Threat Spotlight: Ratsnif - New Network Vermin from OceanLotus

threatvector.cylance.com/en_us/home/threat-spotlight-ratsnif-new-network-vermin-from-oceanlotus.html

The BlackBerry Cylance Threat Research Team

RESEARCH & INTELLIGENCE / 07.01.19 / The BlackBerry Cylance Threat Research Team

Overview

The OceanLotus Group (aka APT32, CobaltKitty | previous reports: The SpyRATs of OceanLotus; OceanLotus APT Group Leveraging Steganography) is using a suite of remote access trojans dubbed "Ratsnif" to leverage new network attack capabilities. Blackberry Cylance threat researchers have analyzed the Ratsnif trojans, which offer a veritable swiss-army knife of network attack techniques. The trojans, under active development since 2016, combine capabilities like packet sniffing, gateway/device ARP poisoning, DNS poisoning, HTTP injection, and MAC spoofing.

We delved into four distinct Ratsnif samples, three of them developed in 2016, the fourth created during the latter half of 2018.

Sample 1

MD5	516ad28f8fa161f086be7ca122351edf
SHA256	b4e3b2a1f1e343d14af8d812d4a29440940b99aaf145b5699dfe277b5bfb8405
Filename	javaw.exe, Client.exe
Path	X:\Project\BotFrame\Debug\Client.exe
Size	1.32 MB (1,387,520 bytes)
File Type	PE32 executable for MS Windows (console) Intel 80386 32-bit

Alias	OceanLotus APT32 Ratsnif
Compile Time	2016-08-05 07:57:13

Overview

The earliest example of Ratsnif uncovered thus far was compiled on the same day that its C2 domain was first activated:

It appears to be a debug build, and closely resembles a later variant from September 2016 that will be the main focus of analysis for the three 2016 variants described in this article.

Sample 2

MD5	b2f8c9ce955d4155d466fbbb7836e08b
SHA256	b214c7a127cb669a523791806353da5c5c04832f123a0a6df118642eee1632a3
Filename	javaw.exe, Client.exe
Path	X:\Project\BotFrame\Debug\Client.exe
Size	1.32 MB (1,387,520 bytes)
File type	PE32 executable for MS Windows (console) Intel 80386 32-bit
Alias	OceanLotus APT32 Ratsnif
Compile Time	2016-08-06 04:30:06

Overview

Compiled less than 24 hours after the previous sample, this build contains only one minor difference in functionality, whereby a call to *pcap_dump_flush()* has been removed prior to recompilation:

Figure 1. Call to pcap_dump_flush in b4e3b2a1f1e343d14af8d812d4a29440940b99aaf145b5699dfe277b5bfb8405

Figure 2. Missing call to pcap_dump_flush in b214c7a127cb669a523791806353da5c5c04832f123a0a6df118642eee1632a3

In addition, the CodeView debugging information has changed, reflecting the new "age" of the sample after recompilation:


Figure 3. Age of 0x14 in b4e3b2a1f1e343d14af8d812d4a29440940b99aaf145b5699dfe277b5bfb8405


Figure 4. Age of 0x15 in b214c7a127cb669a523791806353da5c5c04832f123a0a6df118642eee1632a3

Both samples were submitted to VirusTotal within a minute of being compiled and contain the same path as the PDB information. It seems likely this sample was automatically submitted to an online scanning service by the developer:


Figure 5. VirusTotal submission showing date/time and path

Sample 3

MD5	7f0ac1b4e169edc62856731953dad126
SHA256	b20327c03703ebad191c0ba025a3f26494ff12c5908749e33e71589ae1e1f6b3
Filename	javaw.exe, adobe.exe
Path	N/A
Size	432 KB (442,880 bytes)
File Type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Alias	OceanLotus APT32 Ratsnif
Compile Time	2016-09-13 09:26:42

Overview

Remarkably similar in functionality to the previous samples from August 2016, this sample is a release build and was likely one of the earlier Ratsnifs to be deployed by OceanLotus in-the-wild.

