National Cyber Security Centre
a part of GCHQ

Turla group using Neuron and Nautilus tools alongside Snake malware

Version 2.0

Reference: NCSC-Ops/35-17

23 November 2017

About this document

This report provides new intelligence by the NCSC on two tools used by the Turla group to target the UK. It contains IOCs and signatures for detection by network defenders.

Handling of the Report

Information in this report has been given a Traffic Light Protocol (TLP) of WHITE, which means it can be shared within and beyond the CiSP community with no handling restrictions.

Disclaimer

This report draws on reported information and NCSC investigations into Turla activity in the UK.

Contents

Introduction

Neuron and Nautilus are malicious tools designed to operate on Microsoft Windows platforms, primarily targeting mail servers and web servers. The NCSC has observed these tools being used by the Turla group to maintain persistent network access and to conduct network operations.

The Turla group use a range of tools and techniques, many of which are custom. Using their advanced toolkit, the Turla group compromise networks for the purposes of intelligence collection. The Turla group is known to target government, military, technology, energy and commercial organisations.

The Turla group has operated on targets using a rootkit known as Snake for many years. Like Neuron and Nautilus, Snake provides a platform to steal sensitive data, acts as a gateway for internal network operations and is used to conduct onward attacks against other organisations.

The Turla group are experienced in maintaining covert access through incident response activities. They infect multiple systems within target networks and deploy a diverse range of tools to ensure that they retain a foothold back onto a victim even after the initial infection vector has been mitigated.

The NCSC has observed both Neuron and Nautilus being used in conjunction with the Snake rootkit. In a number of instances, one or both of these tools has been deployed following the successful installation of Snake. The NCSC believes that Neuron and Nautilus are another component of the wider Turla campaign and are not acting as replacements for the Snake rootkit. It is likely that these tools have seen wider deployment since the Snake rootkit has been reported on by the information security industry, providing the group with additional methods of access.

This advisory provides information to detect Neuron and Nautilus infections. The NCSC encourages any organisation that has previously experienced a compromise by the Turla group to be diligent in checking for the presence of these additional tools. Whilst they are commonly deployed alongside the Snake rootkit, these tools can also be operated independently.

Neuron Analysis

Neuron consists of both client and server components. The Neuron client and Neuron service are written using the .NET framework with some codebase overlaps.

The Neuron client is used to infect victim endpoints and extract sensitive information from local client machines. The Neuron server is used to infect network infrastructure such as mail and web servers, and acts as local Command & Control (C2) for the client component. Establishing a local C2 limits interaction with the target network and remote hosts. It also reduces the log footprint of actor infrastructure and enables client interaction to appear more convincing as the traffic is contained within the target network.

The main method of communication between the Neuron client and service is via HTTP requests. The Neuron service creates its own HTTP listener and waits for requests to a configured Neuron URL endpoint. These endpoint names are themed around legitimate web services, such as Microsoft Exchange and Microsoft IIS, which further helps malware traffic appear legitimate. Details of these endpoints are provided in the Neuron service communications section of this advisory.

A subset of Neuron services analysed by the NCSC can receive communications via pipes alongside the HTTP listener, however this functionality is missing from some samples.

One of the main pieces of functionality implemented within Neuron is the synchronising of “StorageFile” objects and “StorageScript” objects between the client and service. These are described in more detail in the Network Communications section.

This malware is referred to as “Neuron” due to the presence of a PDB string within the binary and various other references throughout.

```
c:\Develop\internal\neuron-client\dropper-svc\obj\Release\dropper-svc.pdb
```

Neuron Service

The Neuron service is typically installed on compromised infrastructure such as mail and web servers, and listens for HTTP requests from infected clients. In this way, Neuron service acts as a Command & Control (C2) server inside the victim network for infected Neuron clients. While Neuron service examples observed by the NCSC have been running on servers, it is also possible for it to be run on Windows clients.

The installation of a C2 server inside the victim network allows the actor to evade detection by network gateway based monitoring. While external communications are required for the actor to make connections back to their upstream C2 infrastructure, these communications are often encrypted using the legitimate TLS configuration of the victim network.

The Neuron service and client model enables the communications to appear legitimate, with endpoint victims running the client, and the actor initiating connections to the (typically) outward-facing Neuron infected server.

