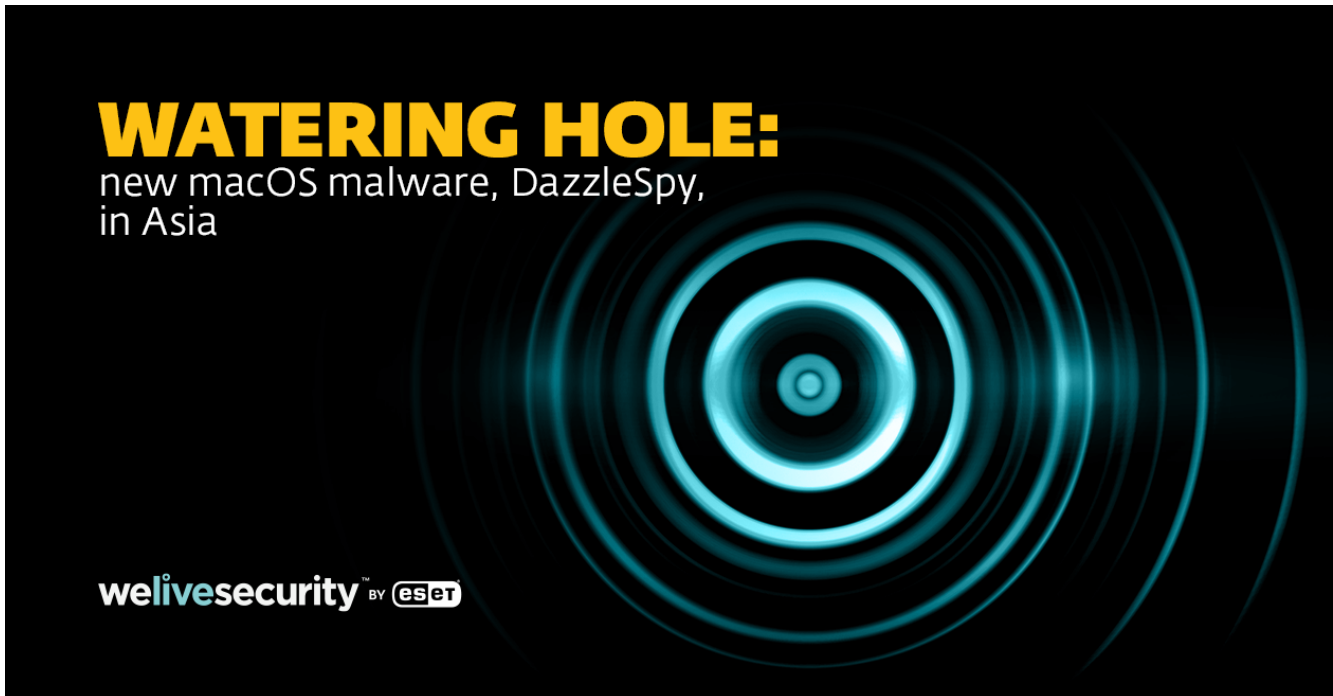


# Watering hole deploys new macOS malware, DazzleSpy, in Asia

[wlvsecurity.com/2022/01/25/watering-hole-deploys-new-macos-malware-dazzlespy-asia/](https://wlvsecurity.com/2022/01/25/watering-hole-deploys-new-macos-malware-dazzlespy-asia/)

January 25, 2022



Hong Kong pro-democracy radio station website compromised to serve a Safari exploit that installed cyberespionage malware on site visitors' Macs

On November 11<sup>th</sup>, Google TAG published a [blogpost](#) about watering-hole attacks leading to exploits for the Safari web browser running on macOS. ESET researchers had been investigating this campaign the week before that publication, uncovering additional details about the targets and malware used to compromise its victims. Here we provide a breakdown of the WebKit exploit used to compromise Mac users and an analysis of the payload, which is a new malware family targeting macOS. But first, let's look at how victims came into contact with the malicious code in the first place.