Threat Features

- C2 over HTTP
- Packet sniffing
- ARP poisoning
- DNS spoofing
- HTTP redirection
- Remote shell

Analysis

Upon execution, Ratsnif creates a run once mutex named "onceinstance", initialises Winsock version 2.2, and harvests system information such as the username, computer name, workstation configuration (via NetWkstaGetInfo API), Windows system directory and network adapter information. This information will then be sent to the attacker's C2 server via an HTTP post to the /*cl_client_online.php* API endpoint. Next, a logging thread is created, which is used to route log messages to the C2 via HTTP POST requests to */cl_client_logs.php.* The malware then proceeds to load wpcap.dll, before importing the following functions:

- pcap_sendqueue_transmit
- pcap_findalldevs
- pcap_freealldevs
- pcap_open_live
- pcap_sendqueue_alloc
- pcap_next_ex
- pcap_sendqueue_queue
- pcap_sendpacket
- pcap_close
- pcap_sendqueue_destroy
- pcap_dump_open
- pcap_dump_ftell
- pcap_dump_flush
- pcap_dump_close
- pcap_dump

With WinPcap successfully loaded, a further HTTP POST request is made to */cl_client_cmd.php,* which is used to obtain a command code from the attacker. This code will check for commands every 10 seconds. C2 commands are decrypted using AES with a hard-coded static key via Windows APIs, before being dispatched by a simple command processor.

C2

All observed Ratsnif samples have been hardcoded with one or more C2 domains, regardless of whether they are used. This sample contains 2 hard-coded domains, although only one appears to have ever been active:

- search[.]webstie[.]net
- dns[.]domain-resolve[.]org (inactive)

The C2 server itself is expected to expose a fairly intuitively named web API, supporting the following endpoints:

URL	Description
/cl_client_online.php	POST containing harvested system information
/cl_client_cmd.php	GET C2 command
/cl_client_cmd_res.php	POST result of C2 command
/cl_client_logs.php	POST log message

The malware contains support for the following commands issued via the *cl_client_cmd.php* HTTP response:

Sample 4

MD5	88eae0d31a6c38cfb615dd75918b47b1
SHA256	7fd526e1a190c10c060bac21de17d2c90eb2985633c9ab74020a2b78acd8a4c8
Filename	N/A
Path	N/A
Size	745 KB (762,880 bytes)
File Type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Alias	OceanLotus APT32 Ratsnif
Compile Time	Wed, 08 Aug 2018 02:52:52 UTC

Overview

Surfacing during the latter half of 2018 and wrapped in a bespoke OceanLotus shellcode loader, this sample was first reported in a blog from Macnica Networks. Compared to the 2016 variants this sample introduces a configuration file and does not rely on C2 for operation. It also adds new features in the form of HTTP injection, protocol parsing, and SSL hijacking.

Threat Features

- Deployed by OceanLotus loader
- Use of separately supplied configuration file, tailored to the victim's network environment (as opposed to backdoor commands in the previous versions)
- Use of separately supplied SSL certificates to perform SSL hijacking
- Use of WolfSSL library (version 3.11) for decryption of SSL traffic (https://github.com/wolfSSL/wolfssl)
- Use of http_parser.c for parsing HTTP traffic (https://elixir.bootlin.com/zephyr/v1.13.0/source/subsys/net/lib/http/http_parser.c)
- Packet sniffing focused on extracting login credentials and other sensitive data via protocol parsing
- ARP poisoning
- DNS spoofing
- HTTP redirection
- HTTP injection

Analysis

For this particular sample, the actual sniffer executable is Base64 encoded within a loader DLL and wrapped in two layers of shellcode. The loader DLL decodes the payload, copies it to memory and executes the 1st stage shellcode, which will decompress the binary and execute the 2nd stage shellcode in a separate thread. The 2nd stage shellcode will inject the sniffer executable into memory and hook several API functions responsible for

returning the process command line (GetCommandLineA, GetCommandLineW, _acmdln, _wcmdln), so they return a hardcoded string instead. The string contains the parameter that specifies a path to the config file, as well as the executable's original path:

```
C:\Users\Administrator\Desktop\api\temp\royal\HkYh9CvH7.exe -p
C:\ProgramData\setting.cfg
```

Figure 6. Embedded command-line

It is not immediately obvious why the attackers used this convoluted method to pass the config path to the malware.