Associated Files

Name	Microsoft.Exchange.Service.exe
Description	Neuron Service
MD5	0f12268221e27406351a6313f902b498
SHA1	b0dbdc81a0e367330007b7e593d8dabf92ca7afd
SHA256	d1d7a96fcadc137e80ad866c838502713db9cdfe59939342b8e3beacf9c7fe29
Size	43008

Name	w3wpdiag.exe
Description	Neuron Service
MD5	371b4380080e3d94ffcae1a7e9a0d5e2
SHA1	f7088075d1c798f27b0d269c97dc877ff16f1401
SHA256	2986bae15cfa78b919d21dc070be944e949a027e8047a812026e35c66ab17353
Size	59392

Name	Updater.exe
Description	Neuron Service
MD5	8229622a9790d75e09a099e8758d5703
SHA1	10586913ceeecd408da4e656c29ed4e91c6b758e
SHA256	2f4d6a3c87770c7d42d1a1b71ed021a083b08f69ccaf63c15428c7bc6f69cb10
Size	44544

Name	w3wpdiag.exe
Description	Neuron Service
MD5	a3bdc385cf68019449027bd6d8cecb4d
SHA1	fe8da5a1e62a8d4f627834b0f26c802a330d8d45
SHA256	0f4e9e391696ed8b9172985bb43cca7d7f2c8a4ae0493e4bf1f15b90f7138259
Size	58880

Name	dropper-svc.exe
Description	Dropper for the Neuron service
MD5	d6ef3c8f2c3f3ddffbb70f5dadfa982c
SHA1	934b288075c122165897276b360c61e77cb7bde0
SHA256	fa543de359d498150cbcb67c1631e726a4b14b0a859573185cede5b12ad2abfb
Size	85008

Infection Vector & Install

The infection vector for the Neuron service is typically via exploitation of application layer vulnerabilities in server software, server misconfigurations, or brute-force attacks on administrative accounts.

Neuron service requires a dropper that essentially performs the same actions as the client dropper, embedding the final payload using the same method detailed in the Neuron client section. The service dropper takes a parameter of the path where the payload will be dropped.

Following execution, the dropper modifies the last access time of the deployed files to match the timestamps of the legitimate file “EdgeTransport.exe”. It is advised that forensic investigators conduct a search for files that have this timestamp applied.

Finally, the dropper executes the following command to remove all installation log files:

```
cmd.exe /c del *.InstallLog *.InstallState
```

Persistence

In order to persist on the compromised hosts, Neuron service installs itself as an automatic service, allowing the infection to persist through a server restart. The Neuron service can be manually stopped and removed, and contains no method of re-establishing execution.

The Neuron service attempts to masquerade as legitimate Microsoft Exchange or Microsoft IIS services. A list of the service names and descriptions used within Neuron samples is as follows:

SERVICE NAME	DISPLAY NAME	DESCRIPTION
MSExchangeService	Microsoft Exchange Service	Host service for the Microsoft Exchange Server management provider. If this service is stopped or disabled, Microsoft Exchange cannot be managed.
W3WPDIAG	Microsoft IIS Diagnostics Service	Host service for the Microsoft IIS management provider. If this service is stopped or disabled, Microsoft IIS cannot be managed.
Updater	Updater	Host service for software update. If this service is stopped or disabled, software cannot be update.

Network Communications

Communications between the client and service are via HTTP requests. The service will establish a HTTP listener, commonly on port 443 (https), however instances have been analysed where port 80 (http) is used instead. The listener waits for requests on the host matching specific URIs defined by the configuration. The following have been defined in the configuration in Neuron samples analysed by the NCSC:

```
https://*:443/ews/exchange/

https://*:443/W3SVC/

https://*:80/W3SVC/
```

Neuron clients send requests to the defined endpoint in order to communicate with the service. In order to make the traffic from clients look legitimate, the actor has chosen to name their endpoints with common Microsoft Windows terms.

Communications are encrypted using RC4 as an additional layer of security. The RC4 key is sent to the connecting client using a pre-configured RSA key.

Parameters for a request are sent in the POST body, with the following values possible:

```
cid

cadataKey

cadata

cadataSig
```

The values for these parameters are base64 encoded and RC4 encrypted using the key exchanged between the client and service. Each parameter performs a different task within the service; for example, “cid” requests the current RC4 key and “cadata” sends an instruction to be run.