## Targets

It was [reported](#) by Felix Aimé from SEKOIA.IO that one of the websites used to propagate the exploits was a fake website targeting Hong Kong activists. We can read on its home page "Liberate Hong Kong, the revolution of our times". The very recent registration date of the [fightforhk\[.\]com](#) domain, October 19<sup>th</sup>, 2021, and the fact that the website is no longer accessible, supports that idea. We could also confirm that the [Internet Archive](#) cached a copy of the web page on November 13<sup>th</sup>. This copy includes the malicious iframe, as seen in Figure 1.



Figure 1. *fightforhk[.]com*, as archived by the Wayback Machine on November 13th

ESET researchers found another website, this time legitimate but compromised, that also distributed the same exploit during the few months prior to the Google TAG publication: the online, Hong Kong, pro-democracy radio station [D100](#). As seen in Figure 2, an iframe was injected into pages served by [bc.d100\[.\]net](#) – the section of the website used by subscribers – between September 30<sup>th</sup> and November 4<sup>th</sup> 2021.

Both distribution methods have something in common: they attract visitors from Hong Kong with pro-democracy sympathies. It seems that they were the primary target of this threat.



*Figure 2. Excerpt of [https://bc.d100\[.\]net/Product/Subscription](https://bc.d100[.]net/Product/Subscription) on November 4th 2021*

## **The exploit chain**

---

As seen in Figure 3, the page hosted on the malicious [amnestyhk\[.\]org](https://amnestyhk[.]org) domain checks for the installed macOS version and redirects to the next stage if the browser is running on macOS 10.15.2 or newer.

```
==]--STATUS: OK==  
  
http-equiv="Content-Type" content="text/html;cha  
a.navigator.userAgent;  
RegExp("^65 X |DNVd_|+)", "g");  
att.cookie += "|1|";  
  
_15_3' || os_ver == "10_15_4" || os_ver == "10_15_5" || os  
_15_7" || os_ver.startsWith("11")  
  
can.href = "/ss/4ba29d5b72266b28.html"  
  
can.href = "/RTHZ7jEQLLMA.html"
```

Figure 3. Content of the defaultaa.html page on amnestyhk[.]org

The next stage, named 4ba29d5b72266b28.html (see Figure 4) simply loads the JavaScript containing the exploit code – mac.js.

```
ntly: monospace;
ivesacu
t src="caps.js"></script>-->
mac.js"></script>
```

Figure 4. Content of the 4ba29d5b72266b28.html page

Note that the script tag to load caps.js has been commented out. The previous version of the exploit loaded Capstone.js from that file, while in the new version, Capstone.js is prepended to the exploit code in mac.js.

## The WebKit exploit

---

The exploit used to gain code execution in the browser is quite complex and had more than 1,000 lines of code once formatted nicely. It's interesting to note that some code, which suggests the vulnerability could also have been exploited on iOS and even on PAC-enabled (Pointer Authentication Code) devices such as the iPhone XS and newer, has been commented out, as seen in Figure 5.

```
1000; ++i) { // DFG(2000 on  
  
constructor_ptr.hi() > 0x10  
  
{  
setup_shellcode(memory, cur_
```

Figure 5. Excerpt of the JavaScript exploit containing comments about how to target iOS and PAC-enabled devices

We have confirmed that the [patch](#) identified by Google TAG does fix the vulnerability. While it is possible this vulnerability was assigned [CVE-2021-1789](#), we couldn't confirm due to the lack of publicly available technical details. Below we outline our understanding of how the vulnerability affects Safari versions prior to 14.1.

The exploit implements two primitives to gain memory read and write access: one to leak the address of an object (`addrof`) and one to create a fake JavaScript object from a given memory address (`fakeobj`). Using these two functions, the exploit creates two arrays of different types that overlap in memory, and thus is able to set a value in one of them that is treated as a pointer when accessed using the other. The technique is well described by Samuel Groß in his [multiple publications](#) on the subject. Below we explain the vulnerability that made the leakage of object addresses possible.

The exploit relies on a side effect caused by modifying an object property to be accessible via a “getter” function while enumerating the object’s properties in JIT-compiled code. The JavaScript engine erroneously speculates that the value of the property is cached in an array and is not the result of calling



In memory, the location of `getter[0]` is the same as this corrupted symbol's value. Thus, reassigning a value to `getter[0]` will change the value of the symbol. Its value is fetched from JavaScript using its `toString` method.

