to run, as well as normal scripting operations such as conditionals and loops. Common uses of batch files include long or repetitive tasks, or the need to run the same set of commands on multiple systems. Adversaries may leverage [cmd](https://attack.mitre.org/software/S0106) to execute various commands and payloads. Common uses include [cmd](https://attack.mitre.org/software/S0106) to execute a single command, or abusing [cmd](https://attack.mitre.org/software/S0106) interactively with input and output forwarded over a command and control channel.

**Name**

Multi-hop Proxy

**ID**

T1090.003

**Description**

To disguise the source of malicious traffic, adversaries may chain together multiple proxies. Typically, a defender will be able to identify the last proxy traffic traversed before it enters their network; the defender may or may not be able to identify any previous proxies before the last-hop proxy. This technique makes identifying the original source of the malicious traffic even more difficult by requiring the defender to trace malicious traffic through several proxies to identify its source. A particular variant of this behavior is to use onion routing networks, such as the publicly available TOR network. (Citation: Onion Routing) In the case of network infrastructure, particularly routers, it is possible for an

adversary to leverage multiple compromised devices to create a multi-hop proxy chain within the Wide-Area Network (WAN) of the enterprise. By leveraging [Patch System Image](https://attack.mitre.org/techniques/T1601/001), adversaries can add custom code to the affected network devices that will implement onion routing between those nodes. This custom onion routing network will transport the encrypted C2 traffic through the compromised population, allowing adversaries to communicate with any device within the onion routing network. This method is dependent upon the [Network Boundary Bridging](https://attack.mitre.org/techniques/T1599) method in order to allow the adversaries to cross the protected network boundary of the Internet perimeter and into the organization's WAN. Protocols such as ICMP may be used as a transport.

**Name**

Exfiltration Over Alternative Protocol

**ID**

T1048

**Description**

Adversaries may steal data by exfiltrating it over a different protocol than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server. Alternate protocols include FTP, SMTP, HTTP/S, DNS, SMB, or any other network protocol not being used as the main command and control channel. Adversaries may also opt to encrypt and/or obfuscate these alternate channels. [Exfiltration Over Alternative Protocol](https://attack.mitre.org/techniques/T1048) can be done using various common operating system utilities such as [Net](https://attack.mitre.org/software/S0039)/SMB or FTP.(Citation: Palo Alto OilRig Oct 2016) On macOS and Linux `curl` may be used to invoke protocols such as HTTP/S or FTP/S to exfiltrate data from a system.(Citation: 20 macOS Common Tools and Techniques) Many IaaS and SaaS platforms (such as Microsoft Exchange, Microsoft SharePoint, GitHub, and AWS S3) support the direct download of files, emails, source code, and other sensitive information via the web console or [Cloud API](https://attack.mitre.org/techniques/T1059/009).

**Name**

Process Discovery

**ID**

T1057

**Description**

Adversaries may attempt to get information about running processes on a system. Information obtained could be used to gain an understanding of common software/applications running on systems within the network. Adversaries may use the information from [Process Discovery](https://attack.mitre.org/techniques/T1057) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. In Windows environments, adversaries could obtain details on running processes using the [Tasklist](https://attack.mitre.org/software/S0057) utility via [cmd](https://attack.mitre.org/software/S0106) or `Get-Process` via [PowerShell](https://attack.mitre.org/techniques/T1059/001). Information about processes can also be extracted from the output of [Native API](https://attack.mitre.org/techniques/T1106) calls such as `CreateToolhelp32Snapshot`. In Mac and Linux, this is accomplished with the `ps` command. Adversaries may also opt to enumerate processes via /proc. On network devices, [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands such as `show processes` can be used to display current running processes.(Citation: US-CERT-TA18-106A)(Citation: show_processes_cisco_cmd)