An example HTTP communication is shown below:

```
POST https://<domain>/ews/exchange/exchange.asmx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: <domain>
Content-Length: <variable>
Expect: 100-continue
Connection: Keep-Alive

cadata=<url_encoded_b64>
```

The following SNORT rules can be used to alert on this traffic. Network collection will need to be in place between the client and server; in most instances, this is between two machines within the same LAN:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (flow: established,from_client; msg: "Web/request\:POST - Neuron A"; content: "cadata="; fast_pattern; content: "Content-Type|3a| application/x-www-form-urlencoded"; content: "Expect|3a| 100-continue"; pcre: "/\ncadata=[a-zA-Z0-9%]{1,5000}/"; content: "POST"; http_method; rev: 1; priority: 1;)


alert tcp $HOME_NET any -> $EXTERNAL_NET any (flow: established,from_client; msg: "Web/request\:POST - Neuron B"; content: "cadata="; fast_pattern; content: "Content-Type|3a| application/x-www-form-urlencoded"; content: "Expect|3a| 100-continue"; pcre: "/\ncadataKey=[a-zA-Z0-9%]{1,5000}/"; content: "POST"; http_method; rev: 1; priority: 1;)


alert tcp $HOME_NET any -> $EXTERNAL_NET any (flow: established,from_client; msg: "Web/request\:POST - Neuron C"; content: "cadata="; fast_pattern; content: "Content-Type|3a| application/x-www-form-urlencoded"; content: "Expect|3a| 100-continue"; pcre: "/\ncid=[a-zA-Z0-9%]{1,5000}/"; content: "POST"; http_method; rev: 1; priority: 1;)


alert tcp $HOME_NET any -> $EXTERNAL_NET any (flow: established,from_client; msg: "Web/request\:POST - Neuron D"; content: "cadata="; fast_pattern; content: "Content-Type|3a| application/x-www-form-urlencoded"; content: "Expect|3a| 100-continue"; pcre: "/\ncadataSig=[a-zA-Z0-9%]{1,5000}/"; content: "POST"; http_method; rev: 1; priority: 1;)
```

In addition to HTTP communications, some observed Neuron service samples have functionality that enables the clients to communicate with it via pipes, for example:

```
pipe://*/Winsock2/w3svc
```

Capabilities

The main functionality of the Neuron service is to return and synchronise StorageFile and StorageScript objects between the client and service.

A StorageFile object contains information about a file including its name, modified date and the file contents; a StorageScript object contains “instructions”. There are multiple instruction types, including the following:

- Executing a command using cmd.exe
- Creating new StorageFiles
- Downloading specified or all StorageFiles

Neuron Client

The Neuron client component is typically installed on endpoint machines within a victim network. Command & Control (C2) is conducted by the Neuron service. The client is designed to collect, package and send documents to the service component for onward exfiltration.

Associated Files

Name	neuron-client.exe
Description	Neuron Client
MD5	4ed42233962a89deaa89fd7b989db081
SHA1	cf731ee0af5c19231ff51af589f7434c0367d508
SHA256	a96c57c35df18ac20d83b08a88e502071bd0033add0914b951adbd1639b0b873
Size	55808

Name	Sign.exe
Description	Dropper for the Neuron client
MD5	3cd5fa46507657f723719b7809d2d1f9
SHA1	34ddc14b9a04eba98c3aa1cb27033e12ec847e03
SHA256	a6dbc36c472b3ba70a98efd0db35e75c340086be15d3c3ab4e39033604d0bcf9
Size	115712

Name	mydoc.doc
Description	Macro document that drops and runs Sign.exe (client dropper)
MD5	66f4f1384105ce7ee1636d34f2afb1c9
SHA1	3f23d152cc7badf728dfd60f6baa5c861a500630
SHA256	42fbb2437faf68bae5c5877bed4d257e14788ff81f670926e1d4bbe731e7981b
Size	591360

Name	N/A
Description	Macro document that drops and runs Sign.exe (client dropper)
MD5	0e430b6b203099f9c305681e1dcff375
SHA1	845f3048fb0cfbdfb35bf6ced47da1d91ff2e2b1
SHA256	bbe3700b5066d524dd961bd47e193ab2c34565577ce91e6d28bdaf609d2d97a8
Size	590336

Infection Vector and Install

The Neuron client infection vector appears to be via spear-phishing victims with documents containing macros. When a document is opened, and macros are enabled, a base64 encoded blob is constructed and written to the %temp% directory as “Signature.crt”; this is then decoded using the legitimate Microsoft binary “certutil.exe”, for example:

```
certutil.exe -decode %TEMP%\Signature.crt %TEMP%\Sign.exe
```

The resulting executable is the Neuron client dropper, which is responsible for setting up any initial configuration, establishing persistence and dropping the main payload to disk.

The main payloads are embedded in the dropper executable and are GZIP compressed and RC4 encrypted with a hardcoded key. The dropper is also responsible for deploying any legitimate DLLs that may also be required – these are stored in the same way.

All files are placed into the directory from which the dropper was executed.

Persistence

The Neuron client executable contains no functionality to establish persistence. Instead, the dropper handles this for the client by creating a scheduled task, enabling it to persist after a reboot.

The task is scheduled to run every 12 minutes (PT12M), with a task ID of “Microsoft Corporation” and a task description constructed from a string retrieved from a randomly selected registry value. To build the task description, a list of value names of length 9 or greater but not containing "\" are retrieved from HKLM\\Software\\Microsoft registry. One of these values is selected and prefixed to the string " updater". This is then used as the description for the scheduled task.

Configuration

The Neuron client configuration is stored in the registry as JSON; it must be set up by the dropper before the client is run as no defaults are specified.

The configuration includes the domains where Neuron service implants have been deployed, so that the client can communicate with them. The configuration also specifies a beacon interval for each domain, along with a keep alive interval and time wait interval.

An example of the server configuration in JSON representation, taken from a Neuron client dropper, is as follows:

```
“Connect”: [
  {
    “URL”: “https://<removed>/ews/exchange/exchange.asmx”,
    “Interval”: 17
  },
    “URL”: “https://<removed>//ews/exchange/exchange.asmx”,
    “Interval”: 32
  {

  }

],

“KeepAliveInterval”: 7,

“CmdTimeWait”: 5
```

Network Communications

Communications are detailed in the Neuron service section. The Neuron client and service primarily communicate via HTTP requests.

As an extra layer of security, the client RC4 encrypts any data being sent. The key used is the Machine GUID retrieved from the registry (SOFTWARE\Microsoft\Cryptography\MachineGuid); if this is not set then the default key “8d963325-01b8-4671-8e82-d0904275ab06” is used.

Capability

Once loaded the Neuron Client will loop indefinitely, performing a sync of storage files with the Neuron service. The interval between synchronisations is specified in the configuration by the “CmdTimeWait” value.

In order to synchronise with the service, the client will retrieve all local StorageFile objects and all StorageFiles on the service (without file data) and compare these for differences. The client retrieves the StorageFiles from the service by sending a POST request with the following data within the parameter “cadata”:

```
{
  “cmd”: 0,
  “data”: “”
}
```

This is encrypted with RC4 and then base64 encoded before being sent.

The service will respond with a list of all StorageFile metadata (i.e. name and date of each StorageFile). This is then used to determine which StorageFiles the client is missing, as well as any files which the service is missing.

The client will send any required files (including file data) to the service by sending the following command data:

```
{
  “cmd”: 1,
  “data”: [
    <list_of_storage_files>
  ]
}
```

Where a storage file object has a JSON representation as follows:

```
{
  “name”: name,
  “data”: data,
  “date”: date
}
```

Finally, the client will download all missing StorageFiles from the service by sending the following command data:

```
{
  “cmd”: 2,
  “data”: <array_of_request_storage_files>
}
```

Where the sent data contains the required StorageFile names, as follows:

```
[
  {
    “name”: “storage.file.1”
  },
  {
    “name”: “storage.file.2”
  }
]
```

These new files are then written to disk, and added to the clients list of StorageFiles.

Nautilus

Nautilus is very similar to Neuron both in the targeting of mail servers and how client communications are performed. This malware is referred to as Nautilus due to its embedded internal DLL name "nautilus-service.dll", again sharing some resemblance to Neuron.

The main payload and configuration of Nautilus is encrypted within a covert store on disk which is located in *"\ProgramData\Microsoft\Windows\Caches\"*. The loader DLL will access this covert store to decrypt the payload (oxygen.dll), which is then loaded into a target process via reflective loading.

The Nautilus service listens for HTTP requests from clients to process tasking requests such as executing commands, deleting files and writing files to disk.

Associated Files

Name	dcomnetsrv.dll
Description	Nautilus Loader DLL
MD5	2f742ec3bb7590602bc3e97326f2476a
SHA1	9d280e3ef1b180449086dda5b92a7b9bbe63dee4
SHA256	a415ab193f6cd832a0de4fcc48d5f53d6f0b06d5e13b3c359878c6c31f3e7ec3
Size	121344

Name	oxygen.dll
Description	Nautilus Injected payload
MD5	ea874ac436223b30743fc9979eed5f2f
SHA1	5ed61ec7de11922582f07c3488ef943b439ee226
SHA256	cefc5cf4d46abb86fb0f7c81549777cf1a2a5bfbe1ce9e7d08128ab8bfc978f8
Size	620568

Persistence

Nautilus achieves persistence by running as a service, dcomnetsrv, which is set to automatically start. It is very likely that this is established by the Nautilus dropper, similar to the Neuron service dropper; however, the NCSC has not yet analysed a sample of this file.

Configuration

The configuration for Nautilus is stored encrypted within a covert store that was located in *"\ProgramData\Microsoft\Windows\Caches\"*.

The server configuration block, which defines the port and URL for Nautilus to listen on, is passed in the identifier *“config_listen.system”*. A sample configuration is shown below:

```
proto=https
host=+
port=443
param=OWA-AUTODISCOVER-EWS
```

Nautilus also stores several other pieces of contextual information within the covert store under the identifier *“ctx.system”*, including an RSA public key:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAg4r6SSnj2PnYbe6C4H8c
M7162eRS+RTE8BYW8cTGdFPSiDiVOblImyddBLu/fW7MSc+BUsmg2l9SVyvJrHJk
0xnr7PRH9Dq7IcTYzQPMSsG1nC2Lej09EtilKwAQP08MIpiredzgXwom3rlH0Trc
HiKxjLhQcuK0Mllsq+54gYPaoi6LkZG/lUxhWuGI1M2i3/dHp40vbwaaL5Sotxuv
jSytDsU75U5T+rCAHVMykiLi/x7PKg40JQoYGMSOPUJsx87i/uy3uHoecl2ns038
b70Gh6KJ4x5mwaKjMRsSm8PUN6ccHSyqetpXuTXoKU5dEDIQLNAwXTZY40d/aTEx
uQIDAQAB
-----END PUBLIC KEY-----
```

The covert store uses a proprietary format to store data. This format stores separate streams (i.e. one for the config and one for the context) with each split into chunks of 4096 bytes and encrypted using RC4. The offset to the next chunk is calculated by taking the decrypted int value at offset 0xFF8 of the decrypted chunk, shifting this left by 0xC and then adding 0x10000. For the first chunk, this initial int value is at offset 0xB4 of the header.

A default RC4 key is used to decrypt the first chunk; this key is hardcoded into Nautilus as *“1B1440D90FC9BCB46A9AC96438FEEA8B”* but is passed into a function that trims the length to 31 bytes, resulting in the final 32 byte initial RC4 key being *“1B1440D90FC9BCB46A9AC96438FEEA8\x00”*.

The RC4 implementation used for encryption of the covert store has been modified from a standard implementation. This may be an attempt to frustrate decryption; however, it is easily spotted when reverse engineering the sample.

The following Python implementation duplicates the modified RC4 XOR loop:

```
def rc4(data, key):
       x = 0
       box = range(256)
       for i in range(256):
              x=(x + box[i] + ord(key[i%len(key)])) % 256
              box[i], box[x] = box[x], box[i]
       out = []
       key = []
       i = box[1]
       j = box[i]
       box[i] = i
       box[1] = j
       for char in data:
              sbb = box[i % 256]
              i += 1
              sbb += j
              kb = box[sbb % 256]
              out.append(chr(ord(char) ^ kb))
       return ''.join(out)
```

A covert store can be identified by RC4 decrypting the 4 bytes at offset 0xFFFC with the default RC4 key followed by comparison with the magic bytes 0x3a29bd32.