The updated JavaScriptCore code now checks whether the object contains properties with GetterSetter after the property enumeration, before considering whether the object's attribute can be accessed "quickly".

Detailing the fake object creation would require an article of its own. In short, it abuses the same bug, although this time the object is manipulated in a way that the JIT-compiled code accesses an item that is out-of-bounds and returns an address that was carefully sprayed on the heap before the fetch.

The rest of the code allows bypassing mitigations, such as the Gigacage, and loads the next stage.

As explained by Google TAG, the JavaScript loads a Mach-O executable file in memory. The rudimentary loader does not implement importing symbols from external libraries; instead, the addresses of `dlopen` and `dlsym` are patched into the loaded Mach-O. These can then be used from the executable to dynamically load and get the addresses of functions from external libraries.

## Privilege escalation to root

---

Now that code execution has been gained, the next stage is a Mach-O that is loaded into memory and executed. This Mach-O exploits a local privilege escalation vulnerability to run the next stage as root. Our examination confirms Google's analysis that the exploited vulnerability was described by Xinru Chi and Tielei Wang in a presentation at zer0con 2021, but it was also presented in more details at MOSEC 2021 by Tielei Wang. The vulnerability has been assigned CVE-2021-30869. Figure 7 shows a call to a function Tielei Wang called `adjust_port_type` in his last presentation. This function, responsible for changing the internal type of a Mach port, is implemented the same way in the Mach-O as was presented at MOSEC. Changing the type of a Mach port shouldn't be possible unless a vulnerability exists.



```
Pseudocode-B  
  
elf();  
y_entry_64(_mach_host_  
  
ED_ENTRY, IKOT_HOST_PRIV  
  
al_port(*v28, 0LL, 2LL
```

Figure 7. Altering the port type from `IKOT_NAMED_ENTRY` to `IKOT_HOST_PRIV` to gain access to special (privileged) Mach ports

To summarize, the Mach-O does the following:

1. Downloads a file from the URL supplied as an argument
2. Decrypts this file using AES-128-EBC and TEA with a custom delta
3. Writes the resulting file to `$TMPDIR/airportpaired` and makes it executable
4. Uses the privilege escalation exploit to remove the `com.apple.quarantineattribute` from the file to avoid asking the user to confirm the launch of the unsigned executable
5. Uses the same privilege escalation to launch the next stage with root privileges

The decrypted payload is where our analysis differs the most from what was described by Google TAG: the payload delivered to vulnerable visitors to the D100 site was new macOS malware we've named DazzleSpy.

## DazzleSpy

---

---

DazzleSpy is a full-featured backdoor that provides attackers a large set of functionalities to control, and exfiltrate files from, a compromised computer. Our sample is a Mach-O binary file compiled for x86\_64 CPU architecture.

## Persistence

---

In order to persist on the compromised device, the malware adds a Property List file (plist; see Figure 8) named com.apple.softwareupdate.plist to the LaunchAgents folder. The malware executable file is named softwareupdate and saved in the \$HOME/.local/ folder.

```
. version="1.0" encoding="UTF-8"?>
<TYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
i://www.apple.com/DTDs/PropertyList-1.0.dtd">
<!-- version="1.0" -->
<!--
<key>KeepAlive</key>
<true/>
<key>Label</key>
<string>com.apple.softwareupdate</string>
<key>ProgramArguments</key>
<array>
  <string>/var/root/.local/softwareupdate</string>
  <string>./</string>
</array>
<key>RunAtLoad</key>
<true/>
<key>SuccessfulExit</key>
<true/>
</dict>
</plist>
```

Figure 8. Property List file in LaunchAgents folder

## C&C communications

---

DazzleSpy connects to a hardcoded C&C server; the IP address and port found in the sample we decrypted was 88.218.192[.]128:5633. At first, the malware performs a TLS handshake, then uses a custom protocol to exchange JSON objects to deliver commands from the C&C server to compromised

Macs. DazzleSpy's binary contains an X.509 certificate used as a certificate authority (CA). It verifies that the server's certificate is issued by that authority. In practice, the same self-signed certificate is used for both the CA and the C&C server. The technique protects the malware's communications from potential eavesdropping by refusing to send data if end-to-end encryption is not possible.