**Name**

System Network Connections Discovery

**ID**

T1049

**Description**

Adversaries may attempt to get a listing of network connections to or from the compromised system they are currently accessing or from remote systems by querying for information over the network. An adversary who gains access to a system that is part of a cloud-based environment may map out Virtual Private Clouds or Virtual Networks in order to determine what systems and services are connected. The actions performed are likely the same types of discovery techniques depending on the operating system, but the resulting information may include details about the networked cloud environment

relevant to the adversary's goals. Cloud providers may have different ways in which their virtual networks operate.(Citation: Amazon AWS VPC Guide)(Citation: Microsoft Azure Virtual Network Overview)(Citation: Google VPC Overview) Similarly, adversaries who gain access to network devices may also perform similar discovery activities to gather information about connected systems and services. Utilities and commands that acquire this information include [netstat](https://attack.mitre.org/software/S0104), "net use," and "net session" with [Net](https://attack.mitre.org/software/S0039). In Mac and Linux, [netstat](https://attack.mitre.org/software/S0104) and `lsof` can be used to list current connections. `who -a` and `w` can be used to show which users are currently logged in, similar to "net session". Additionally, built-in features native to network devices and [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) may be used (e.g. `show ip sockets`, `show tcp brief`).(Citation: US-CERT-TA18-106A)

**Name**

Ingress Tool Transfer

**ID**

T1105

**Description**

Adversaries may transfer tools or other files from an external system into a compromised environment. Tools or files may be copied from an external adversary-controlled system to the victim network through the command and control channel or through alternate protocols such as [ftp](https://attack.mitre.org/software/S0095). Once present, adversaries may also transfer/spread tools between victim devices within a compromised environment (i.e. [Lateral Tool Transfer](https://attack.mitre.org/techniques/T1570)). On Windows, adversaries may use various utilities to download tools, such as `copy`, `finger`, [certutil](https://attack.mitre.org/software/S0160), and [PowerShell](https://attack.mitre.org/techniques/T1059/001) commands such as `IEX(New-Object Net.WebClient).downloadString()` and `Invoke-WebRequest`. On Linux and macOS systems, a variety of utilities also exist, such as `curl`, `scp`, `sftp`, `tftp`, `rsync`, `finger`, and `wget`.(Citation: t1105_lolbas) Adversaries may also abuse installers and package managers, such as `yum` or `winget`, to download tools to victim hosts. Files can also be transferred using various [Web Service](https://attack.mitre.org/techniques/T1102)s as well as native or otherwise present tools on the victim system.(Citation: PTSecurity Cobalt Dec 2016) In some cases, adversaries may be able to leverage services that sync between a web-based and an on-premises client, such as Dropbox or OneDrive, to transfer files onto victim systems. For example, by compromising a cloud account and logging into the service's web portal,

an adversary may be able to trigger an automatic syncing process that transfers the file onto the victim's machine.(Citation: Dropbox Malware Sync)

**Name**

Process Injection

**ID**

T1055

**Description**

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process. There are many different ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific. More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

**Name**

Malicious File

**ID**

T1204.002

**Description**

An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from

[Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001). Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl. Adversaries may employ various forms of [Masquerading](https://attack.mitre.org/techniques/T1036) and [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to increase the likelihood that a user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.(Citation: Password Protected Word Docs) While [Malicious File](https://attack.mitre.org/techniques/T1204/002) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534).

**Name**

Domain Account

**ID**

T1087.002

**Description**

Adversaries may attempt to get a listing of domain accounts. This information can help adversaries determine which domain accounts exist to aid in follow-on behavior such as targeting specific accounts which possess particular privileges. Commands such as `net user /domain` and `net group /domain` of the [Net](https://attack.mitre.org/software/S0039) utility, `dscacheutil -q group`on macOS, and `ldapsearch` on Linux can list domain users and groups. [PowerShell](https://attack.mitre.org/techniques/T1059/001) cmdlets including `Get-ADUser` and `Get-ADGroupMember` may enumerate members of Active Directory groups.