Communications

Communication with clients is performed in a similar fashion to Neuron. Nautilus listens for incoming connections from clients on port 443 that are addressed to the URL “/OWA-AUTODISCOVER-EWS”; this URL path could be modified. Nautilus is commonly installed on a victim mail server, enabling the pre-installed TLS configuration to be used.

Data sent to the service is encoded in the referrer header, which is masquerading as a legitimate Bing search. The format string used to create this is as follows:

```
Referer: http://www.bing.com/search?q=%s&go=Submit&qs=n&pq=%s&sc=0-11&sp=-
1&sk=&cvid=%s&first=21&FORM=%s
```

Capability

The malware can take commands from connecting clients to perform on the infected host. The commands take the format "O_001", "O_002" and so on. A subset of these commands allow Nautilus to be tasked with the following:

- O_001: Execute a cmd.exe command
- O_002: Read file
- O_003: Write file
- O_007: Delete file
- O_008: GetTempPathA
- O_009: Sleep
- O_010: Create directory
- O_011: Check if directory
- O_012 Duplicate of O_011

There also appear to be some separately processed commands containing the following functionality:

- O_100 Shutdown (implant)
- O_101 Uninstall

ErrorFE.aspx

Alongside the Neuron and Nautilus toolkits, the NCSC identified a file named “errorFE.aspx”. This file was installed on a number of victims following the successful exploitation of web application software, and provides additional persistence to enable the deployment of further tools.

The script defines its working directory as the value of the Windows environment variable “temp”, using this location to drop and execute files and collect data.

This script accepts web requests and extracts the cookie parameter; valid data in the cookie is base64 encoded and AES encrypted using hardcoded values.

The script supports the processing of multiple cookies from a single request, indicating it is possible to issue multiple commands in a single request. When the cookie value is decoded and decrypted, the script expects one of the following commands followed by any additional parameters:

Command	**Function**
put	Accepts a file name and writes the contents of “data” request parameter to a file in the working directory
update	Overwrites the shell itself with the content of the “data” request parameter
time	Updates the timestamp on a specific file with a specified timestamp (creation, last write and access).
cmd	Executes a provided command using “cmd.exe”
del	Deletes a specified file
get	Gets a specified filename from the working directory and returns its contents to the requestor

Appendix A

Neuron Client

File Name	neuron-client.exe
Description	Neuron Client
File Size (bytes)	55808
MD5	4ed42233962a89deaa89fd7b989db081
SHA1	cf731ee0af5c19231ff51af589f7434c0367d508
SHA256	a96c57c35df18ac20d83b08a88e502071bd0033add0914b951adbd1639b0b873

File Name	Sign.exe
Description	Dropper for the Neuron Client
File Size (bytes)	115712
MD5	3cd5fa46507657f723719b7809d2d1f9
SHA1	34ddc14b9a04eba98c3aa1cb27033e12ec847e03
SHA256	a6dbc36c472b3ba70a98efd0db35e75c340086be15d3c3ab4e39033604d0bcf9

File Name	mydoc.doc
Description	Macro document that drops and runs Sign.exe (client dropper)
File Size (bytes)	591360
MD5	66f4f1384105ce7ee1636d34f2afb1c9
SHA1	3f23d152cc7badf728dfd60f6baa5c861a500630
SHA256	42fbb2437faf68bae5c5877bed4d257e14788ff81f670926e1d4bbe731e7981b

File Name	N/A
Description	Macro document that drops and runs Sign.exe (client dropper)
File Size (bytes)	590336
MD5	0e430b6b203099f9c305681e1dcff375
SHA1	845f3048fb0cfbdfb35bf6ced47da1d91ff2e2b1
SHA256	bbe3700b5066d524dd961bd47e193ab2c34565577ce91e6d28bdaf609d2d97a8

Neuron Service

File Name	Microsoft.Exchange.Service.exe
Description	Neuron Service
File Size (bytes)	43008
MD5	0f12268221e27406351a6313f902b498
SHA1	b0dbdc81a0e367330007b7e593d8dabf92ca7afd
SHA256	d1d7a96fcadc137e80ad866c838502713db9cdfe59939342b8e3beacf9c7fe29