The configuration file is a simple text file, Base64 encoded, where the first line is ignored, and each subsequent line specifies a parameter. For example:

```
[unused_line]
-ip [ATTACKER IP ADDRESS]
-ga [DEFAULT GATEWAY]
-subnet [SUBNET MASK]
-sniff -ssl_ip [IP ADDRESS]
-html_inject [BROWSER PROCESS NAME]
-dlog_ip [IP ADDRESS]
-mac [ATTACKER MAC ADDRESS] "true"|"false"
-name [DOMAIN NAME] [REDIRECTION IP]
-all
-dnsttl [INT VALUE]
-log [LOGFILE PATH]
-pass [CREDENTIALS DUMP PATH]
-dwn_ip [IP ADDRESS]
```

Figure 7. Configuration file options

However, there is a bug in parsing the value of the *dwn_ip* parameter, which will result in a memory read violation if the value is present in the configuration:

Figure 8: Bug in the code: the value of "dwn_ip" is passed as a string, while print_debug_msg expects a pointer to a string

Once executed, the sniffer will read the configuration from the specified file, decode it using Base64 and parse it to an in-memory structure. If the "-sniff" parameter is specified in the configuration, the malware will add a firewall exception and disable Large Send Offload (LSO) for each network adapter in the registry:

```
netsh advfirewall firewall add rule name=\"Core Networking - Router Solicitation\" dir=in action=allow
program={self_path} enable=yes
```

Figure 9. Command-line used to add Windows firewall rule

```
wmic path win32_networkadapter where index=%d call disable
```

Figure 10. Command-line used to disable network adapters prior to disabling LSOs

After importing the same APIs from wpcap.dll as the 2016 variants (with the addition of pcap_geterr), the malware creates threads responsible for ARP poisoning and DNS spoofing.

In order to be able to decrypt the SSL traffic, the malware performs SSL hijacking, using an open source library called WolfSSL and separately supplied certificate and private key files. For that purpose, it creates an internal WolfSSL server, listening on the first available port in the range 65000 – 65535:


Figure 11: Use of WolfSSL

Unlike the 2016 variants of Ratsnif that stored all packets to a PCAP file, the 2018 variant employs multiple sniffer classes for harvesting sensitive information from packets. This will minimize the amount of data the attacker has to collect, exfiltrate and process, and also reveals what information the attacker is interested in.

The malware can sniff traffic for the following protocols/ports:

Interface	Ports	Headers
CSniffFtp	21, 990	ABOR ACCT ADAT ALLO APPE AUTH CCC CDUP CONF CWD DELE ENC EPRT EPSV FEAT HELP HOST LANG LIST LPRT LPSV MDTM MIC MKD MLSD MLST MODE NLST OPTS PASS PASV PBSZ PORT PROT PWD QUIT REIN REST RETR RMD RNFR RNTO SITE SIZE SMNT STAT STOR STOU STRU SYST TYPE USER XCUP XMKD XPWD XRCP XRMD XRSQ XSEM XSEN 230
CSniffImap	143, 993	CAPABILITY LOGOUT STARTTLS AUTHENTICATE LOGIN SELECT EXAMINE CREATE RENAME LSUB STATUS APPEND CHECK CLOSE EXPUNGE FETCH STORE UID
CSniffLdap	389, 636, 10389, 10636	Various
CSniffNntp	119	AUTHINFO USER AUTHINFO PASS ANONYMOUS 281
CSniffPop	110, 995	RCEV RCVD RSET +OK USER PASS RETR QUIT
CSniffSmb	445	Various
CSniffSmtp	25, 465	HELO MAIL RCPT SEND SOML SAML VRFY EXPN TURN FROM
CSniffTds	1433	SELECT name, password_hash FROM master.sys.sql_logins where is_disabled = 0; -- priv
CSniffTelnet	23	Login Failed login: password:
SniffHttp2	80, 443	Various

Each sniffer class interface contains two methods for extracting sensitive information from the incoming and outgoing packets, respectively. These typically rely on searching for cleartext header strings to facilitate credential theft:


Figure 12. Searching for login and password commands in the Telnet protocol

In addition, the HTTP sniffer interface is also able to perform injection to insert arbitrary attacker supplied content into HTML.

C2

Although this sample contains a Base64 encoded C2 URL hardcoded in the .rdata section (the same address as in the 2016 versions), the malware never seems to use it; instead, it logs the captured information into text files for further exfiltration by another module.

Example

To recreate conditions in which the sample would operate, a default gateway was configured on 192.168.8.135 and was running iNetSim to act as the DNS and HTTP servers. The attacker machine was located at 192.168.8.134 and the victim at 192.168.8.138. Ratsnif was configured to operate as follows:

```
TEST CONFIG
-ip "192.168.8.134"
-ga "192.168.8.135"
-subnet "255.255.255.0"
-sniff
-ssl_ip "192.168.8.254"
-html_inject "iexplore.exe"
-dlog_ip "192.168.8.254"
-mac "00:0C:29:59:62:46" "true"
-name "www.google.com" "192.168.8.135"
-dnsttl "100"
-log "C:\ratsnif.log"
-pass "C:\ratsnif.pcap"
-dwn_ip
```