**Name**

System Owner/User Discovery

**ID**

T1033

**Description**

Adversaries may attempt to identify the primary user, currently logged in user, set of users that commonly uses a system, or whether a user is actively using the system. They may do this, for example, by retrieving account usernames or by using [OS Credential Dumping](https://attack.mitre.org/techniques/T1003). The information may be collected in a number of different ways using other Discovery techniques, because user and username details are prevalent throughout a system and include running process ownership, file/directory ownership, session information, and system logs. Adversaries may use the information from [System Owner/User Discovery](https://attack.mitre.org/techniques/T1033) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. Various utilities and commands may acquire this information, including `whoami`. In macOS and Linux, the currently logged in user can be identified with `w` and `who`. On macOS the `dscl . list /Users | grep -v '_'` command can also be used to enumerate user accounts. Environment variables, such as `%USERNAME%` and `$USER`, may also be used to access this information. On network devices, [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands such as `show users` and `show ssh` can be used to display users currently logged into the device.(Citation: show_ssh_users_cmd_cisco)(Citation: US-CERT TA18-106A Network Infrastructure Devices 2018)

**Name**

Windows Management Instrumentation

**ID**

T1047

**Description**

Adversaries may abuse Windows Management Instrumentation (WMI) to execute malicious commands and payloads. WMI is an administration feature that provides a uniform environment to access Windows system components. The WMI service enables both local and remote access, though the latter is facilitated by [Remote Services](https://attack.mitre.org/techniques/T1021) such as [Distributed Component Object Model](https://attack.mitre.org/techniques/T1021/003) (DCOM) and [Windows Remote Management]

(https://attack.mitre.org/techniques/T1021/006) (WinRM).(Citation: MSDN WMI) Remote WMI over DCOM operates using port 135, whereas WMI over WinRM operates over port 5985 when using HTTP and 5986 for HTTPS.(Citation: MSDN WMI)(Citation: FireEye WMI 2015) An adversary can use WMI to interact with local and remote systems and use it as a means to execute various behaviors, such as gathering information for Discovery as well as remote Execution of files as part of Lateral Movement. (Citation: FireEye WMI SANS 2015) (Citation: FireEye WMI 2015)

**Name**

Scheduled Task

**ID**

T1053.005

**Description**

Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The [schtasks](https://attack.mitre.org/software/S0111) utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task. The deprecated [at](https://attack.mitre.org/software/S0110) utility could also be abused by adversaries (ex: [At](https://attack.mitre.org/techniques/T1053/002)), though `at.exe` can not access tasks created with `schtasks` or the Control Panel. An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to [System Binary Proxy Execution](https://attack.mitre.org/techniques/T1218), adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.(Citation: ProofPoint Serpent) Adversaries may also create "hidden" scheduled tasks (i.e. [Hide Artifacts](https://attack.mitre.org/techniques/T1564)) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions).(Citation: SigmaHQ)(Citation: Tarrask scheduled task) Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., `Index`

value) within associated registry keys.(Citation: Defending Against Scheduled Task Attacks in Windows Environments)

**Name**

Dynamic-link Library Injection

**ID**

T1055.001

**Description**

Adversaries may inject dynamic-link libraries (DLLs) into processes in order to evade process-based defenses as well as possibly elevate privileges. DLL injection is a method of executing arbitrary code in the address space of a separate live process. DLL injection is commonly performed by writing the path to a DLL in the virtual address space of the target process before loading the DLL by invoking a new thread. The write can be performed with native Windows API calls such as `VirtualAllocEx` and `WriteProcessMemory`, then invoked with `CreateRemoteThread` (which calls the `LoadLibrary` API responsible for loading the DLL). (Citation: Elastic Process Injection July 2017) Variations of this method such as reflective DLL injection (writing a self-mapping DLL into a process) and memory module (map DLL when writing into process) overcome the address relocation issue as well as the additional APIs to invoke execution (since these methods load and execute the files in memory by manually preforming the function of `LoadLibrary`).(Citation: Elastic HuntingNMemory June 2017)(Citation: Elastic Process Injection July 2017) Another variation of this method, often referred to as Module Stomping/Overloading or DLL Hollowing, may be leveraged to conceal injected code within a process. This method involves loading a legitimate DLL into a remote process then manually overwriting the module's `AddressOfEntryPoint` before starting a new thread in the target process.(Citation: Module Stomping for Shellcode Injection) This variation allows attackers to hide malicious injected code by potentially backing its execution with a legitimate DLL file on disk.(Citation: Hiding Malicious Code with Module Stomping) Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via DLL injection may also evade detection from security products since the execution is masked under a legitimate process.