Table 1 contains the list of commands supported by DazzleSpy. The first column is the name of the command that must be present in the JSON object received from the C&C server; many support optional or mandatory parameters.

*Table 1. DazzleSpy C&C commands*

<b>Command name</b>	<b>Purpose</b>
heartbeat	Sends heartbeat response.
info	Collects information about compromised computer, including: <ul style="list-style-type: none"> <li>• Hardware UUID and Mac serial number</li> <li>• Username</li> <li>• Information about disks and their sizes</li> <li>• macOS version</li> <li>• Current date and time</li> <li>• Wi-Fi SSID</li> <li>• IP addresses</li> <li>• Malware binary path and MD5 hash of the main executable</li> <li>• Malware version</li> <li>• System Integrity Protection status</li> <li>• Current privileges</li> <li>• Whether it's possible to use <a href="#">CVE-2019-8526</a> to dump the keychain</li> </ul>
searchFile	Searches for the specified file on the compromised computer.
scanFiles	Enumerates files in Desktop, Downloads, and Documents folders.
cmd	Executes the supplied shell command.
restartCMD	Restarts shell session.
restart	Depending on the supplied parameter: restarts C&C command session, shell session or RDP session, or cleans possible malware traces (fsck_hfs.log file and application logs).
processInfo	Enumerates running processes.
keychain	Dumps the keychain using a CVE-2019-8526 exploit if the macOS version is lower than 10.14.4. The public <a href="#">KeySteal</a> implementation is used.
downloadFileInfo	Enumerates the supplied folder, or provides creation and modification timestamps and SHA-1 hash for a supplied filename.
downloadFile	Exfiltrates a file from the supplied path.
file	File operations: provides information, renames, removes, moves, or runs a file at the supplied path.
uninstall	Deletes itself from the compromised computer.
RDPInfo	Provides information about a remote screen session.

Command name	Purpose
RDP	Starts or ends a remote screen session.
mouseEvent	Provides mouse events for a remote screen session.
acceptFileInfo	Prepares for file transfer (creates the folder at the supplied path, changes file attributes if it exists).
acceptFile	Writes the supplied file to disk. With additional parameters, updates itself or writes files required for exploiting the CVE-2019-8526 vulnerability.
socks5	Starts or ends SOCKS5 session (not implemented).
recoveryInfo	These seem like file recovery functions that involve scanning a partition. These functions do not seem to work and are probably still in development; they contain lots of hardcoded values.
recovery	

## Artifacts

While analyzing the DazzleSpy binary we found a number of interesting artifacts that might suggest an internal name for the malware and the authors' origin.

In several places (for example, see Figure 9) the malware refers to `osxrk` and the string `1.1.0` seems likely to be an internal version number.

```

if ( (v707 & 1) != 0 )
{
    v55 = objc_msgSend(val, "V_ID");
    v706 = objc_retainAutoreleasedReturnValue(v55);
    objc_msgSend(v990, "setObject:forKey:", v706, CFSTR("osxrkID"));
    objc_release(v706);
}
else
{
    objc_msgSend(v990, "setObject:forKey:", CFSTR("1.1.0"), CFSTR("osxrkID"));
}

```

Figure 9. Possible internal name and version number of the DazzleSpy malware

Moreover, it seems DazzleSpy's authors were not so concerned about operational security as they have left the username wanging in paths embedded in the binary. Figure 10 contains paths that reveal this username and internal module names.