File Name	w3wpdiag.exe
Description	Neuron Service
File Size (bytes)	59392
MD5	371b4380080e3d94ffcae1a7e9a0d5e2
SHA1	f7088075d1c798f27b0d269c97dc877ff16f1401
SHA256	2986bae15cfa78b919d21dc070be944e949a027e8047a812026e35c66ab17353

File Name	Updater.exe
Description	Neuron Service
File Size (bytes)	44544
MD5	8229622a9790d75e09a099e8758d5703
SHA1	10586913ceeecd408da4e656c29ed4e91c6b758e
SHA256	2f4d6a3c87770c7d42d1a1b71ed021a083b08f69ccaf63c15428c7bc6f69cb10

File Name	w3wpdiag.exe
Description	Neuron Service
File Size (bytes)	58880
MD5	a3bdc385cf68019449027bd6d8cecb4d
SHA1	fe8da5a1e62a8d4f627834b0f26c802a330d8d45
SHA256	0f4e9e391696ed8b9172985bb43cca7d7f2c8a4ae0493e4bf1f15b90f7138259

File Name	dropper-svc.exe
Description	Dropper for the Neuron service
File Size (bytes)	85008
MD5	d6ef3c8f2c3f3ddffbb70f5dadfa982c
SHA1	934b288075c122165897276b360c61e77cb7bde0
SHA256	fa543de359d498150cbcb67c1631e726a4b14b0a859573185cede5b12ad2abfb

Neuron Yara

```
rule neuron_common_strings {

  meta:

    description = "Rule for detection of Neuron based on commonly used strings"

    author = "NCSC UK"

    hash = "d1d7a96fcadc137e80ad866c838502713db9cdfe59939342b8e3beacf9c7fe29"

  strings:

    $strServiceName = "MSExchangeService" ascii

    $strReqParameter_1 = "cadataKey" wide

    $strReqParameter_2 = "cid" wide

    $strReqParameter_3 = "cadata" wide

    $strReqParameter_4 = "cadataSig" wide

    $strEmbeddedKey =
"PFJTQUtleVZhbHVlPjxNb2R1bHVzPnZ3WXRKcnNRZjVTcCtWVG9Rb2xuaEVkMHVwWDFrVElFTUNTNEFnRkRCclNm
clpKS0owN3BYYjh2b2FxdUtseXF2RzBJcHV0YXhDMVRYazRoeFNrdEpzbHljU3RFaHBUc1l4OVBEcURabVVZVklVb
HlwSFN1K3ljWUJWVFdubTZmN0JTNW1pYnM0UWhMZElRbnl1ajFMQyt6TUhwZ0xmdEc2b1d5b0hyd1ZNaz08L01vZH
VsdXM+PEV4cG9uZW50PkFRQUI8L0V4cG9uZW50PjwvUlNBS2V5VmFsdWU+" wide

    $strDefaultKey = "8d963325-01b8-4671-8e82-d0904275ab06" wide

    $strIdentifier = "MSXEWS" wide

    $strListenEndpoint = "443/ews/exchange/" wide

    $strB64RegKeySubstring = "U09GVFdBUkVcTWljcm9zb2Z0XENyeXB0b2dyYXBo" wide

    $strName = "neuron_service" ascii

    $dotnetMagic = "BSJB" ascii


  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and $dotnetMagic and 6 of
($str*)

}
```

```
rule neuron_standalone_signature {

  meta:

    description = "Rule for detection of Neuron based on a standalone signature from .NET
metadata"

    author = "NCSC UK"

    hash = "d1d7a96fcadc137e80ad866c838502713db9cdfe59939342b8e3beacf9c7fe29"

  strings:

    $a =
{eb073d151231011234080e12818d1d051281311d1281211d1281211d128121081d1281211d1281211d128121
1d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211
d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281211d1281}

    $dotnetMagic = "BSJB" ascii

  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and all of them

}
```

```
rule neuron_functions_classes_and_vars {

  meta:

    description = "Rule for detection of Neuron based on .NET function, variable and
class names"

    author = "NCSC UK"

    hash = "d1d7a96fcadc137e80ad866c838502713db9cdfe59939342b8e3beacf9c7fe29"

  strings:

    $class1 = "StorageUtils" ascii

    $class2 = "WebServer" ascii

    $class3 = "StorageFile" ascii

    $class4 = "StorageScript" ascii

    $class5 = "ServerConfig" ascii

    $class6 = "CommandScript" ascii

    $class7 = "MSExchangeService" ascii

    $class8 = "W3WPDIAG" ascii

    $func1 = "AddConfigAsString" ascii

    $func2 = "DelConfigAsString" ascii

    $func3 = "GetConfigAsString" ascii

    $func4 = "EncryptScript" ascii

    $func5 = "ExecCMD" ascii

    $func6 = "KillOldThread" ascii

    $func7 = "FindSPath" ascii

    $var1 = "CommandTimeWait" ascii

    $dotnetMagic = "BSJB" ascii

  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and $dotnetMagic and 6 of
them

}
```