**Name**

DLL Side-Loading

**ID**

T1574.002

**Description**

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to [DLL Search Order Hijacking](https://attack.mitre.org/techniques/T1574/001), side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s). Side-loading takes advantage of the DLL search order used by the loader by positioning both the victim application and malicious payload(s) alongside each other. Adversaries likely use side-loading as a means of masking actions they perform under a legitimate, trusted, and potentially elevated system or software process. Benign executables used to side-load payloads may not be flagged during delivery and/or execution. Adversary payloads may also be encrypted/packed or otherwise obfuscated until loaded into the memory of the trusted process.(Citation: FireEye DLL Side-Loading)

**Name**

Remote System Discovery

**ID**

T1018

**Description**

Adversaries may attempt to get a listing of other systems by IP address, hostname, or other logical identifier on a network that may be used for Lateral Movement from the current system. Functionality could exist within remote access tools to enable this, but utilities available on the operating system could also be used such as [Ping](https://attack.mitre.org/software/S0097) or `net view` using [Net](https://attack.mitre.org/software/S0039). Adversaries may also analyze data from local host files (ex: `C:

\Windows\System32\Drivers\etc\hosts` or `/etc/hosts`) or other passive means (such as local [Arp](https://attack.mitre.org/software/S0099) cache entries) in order to discover the presence of remote systems in an environment. Adversaries may also target discovery of network infrastructure as well as leverage [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands on network devices to gather detailed information about systems within a network (e.g. `show cdp neighbors`, `show arp`).(Citation: US-CERT-TA18-106A)(Citation: CISA AR21-126A FIVEHANDS May 2021)

**Name**

File Deletion

**ID**

T1070.004

**Description**

Adversaries may delete files left behind by the actions of their intrusion activity. Malware, tools, or other non-native files dropped or created on a system by an adversary (ex: [Ingress Tool Transfer](https://attack.mitre.org/techniques/T1105)) may leave traces to indicate to what was done within a network and how. Removal of these files can occur during an intrusion, or as part of a post-intrusion process to minimize the adversary's footprint. There are tools available from the host operating system to perform cleanup, but adversaries may use other tools as well.(Citation: Microsoft SDelete July 2016) Examples of built-in [Command and Scripting Interpreter](https://attack.mitre.org/techniques/T1059) functions include `del` on Windows and `rm` or `unlink` on Linux and macOS.

**Name**

Windows Command Shell

**ID**

T1059.003

**Description**

Adversaries may abuse the Windows command shell for execution. The Windows command shell ([cmd](https://attack.mitre.org/software/S0106)) is the primary command prompt on Windows systems. The Windows command prompt can be used to control almost any aspect of a system, with various permission levels required for different subsets of commands. The command prompt can be invoked remotely via [Remote Services](https://attack.mitre.org/techniques/T1021) such as [SSH](https://attack.mitre.org/techniques/T1021/004).(Citation: SSH in Windows) Batch files (ex: .bat or .cmd) also provide the shell with a list of sequential commands to run, as well as normal scripting operations such as conditionals and loops. Common uses of batch files include long or repetitive tasks, or the need to run the same set of commands on multiple systems. Adversaries may leverage [cmd](https://attack.mitre.org/software/S0106) to execute various commands and payloads. Common uses include [cmd](https://attack.mitre.org/software/S0106) to execute a single command, or abusing [cmd](https://attack.mitre.org/software/S0106) interactively with input and output forwarded over a command and control channel.