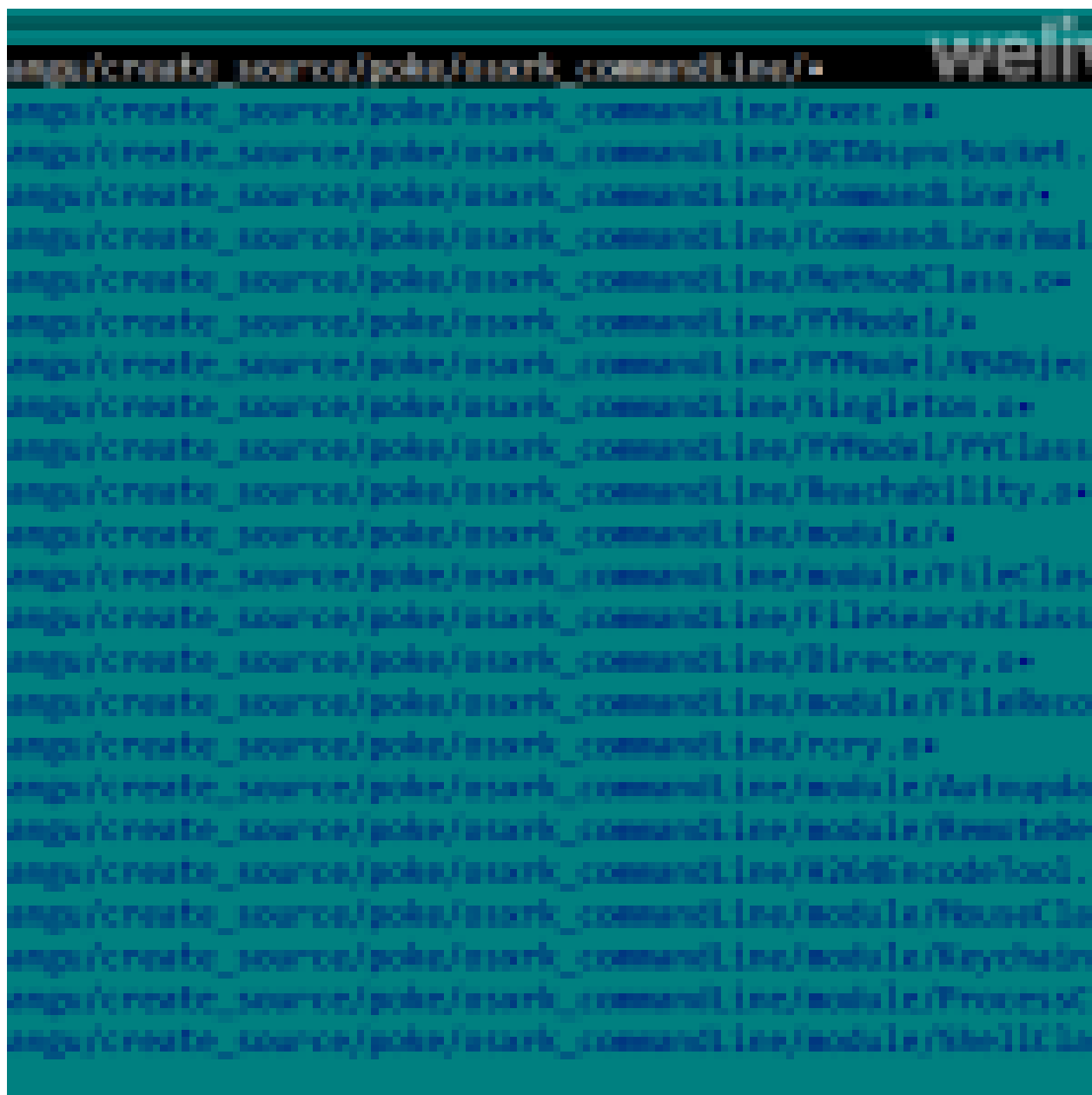


Figure 10. Paths embedded in the DazzleSpy binary

Once the malware obtains the current date and time on a compromised computer, as you see in Figure 11, it converts the obtained date to the Asia/Shanghai time zone (aka China Standard Time), before sending it to the C&C server.

```

1 id __cdecl -[MethodClass getSystemDate](MethodClass *self, SEL a2)
2 {
3     id v2; // rax
4     id v3; // rdi
5     id v4; // rdi
6     NSTimeZone *v5; // rax
7     id v6; // rax
8     id v8; // [rsp+10h] [rbp-50h]
9     NSTimeZone *v9; // [rsp+18h] [rbp-48h]
10    id location; // [rsp+28h] [rbp-38h] BYREF
11    id v11; // [rsp+30h] [rbp-30h] BYREF
12    id v12; // [rsp+38h] [rbp-28h] BYREF
13    double v13; // [rsp+40h] [rbp-20h]
14    id v14[3]; // [rsp+48h] [rbp-18h] BYREF
15
16    v14[2] = self;
17    v14[1] = a2;
18    v2 = objc_msgSend(&OBJC_CLASS__NSDate, "date");
19    v14[0] = objc_retainAutoreleasedReturnValue(v2);
20    v13 = (objc_msgSend)(v14[0], "timeIntervalSinceNow");
21    v3 = objc_alloc(&OBJC_CLASS__NSDate);
22    v12 = objc_msgSend(v3, "initWithTimeIntervalSinceNow:", v13);
23    v4 = objc_alloc(&OBJC_CLASS__NSDateFormatter);
24    v11 = objc_msgSend(v4, "init");
25    v5 = objc_msgSend(&OBJC_CLASS__NSTimeZone, "timeZoneWithName:", CFSTR("Asia/Shanghai"));
26    v9 = objc_retainAutoreleasedReturnValue(v5);
27    objc_msgSend(v11, "setTimeZone:", v9);
28    objc_release(v9);
29    objc_msgSend(v11, "setDateFormat:", CFSTR("yyyy年MM月dd日HH小时mm分ss秒"));
30    v6 = objc_msgSend(v11, "stringFromDate:", v12);
31    location = objc_retainAutoreleasedReturnValue(v6);
32    v8 = objc_retain(location);
33    objc_storeStrong(&location, 0LL);
34    objc_storeStrong(&v11, 0LL);
35    objc_storeStrong(&v12, 0LL);
36    objc_storeStrong(v14, 0LL);
37    return objc_autoreleaseReturnValue(v8);
38 }

```

Figure 11. Decompiled code of the getSystemDate function

In addition, it should be noted that the DazzleSpy malware contains a number of internal messages in Chinese, for example as seen in Figure 12.

```

else
{
    v576 = val;