Nautilus

File Name	dcomnetsrv.dll
Description	Nautilus Loader DLL
File Size (bytes)	121344
MD5	2f742ec3bb7590602bc3e97326f2476a
SHA1	9d280e3ef1b180449086dda5b92a7b9bbe63dee4
SHA256	a415ab193f6cd832a0de4fcc48d5f53d6f0b06d5e13b3c359878c6c31f3e7ec3

File Name	oxygen.dll
Description	Nautilus Injected payload
File Size (bytes)	620568
MD5	ea874ac436223b30743fc9979eed5f2f
SHA1	5ed61ec7de11922582f07c3488ef943b439ee226
SHA256	cefc5cf4d46abb86fb0f7c81549777cf1a2a5bfbe1ce9e7d08128ab8bfc978f8

Nautilus Yara

```
rule nautilus_modified_rc4_loop {

  meta:

    description = "Rule for detection of Nautilus based on assembly code for a modified
RC4 loop"

    author = "NCSC UK"

    hash = "a415ab193f6cd832a0de4fcc48d5f53d6f0b06d5e13b3c359878c6c31f3e7ec3"

  strings:

    $a = {42 0F B6 14 04 41 FF C0 03 D7 0F B6 CA 8A 14 0C 43 32 14 13 41 88 12 49 FF C2
49 FF C9}

  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and $a

}
```

```
rule nautilus_rc4_key {

  meta:

    description = "Rule for detection of Nautilus based on a hardcoded RC4 key"

    author = "NCSC UK"

    hash = "a415ab193f6cd832a0de4fcc48d5f53d6f0b06d5e13b3c359878c6c31f3e7ec3"

  strings:

    $key = {31 42 31 34 34 30 44 39 30 46 43 39 42 43 42 34 36 41 39 41 43 39 36 34 33 38
46 45 45 41 38 42}

  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and $key

}
```

```
rule nautilus_common_strings {

  meta:

    description = "Rule for detection of Nautilus based on common plaintext strings"

    author = "NCSC UK"

    hash = "a415ab193f6cd832a0de4fcc48d5f53d6f0b06d5e13b3c359878c6c31f3e7ec3"

  strings:

    $ = "nautilus-service.dll" ascii

    $ = "oxygen.dll" ascii

    $ = "config_listen.system" ascii

    $ = "ctx.system" ascii

    $ = "3FDA3998-BEF5-426D-82D8-1A71F29ADDC3" ascii

    $ = "C:\\ProgramData\\Microsoft\\Windows\\Caches\\{%s}.2.ver0x0000000000000001.db"
ascii

  condition:

    (uint16(0) == 0x5A4D and uint16(uint32(0x3c)) == 0x4550) and 3 of them

}
```

Additional Indicators for Forensic Analysis

The following indicators can be used to search for the presence of Neuron and Nautilus malware within forensic analysis tools.

```
$zf(-1,
$zf(-2,
{"instructions":[{"type":
App_Web_juvjerf3.dll
App_Web_vcplrg8q.dll
ar_all2.txt
ar_sa.txt
Convert.FromBase64String(temp[1])
D68gq#5p0(3Ndsk!
dx11.exe
ERRORF~1.ASP
errorFE.aspx
errorfe.aspx.f5dba9b9.compiled
intelliAdminRpc
J8fs4F4rnP7nFl#f
lsa.exe
Msnb.exe
msrpc.exe
Neuron_service
owa.exe
owa_ar2.bat
rexec.exe
payload.x64.dll.system
service.x64.dll.system
```