**Name**

Multi-hop Proxy

**ID**

T1090.003

**Description**

To disguise the source of malicious traffic, adversaries may chain together multiple proxies. Typically, a defender will be able to identify the last proxy traffic traversed before it enters their network; the defender may or may not be able to identify any previous proxies before the last-hop proxy. This technique makes identifying the original source of the malicious traffic even more difficult by requiring the defender to trace malicious traffic through several proxies to identify its source. A particular variant of this behavior is to use onion routing networks, such as the publicly available TOR network. (Citation: Onion Routing) In the case of network infrastructure, particularly routers, it is possible for an adversary to leverage multiple compromised devices to create a multi-hop proxy chain within the Wide-Area Network (WAN) of the enterprise. By leveraging [Patch System Image](https://attack.mitre.org/techniques/T1601/001), adversaries can add custom code to the affected network devices that will implement onion routing between those nodes. This custom onion routing network will transport the encrypted C2 traffic through the compromised population, allowing adversaries to communicate with any device within the onion routing network. This method is dependent upon the [Network Boundary Bridging]

(https://attack.mitre.org/techniques/T1599) method in order to allow the adversaries to cross the protected network boundary of the Internet perimeter and into the organization's WAN. Protocols such as ICMP may be used as a transport.

**Name**

Exfiltration Over Alternative Protocol

**ID**

T1048

**Description**

Adversaries may steal data by exfiltrating it over a different protocol than that of the existing command and control channel. The data may also be sent to an alternate network location from the main command and control server. Alternate protocols include FTP, SMTP, HTTP/S, DNS, SMB, or any other network protocol not being used as the main command and control channel. Adversaries may also opt to encrypt and/or obfuscate these alternate channels. [Exfiltration Over Alternative Protocol](https://attack.mitre.org/techniques/T1048) can be done using various common operating system utilities such as [Net](https://attack.mitre.org/software/S0039)/SMB or FTP.(Citation: Palo Alto OilRig Oct 2016) On macOS and Linux `curl` may be used to invoke protocols such as HTTP/S or FTP/S to exfiltrate data from a system.(Citation: 20 macOS Common Tools and Techniques) Many IaaS and SaaS platforms (such as Microsoft Exchange, Microsoft SharePoint, GitHub, and AWS S3) support the direct download of files, emails, source code, and other sensitive information via the web console or [Cloud API](https://attack.mitre.org/techniques/T1059/009).

**Name**

Process Discovery

**ID**

T1057

**Description**

Adversaries may attempt to get information about running processes on a system. Information obtained could be used to gain an understanding of common software/applications running on systems within the network. Adversaries may use the information from [Process Discovery](https://attack.mitre.org/techniques/T1057) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. In Windows environments, adversaries could obtain details on running processes using the [Tasklist](https://attack.mitre.org/software/S0057) utility via [cmd](https://attack.mitre.org/software/S0106) or `Get-Process` via [PowerShell](https://attack.mitre.org/techniques/T1059/001). Information about processes can also be extracted from the output of [Native API](https://attack.mitre.org/techniques/T1106) calls such as `CreateToolhelp32Snapshot`. In Mac and Linux, this is accomplished with the `ps` command. Adversaries may also opt to enumerate processes via /proc. On network devices, [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands such as `show processes` can be used to display current running processes.(Citation: US-CERT-TA18-106A)(Citation: show_processes_cisco_cmd)

**Name**

System Network Connections Discovery

**ID**

T1049

**Description**

Adversaries may attempt to get a listing of network connections to or from the compromised system they are currently accessing or from remote systems by querying for information over the network. An adversary who gains access to a system that is part of a cloud-based environment may map out Virtual Private Clouds or Virtual Networks in order to determine what systems and services are connected. The actions performed are likely the same types of discovery techniques depending on the operating system, but the resulting information may include details about the networked cloud environment relevant to the adversary's goals. Cloud providers may have different ways in which their virtual networks operate.(Citation: Amazon AWS VPC Guide)(Citation: Microsoft Azure Virtual Network Overview)(Citation: Google VPC Overview) Similarly, adversaries who gain access to network devices may also perform similar discovery activities to gather information about connected systems and services. Utilities and commands that acquire this information include [netstat](https://attack.mitre.org/software/S0104), "net use," and "net session" with [Net](https://attack.mitre.org/software/S0039). In Mac and Linux, [netstat]