    v177 = objc_msgSend(
        &OBJC_CLASS__NSString,
        "stringWithFormat:",
        CFSTR("\n%s\n%d\n%s"),
        "Singleton.m",
        828LL,
        "-[Singleton analysisData:Socket:]");
    v575 = objc_retainAutoreleasedReturnValue(v177);
    objc_msgSend(v576, "poError:location:", CFSTR("没有这个文件, 或者已经被删除!"), v575);
    objc_release(v575);
    v998 = 1;
}

```

Figure 12. Internal error message in Chinese

## Conclusion

Given the complexity of the exploits used in this campaign, we assess that the group behind this operation has strong technical capabilities. While there is information published online about the local privilege escalation (LPE) vulnerability used here, we couldn't find anything about the specific WebKit vulnerability used to gain code execution in Safari. It's also interesting that end-to-end encryption is

enforced in DazzleSpy and it won't communicate with its C&C server if anyone tries to eavesdrop on the unencrypted transmission by inserting a TLS-inspection proxy between the compromised system and the C&C server.

The watering-hole operations this group has pursued show that its targets are likely to be politically active, pro-democracy individuals in Hong Kong. This campaign has similarities with one from 2020 where LightSpy iOS malware (described by [TrendMicro](#) and [Kaspersky](#)) was distributed the same way, using iframe injection on websites for Hong Kong citizens leading to a WebKit exploit. We cannot confirm at this point whether both campaigns are from the same group, but ESET Research will continue to track and report on similar malicious activities.

## Indicators of Compromise (IoCs)

---

### Samples

---

SHA-1	Filename	ESET detection name	Description
F3772A23595C0B51AE32D8E7D601ACBE530C7E97	mac.js	JS/Exploit.Agent.NQK	JavaScript code with WebKit exploit launching an LPE.
95889E0EF3D31367583DD31FB5F25743FE92D81D	N/A	OSX/Exploit.Agent.C	Mach-O file with LPE launching next stage.
EE0678E58868EBD6603CC2E06A134680D2012C1B	server.enc	OSX/DazzleSpy	DazzleSpy Mach-O after decryption of server.enc.