Figure 13. Configuration used for testing

Figure 14 shows the malware sending ARP packets asking for the MAC addresses of all the machines on the subnet specified in the config file, whilst ignoring itself (192.168.8.134) and the default gateway (192.168.8.135):


Figure 14. ARP Broadcasts

Figure 15 shows the malware sending ARP packets asking for the MAC addresses of all the machines on the subnet specified in the config file, whilst ignoring itself (192.168.8.134) and the default gateway (192.168.8.135): ARP Broadcasts

Once it has MAC addresses for all machines on the subnet, Ratsnif will then send unsolicited ARP packets to those addresses, updating the MAC address of the default gateway for each victim:


Figure 15. ARP Poisoning

Figure 16 shows the effect on the victim machine, with the attacker IP address and the default gateway IP address (192.168.8.135) both now sharing the same physical address:


Figure 16. arp -a results showing poisoned ARP Table on the victim machine

Once the ARP table is poisoned, all traffic destined for the default gateway will be routed through Ratsnif and can be stored and manipulated prior to retransmission.

Finally, Figure 17 shows a poisoned DNS response for www.google.com, whereby the DNS query was intercepted by Ratsnif, modified to point to an attacker controlled IP address and the fake response sent to the original requestor:


Figure 17. Ratsnif log file output showing ARP poisoning and DNS spoofing in action

C2

search.webstie.net

Whois

Attribute	Value
Server	whois.web4africa.net
Registrar	WEB4AFRICA INC
Email	contact@privacyprotect.org
Name	Domain Admin, C/O ID#10760
Organization	Privacy Protection Service INC d/b/a PrivacyProtect.org
Street	PO Box 16
City	Nobby Beach
State	Queensland
Postal	QLD 4218
Country	AUSTRALIA
Phone	4536946676

NameServers	ns21.cloudns.net
	ns22.cloudns.net
	ns23.cloudns.net
	ns24.cloudns.net

History

Obtained via Shodan, this history shows when the C2 server exposed various ports, including HTTP, SMB and RDP, for the purpose of controlling Ratsnif and other OceanLotus malware:

Figure 18. Shodan history for search.webstie.net

Conclusions

Ratsnif is an intriguing discovery considering the length of time it has remained undetected, likely due to limited deployment. It offers a rare glimpse of over two years of feature development, allowing us to observe how threat actors tailor tooling to their nefarious purposes. While all samples borrow heavily from open-source code/snippets, overall development quality is deemed to be poor. Simply put, Ratsnif does not meet the usual high standards observed in OceanLotus malware.

Appendix

Indicators of Compromise (IOCs)

Indicator	Type	Description
b4e3b2a1f1e343d14af8d812d4a29440940b99aaf145b5699dfe277b5bfb8405	SHA256	Ratsnif
b214c7a127cb669a523791806353da5c5c04832f123a0a6df118642eee1632a3	SHA256	Ratsnif
b20327c03703ebad191c0ba025a3f26494ff12c5908749e33e71589ae1e1f6b3	SHA256	Ratsnif
7fd526e1a190c10c060bac21de17d2c90eb2985633c9ab74020a2b78acd8a4c8	SHA256	Ratsnif
onceinstance	Mutex	Mutex name
search[.]webstie[.]net	Domain	C2
66.85.185.126	IP	search[.]webstie[.]net

dns[.]domain-resolve[.]org	Domain	C2
X:\Project\BotFrame\Debug\Client.pdb	PDB	PDB Path
ntdata.tmp	File	Packet capture output
Core Networking - Router Solicitation	Windows Firewall Rule	7fd5…

MITRE

Tactic	ID	Name	Notes
Discovery	T1040	Network Sniffing	Sniffs packets and saves to file
T1046	Network Service Scanning	ARP/SMB	
T1082	System Information Discovery	User/computer name, system directory and workstation information	
Command and Control	T1043	Commonly Used Port	HTTP/HTTPS
T1065	Uncommonly Used Port	65000 - 65536	
T1001	Data Obfuscation	RSA/AES C2 encryption	
Impact	T1493	Transmitted Data Manipulation	Performs packet interception, modification and retransmission

The BlackBerry Cylance Threat Research Team

About The BlackBerry Cylance Threat Research Team

The BlackBerry Cylance Threat Research team examines malware and suspected malware to better identify its abilities, function and attack vectors. Threat Research is on the frontline of information security and often deeply examines malicious software, which puts us in a unique position to discuss never-seen-before threats.

Back