(https://attack.mitre.org/software/S0104) and `lsof` can be used to list current connections. `who -a` and `w` can be used to show which users are currently logged in, similar to "net session". Additionally, built-in features native to network devices and [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) may be used (e.g. `show ip sockets`, `show tcp brief`).(Citation: US-CERT-TA18-106A)

**Name**

Ingress Tool Transfer

**ID**

T1105

**Description**

Adversaries may transfer tools or other files from an external system into a compromised environment. Tools or files may be copied from an external adversary-controlled system to the victim network through the command and control channel or through alternate protocols such as [ftp](https://attack.mitre.org/software/S0095). Once present, adversaries may also transfer/spread tools between victim devices within a compromised environment (i.e. [Lateral Tool Transfer](https://attack.mitre.org/techniques/T1570)). On Windows, adversaries may use various utilities to download tools, such as `copy`, `finger`, [certutil](https://attack.mitre.org/software/S0160), and [PowerShell](https://attack.mitre.org/techniques/T1059/001) commands such as `IEX(New-Object Net.WebClient).downloadString()` and `Invoke-WebRequest`. On Linux and macOS systems, a variety of utilities also exist, such as `curl`, `scp`, `sftp`, `tftp`, `rsync`, `finger`, and `wget`.(Citation: t1105_lolbas) Adversaries may also abuse installers and package managers, such as `yum` or `winget`, to download tools to victim hosts. Files can also be transferred using various [Web Service](https://attack.mitre.org/techniques/T1102)s as well as native or otherwise present tools on the victim system.(Citation: PTSecurity Cobalt Dec 2016) In some cases, adversaries may be able to leverage services that sync between a web-based and an on-premises client, such as Dropbox or OneDrive, to transfer files onto victim systems. For example, by compromising a cloud account and logging into the service's web portal, an adversary may be able to trigger an automatic syncing process that transfers the file onto the victim's machine.(Citation: Dropbox Malware Sync)

**Name**

Process Injection

**ID**

T1055

**Description**

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process. There are many different ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific. More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

**Name**

Malicious File

**ID**

T1204.002

**Description**

An adversary may rely upon a user opening a malicious file in order to gain execution. Users may be subjected to social engineering to get them to open a file that will lead to code execution. This user action will typically be observed as follow-on behavior from [Spearphishing Attachment](https://attack.mitre.org/techniques/T1566/001). Adversaries may use several types of files that require a user to execute them, including .doc, .pdf, .xls, .rtf, .scr, .exe, .lnk, .pif, and .cpl. Adversaries may employ various forms of [Masquerading](https://attack.mitre.org/techniques/T1036) and [Obfuscated Files or Information](https://attack.mitre.org/techniques/T1027) to increase the likelihood that a

user will open and successfully execute a malicious file. These methods may include using a familiar naming convention and/or password protecting the file and supplying instructions to a user on how to open it.(Citation: Password Protected Word Docs) While [Malicious File](https://attack.mitre.org/techniques/T1204/002) frequently occurs shortly after Initial Access it may occur at other phases of an intrusion, such as when an adversary places a file in a shared directory or on a user's desktop hoping that a user will click on it. This activity may also be seen shortly after [Internal Spearphishing](https://attack.mitre.org/techniques/T1534).

**Name**

Domain Account

**ID**

T1087.002

**Description**

Adversaries may attempt to get a listing of domain accounts. This information can help adversaries determine which domain accounts exist to aid in follow-on behavior such as targeting specific accounts which possess particular privileges. Commands such as `net user /domain` and `net group /domain` of the [Net](https://attack.mitre.org/software/S0039) utility, `dscacheutil -q group`on macOS, and `ldapsearch` on Linux can list domain users and groups. [PowerShell](https://attack.mitre.org/techniques/T1059/001) cmdlets including `Get-ADUser` and `Get-ADGroupMember` may enumerate members of Active Directory groups.