### Filenames

---

- \$HOME/Library/LaunchAgents/com.apple.softwareupdate.plist
- \$HOME/.local/softwareupdate
- \$HOME/.local/security.zip
- \$HOME/.local/security/keysteaDaemon
- \$HOME/.local/security/libkeysteaClient.dylib

### Network

---

#### URLs of Safari exploit

---

- [https://amnestyhk\[.\]org/ss/defaultaa.html](https://amnestyhk[.]org/ss/defaultaa.html)
- [https://amnestyhk\[.\]org/ss/4ba29d5b72266b28.html](https://amnestyhk[.]org/ss/4ba29d5b72266b28.html)
- [https://amnestyhk\[.\]org/ss/mac.js](https://amnestyhk[.]org/ss/mac.js)
- [https://amnestyhk\[.\]org/ss/server.enc](https://amnestyhk[.]org/ss/server.enc)

## DazzleSpy C&C server

---

88.218.192[.]128:5633

## DazzleSpy CA certificate

---

SHA-256: 1F862B89CC5557F8309A6739DF30DC4AB0865668193FDF70BA93F05D4F8C8B8

```
1 Certificate:
2 Data:
3 Version: 1 (0x0)
4 Serial Number: 10557282746731470350 (0x928300b9284a1e0e)
5 Signature Algorithm: sha256WithRSAEncryption
6 Issuer: C=11, ST=11, L=11, O=11, OU=11, CN=11/emailAddress=11@qq.com
7 Validity
8 Not Before: May 18 07:26:17 2021 GMT
9 Not After : May 16 07:26:17 2031 GMT
10 Subject: C=11, ST=11, L=11, O=11, OU=11, CN=11/emailAddress=11@qq.com
11     Subject Public Key Info:
12     Public Key Algorithm: rsaEncryption
13     Public-Key: (2048 bit)
14     Modulus: ...
15     Exponent: 65537 (0x10001)
16     Signature Algorithm: sha256WithRSAEncryption
17     -----BEGIN CERTIFICATE-----
18     MIIDTCCAjQCCQCSgwC5KEoeDjANBgkqhkiG9w0BAQsFADBoMQswCQYDVQQGEwIx
19     MTELMakGA1UECAwCMTEExCzAJBgNVBACMAjExMQswCQYDVQQKDAIxMTELMakGA1UE
20     CwwCMTEExCzAJBgNVBAMMAjExMRgwFgYJKoZIhvcNAQkBFgkxMUBxcS5jb20wHhcN
21     MjEwNTE4MDcyNjE3WhcNMzEwNTE2MDcyNjE3WjBoMQswCQYDVQQGEwIxMTELMakG
22     A1UECAwCMTEExCzAJBgNVBACMAjExMQswCQYDVQQKDAIxMTELMakGA1UECwwCMTEEx
23     CzAJBgNVBAMMAjExMRgwFgYJKoZIhvcNAQkBFgkxMUBxcS5jb20wgwEiMA0GCSqG
24     SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDFfrP+LbCk9KhPH2gQ3V5IBWCpuM+yzBzn
25     ofL2RJiTMedP467Js4wzrP+qCkXs9STaOZCvYRFaCmfY9bG7PsrgqG90OHfVktG
26     5xldEpd5XPI+GYI/48ridpE7mgw+KO0oRxoyUO1if9nRXvHNGmx0C3i9Rb6ahynv
27     dEBAZVxeX20fDHMr0dvVe4TKst9g5W02o31zU54mx2f7m2Kgit+n+UsDA/uBRF/c
```



28 GcWsvQFVlCGuFmBDt58t98BO5nEmI3iDEfUi8FTf2HVSS0LAYC83IkwZyWpML9Jn  
 29 uVg67KFKprNMmzBxDK0eDa9ZHObohj3iscM3IYXICnicbOLYTCvRAgMBAAEwDQYJ  
 30 KoZlhvcNAQELBQADggEBAAvkJC5Fi8+Kz8roBhzCY3ayPLMMMj49aHGU/JDLZwsh  
 31 WSng5/eY7LrGkTqP0tKay/rrxQvyMeZftvB0DMCbxu0vndK/jTqruxS+ZXDkqyOb  
 32 ykU0Z6TqRZ/ltgck9ii4R6PgUEynrJVZHtUHDtemulpHgPRjkFDA4emOui1kFdNT  
 33 gnUr0vgh12KIVNAm64UVh9kkneCTFZtYeCAGNw5kFknv5OgsjcaueqCsm3a3dxFq  
 34 7JqReIV1WDx+QEBXgM4itvQRY+d5pv5eOlz8sBzxFR7+Gh/Q9aJoPL+ZX7kouMEU  
 35 bKwsEwNcRwZWQu41ghFi/8MdqBxb2Nb9H4gCupqKdil=  
 36 -----END CERTIFICATE-----