**Name**

System Owner/User Discovery

**ID**

T1033

**Description**

Adversaries may attempt to identify the primary user, currently logged in user, set of users that commonly uses a system, or whether a user is actively using the system. They may do this, for example, by retrieving account usernames or by using [OS Credential Dumping](https://attack.mitre.org/techniques/T1003). The information may be collected in a number of different ways using other Discovery techniques, because user and username details are prevalent throughout a system and include running process ownership, file/directory ownership, session information, and system logs. Adversaries may use the information from [System Owner/User Discovery](https://attack.mitre.org/techniques/T1033) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions. Various utilities and commands may acquire this information, including `whoami`. In macOS and Linux, the currently logged in user can be identified with `w` and `who`. On macOS the `dscl . list /Users | grep -v '_'` command can also be used to enumerate user accounts. Environment variables, such as `%USERNAME%` and `$USER`, may also be used to access this information. On network devices, [Network Device CLI](https://attack.mitre.org/techniques/T1059/008) commands such as `show users` and `show ssh` can be used to display users currently logged into the device.(Citation: show_ssh_users_cmd_cisco)(Citation: US-CERT TA18-106A Network Infrastructure Devices 2018)

**Name**

Windows Management Instrumentation

**ID**

T1047

**Description**

Adversaries may abuse Windows Management Instrumentation (WMI) to execute malicious commands and payloads. WMI is an administration feature that provides a uniform environment to access Windows system components. The WMI service enables both local and remote access, though the latter is facilitated by [Remote Services](https://attack.mitre.org/techniques/T1021) such as [Distributed Component Object Model](https://attack.mitre.org/techniques/T1021/003) (DCOM) and [Windows Remote Management](https://attack.mitre.org/techniques/T1021/006) (WinRM).(Citation: MSDN WMI) Remote WMI over DCOM operates using port 135, whereas WMI over WinRM operates over port 5985 when using HTTP and 5986 for HTTPS.(Citation: MSDN WMI)(Citation: FireEye WMI 2015) An adversary can use WMI to interact with local and remote systems and use it as a means to execute various behaviors, such as gathering information for Discovery as well as remote

Execution of files as part of Lateral Movement. (Citation: FireEye WMI SANS 2015) (Citation: FireEye WMI 2015)

**Name**

Scheduled Task

**ID**

T1053.005

**Description**

Adversaries may abuse the Windows Task Scheduler to perform task scheduling for initial or recurring execution of malicious code. There are multiple ways to access the Task Scheduler in Windows. The [schtasks](https://attack.mitre.org/software/S0111) utility can be run directly on the command line, or the Task Scheduler can be opened through the GUI within the Administrator Tools section of the Control Panel. In some cases, adversaries have used a .NET wrapper for the Windows Task Scheduler, and alternatively, adversaries have used the Windows netapi32 library to create a scheduled task. The deprecated [at](https://attack.mitre.org/software/S0110) utility could also be abused by adversaries (ex: [At](https://attack.mitre.org/techniques/T1053/002)), though `at.exe` can not access tasks created with `schtasks` or the Control Panel. An adversary may use Windows Task Scheduler to execute programs at system startup or on a scheduled basis for persistence. The Windows Task Scheduler can also be abused to conduct remote Execution as part of Lateral Movement and/or to run a process under the context of a specified account (such as SYSTEM). Similar to [System Binary Proxy Execution](https://attack.mitre.org/techniques/T1218), adversaries have also abused the Windows Task Scheduler to potentially mask one-time execution under signed/trusted system processes.(Citation: ProofPoint Serpent) Adversaries may also create "hidden" scheduled tasks (i.e. [Hide Artifacts](https://attack.mitre.org/techniques/T1564)) that may not be visible to defender tools and manual queries used to enumerate tasks. Specifically, an adversary may hide a task from `schtasks /query` and the Task Scheduler by deleting the associated Security Descriptor (SD) registry value (where deletion of this value must be completed using SYSTEM permissions).(Citation: SigmaHQ)(Citation: Tarrask scheduled task) Adversaries may also employ alternate methods to hide tasks, such as altering the metadata (e.g., `Index` value) within associated registry keys.(Citation: Defending Against Scheduled Task Attacks in Windows Environments)