## MITRE ATT&CK techniques

This table was built using version 10 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Resource Development	<u>T1583.001</u>	Acquire Infrastructure: Domains	Domain names such as amnestyhq[.]org were acquired to use on compromised web servers.
	<u>T1583.004</u>	Acquire Infrastructure: Server	Servers (or virtual servers) were rented to serve WebKit exploits and used as C&C servers for DazzleSpy.
	<u>T1584.004</u>	Compromise Infrastructure: Server	A legitimate website was compromised to add an iframe loading malicious JavaScript code.
	<u>T1587.001</u>	Develop Capabilities: Malware	DazzleSpy is macOS malware developed to steal information from its victims.
	<u>T1587.003</u>	Develop Capabilities: Digital Certificates	DazzleSpy verifies the authenticity of its C&C server using an X.509 certificate.
	<u>T1587.004</u>	Develop Capabilities: Exploits	An undocumented Safari exploit was used to compromise the targets.
	<u>T1608.004</u>	Stage Capabilities: Drive-by Target	This operation compromised a website that is likely to be visited by its targets, to distribute malware.

<b>Tactic</b>	<b>ID</b>	<b>Name</b>	<b>Description</b>
Initial Access	<u>T1189</u>	Drive-by Compromise	The compromised website served the exploit to visitors using Safari on a Mac.
Execution	<u>T1569</u>	System Services	The exploit sends Mach messages to launchd to remove the quarantine flag and to kuncd to launch the malware.
Persistence	<u>T1543.001</u>	Create or Modify System Process: Launch Agent	DazzleSpy persists by installing a Launch Agent.
Privilege Escalation	<u>T1068</u>	Exploitation for Privilege Escalation	An LPE exploit for macOS is used to elevate privileges to root.
Defense Evasion	<u>T1620</u>	Reflective Code Loading	The LPE exploit downloading the next stage is loaded and executed in memory only.
Credential Access	<u>T1555.001</u>	Credentials from Password Stores: Keychain	DazzleSpy can steal credentials from the macOS keychain.
Discovery	<u>T1083</u>	File and Directory Discovery	DazzleSpy can be used to enumerate files in specific folders.
<u>T1057</u>	Process Discovery	DazzleSpy can obtain the list of running processes.	
<u>T1082</u>	System Information Discovery	DazzleSpy can obtain the macOS version.	
<u>T1016</u>	System Network Configuration Discovery	DazzleSpy can obtain the IP address and Wi-Fi SSID.	
<u>T1033</u>	System Owner/User Discovery	DazzleSpy can obtain the current username from a compromised Mac.	
<u>T1124</u>	System Time Discovery	DazzleSpy can obtain the system time on a compromised Mac.	
Collection	<u>T1005</u>	Data from Local System	DazzleSpy can search for documents on the compromised system.
<u>T1113</u>	Screen Capture	DazzleSpy has the ability to record screen activity.	
Command and Control	<u>T1071</u>	Application Layer Protocol	DazzleSpy uses a custom JSON-based protocol for its C&C communications.

Tactic	ID	Name	Description
<u>T1132.001</u>	Data Encoding: Standard Encoding	DazzleSpy uses base64 to encode parts of its C&C communications.	
<u>T1573</u>	Encrypted Channel	DazzleSpy uses TLS encryption.	
<u>T1571</u>	Non-Standard Port	DazzleSpy uses TCP port 5633.	
Exfiltration	<u>T1041</u>	Exfiltration Over C2 Channel	DazzleSpy exfiltrates data over its C&C communications channel.



25 Jan 2022 - 11:30AM

***Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center***

**Newsletter**

---

**Discussion**

---