# Country

| Name |
| --- |
| Saudi Arabia |

| Name |
| --- |
| Saudi Arabia |

# Sector

| Name |
| --- |
| Non-Governmental Organizations (NGOs) |
| **Description** |
| A legally constituted non-commercial organization created by natural or legal persons with no participation or representation of any government. |
| **Name** |
| Non-Governmental Organizations (NGOs) |
| **Description** |
| A legally constituted non-commercial organization created by natural or legal persons with no participation or representation of any government. |

# IPv4-Addr

| Value |
| --- |
| 70.34.195.221 |
| 70.34.194.185 |
| 217.69.1.128 |
| 208.85.20.130 |
| 140.82.33.130 |
| 139.84.232.245 |
| 139.84.229.192 |
| 108.61.189.125 |
| 108.181.20.36 |
| 70.34.208.197 |
| 70.34.195.221 |
| 70.34.194.185 |
| 217.69.1.128 |

| |
|---|
| 208.85.20.130 |
| 140.82.33.130 |
| 139.84.232.245 |
| 139.84.229.192 |
| 108.61.189.125 |
| 108.181.20.36 |
| 70.34.208.197 |

# StixFile

| Value |
| --- |
| f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2 |
| d7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87 |
| d5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71 |
| d267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef |
| c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74 |
| b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897 |
| a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d |
| 7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3 |
| 73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095 |
| 5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852 |
| 5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e |
| 5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480 |
| 4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3 |

3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1

29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd

1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778

1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071

0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d

0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf

7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea

f71f7c68209ea8218463df397e5c39ef5f916f138dc001feb3a60ef585bd2ac2

d7dfa7009a9d808b744df8ed4f5852bd03ffb82f7a07a258ea8b5e0290fb7d87

d5d16d9bb75d461922eade2597c233255871dc74659f0169f3d3f40f5273ab71

d267e2a6311fe4e2dfd0237652223add300b9a5233b555e131325a2612e1d7ef

c6419df4bbda5b75ea4a0b8e8acd2100b149443584390c91a218e7735561ef74

b5b3627606a5c5e720fa32fb9cb90aa813c630673d23c97a81012b832799a897

a99a9f2853ff0ca5b91767096c7f7e977b43e62dd93bde6d79e3407bc01f661d

7abf74260ae5b771182e95bc360fefa1b635b56b3aa05922506d55c5d15517c3

73c7459e0c3ba00c0566f7baa710dd8b88ef3cf75ee0e76d36c5d8cd73083095

5eeab7b795a3303c368c72ef09a345f3a4f02301ec443e98319d600e8287e852

5655a2981fa4821fe09c997c84839c16d582d65243c782f45e14c96a977c594e

5226b67b5d49720981841fab64794533fe0530409ba2975e6125a4bc008f2480

4b16ea1b1273f8746cf399c71bfc1f5bff7378b5414b4ea044c55e0ee08c89d3

3adcc81446f0e8ed1a2bc1e815613eb5622afba57941d651faa2b5bc4b2f13c1

29741f7987ab61b85adb310a7ab2f44405822f1719fa431c8f49007b64f6f5cd

1aea1e7098221f2cc76ccd45078d9a216236b4e7e295dfa68e8a25aab3abe778

1480b2038395f9edd2c21dff68eb29a4d6177708b70b687f758af60c8b02f071

0a5aa03e35d6d9218342b2bec753a9800570c000964801cf6bfe45a9bb393c0d

0058d495254bf3760b30b5950d646f9a38506cef8f297c49c3b73c208ab723bf

7905bd9bb4d277a81935a22f975a0030faa9e5c9dbb9f6152c2f56ba1cd0cdea

# External References

- https://raw.githubusercontent.com/Cisco-Talos/IOCs/main/2024/01/new-zardoor-backdoor.txt
- https://blog.talosintelligence.com/new-zardoor-backdoor/
- https://otx.alienvault.com/pulse/65c4f5c71ba1f7039cfb41c6