'RustBucket' malware targets macOS

jamf.com/blog/bluenoroff-apt-targets-macos-rustbucket-malware/

Jamf Blog



April 21, 2023 by Jamf Threat Labs

BlueNoroff APT group targets macOS with 'RustBucket' Malware

Jamf Threat Labs

Learn about the macOS malware variant discovered by Jamf Threat Labs named 'RustBucket'. What it does, how it works to compromise macOS devices, where it comes from and what administrators can do to protect their Apple fleet.

By Ferdous Saljooki and Jaron Bradley

Jamf Threat Labs has discovered a macOS malware family that communicates with command and control (C2) servers to download and execute various payloads. We track and protect against this malware family under the name 'RustBucket' and suspect it to be attributed to a North Korean, state-sponsored threat actor. The APT group called BlueNoroff is thought to act as a sub-group to the well-known Lazarus Group and is believed to be behind this attack. This attribution is due to the similarities noted in a <u>Kaspersky blog</u> entry documenting an attack on the Windows side. These similarities include malicious tooling on macOS that closely aligns with the workflow and social engineering patterns of those employed in the campaign.

Stage-One

The stage-one malware (0be69bb9836b2a266bfd9a8b93bb412b6e4ce1be) was discovered while performing normal hunting routines for compiled AppleScript applications that contained various suspicious commands. Among our results, we identified a suspicious AppleScript file titled main.scpt contained within an unsigned application named Internal PDF Viewer.app. It should be noted that we have no reason to believe this application is allowed to execute without the user manually overriding Gatekeeper.

Internal PDF Viewer
Application
Version 1.0
com.apple.ScriptEditor.id.asc
Intel-64-bit
Copyright -
277 КВ
Last modified Mar 16, 2023 at 10:28:46 PM
App Sandbox 😑 Not enabled
Hardening 😑 Not enabled
Notarization 🛛 😑 None detected
Gatekeeper 🛛 😑 Can't evaluate
Signed By 🙁 No signature
Open With Apparency

The directory structure for the stage-one dropper is shown below. As with all compiled AppleScript applications, the primary app code is within the main.scpt file, located within the /Contents/Resources/Scripts/ directory.

Although the AppleScript was compiled, we were able to extract its contents by loading it into the macOS Script Editor application. When launched, the dropper executes the code seen below:

The stage-one simply executes various do shell script commands to download the stagetwo from the C2 using curl. The malware writes and extracts the contents of the zip file to the /Users/Shared/ directory and executes the stage-two application also named Internal PDF Viewer.app. By breaking up the malware into several components or stages, the malware author makes analysis more difficult, especially if the C2 goes offline. This is a clever but common technique used by malware authors to thwart analysis.

At the time of our analysis, both the stage-one and stage-two components of this malware were undetected on VirusTotal.

\bigcirc	O No security vendors and no sandboxes flagged this file as malicious		@ ℃ 跳
0 Community Score	7c66d2d75be43d2c17e75d37c39344a9b5d29ee5c58611178aa7d9t34208eb48 main.scpt appinscript	1.12 KB 2023-04-11 12:24:25 UTC Size 1 day ago	0
\bigcirc	⊘ No security vendors and no sandboxes flagged this file as malicious		(中) ※
Community Score	bea33tb3205319868784c028418411ee796d6ee3dfe9309f143e7e8106116a49 Internal PDF Viewer macho 66bits	101.02 KB 2023-03-25 00:52:54 UTC Size 18 days ago	

Stage-Two

Although the stage-two (ca59874172660e6180af2815c3a42c85169aa0b2) application name and icons look very similar to stage-one, the directory structures are different and there is no use of AppleScript in the latter. The application version, size and bundle identifier com.apple.pdfViewer — are also notably different, masquerading as a legitimate Apple bundle identifier. This application is signed with an ad-hoc signature as well.

	Internal PDF Viewer
PDF	Application Version 3.0 com.apple.pdfViewer Notecomble and Copyright 842 KB Last modified Mar 2, 2023 at 12:36:53 AM App Sandbox P Not enabled Hardening Not enabled Matarization None detected Gatekeeper None detected Gatekeeper Can't evaluate Signed By Ad-hoc signature

The application layout is that of a much more traditional app and is written in Objective-C.

When the Internal PDF Viewer application is launched, the user is presented with a PDF viewing application where they can select and open PDF documents. The application, although basic, does actually operate as a functional PDF viewer. A task that isn't overly difficult using Apple's well-built PDFKit Framework.



Upon execution, the application does not perform any malicious actions yet. In order for the malware to take the next step and communicate with the attacker, the correct PDF must be loaded. We were able to track down a malicious PDF

(7e69cb4f9c37fad13de85e91b5a05a816d14f490) we believe to be tied to this campaign, as it meets all the criteria in order to trigger malicious behaviors.

For example, when the malicious PDF is double-clicked from within Finder the user will see the following:



This minimal message informs the user that they must open the PDF using the necessary application in order to see the full details.

When opened within the malicious PDF viewer, the user will see a document (9 pages in total) that shows a venture capital firm that is interested in investing in different tech startups. From what we can tell, the PDF was created by taking the website of a small but legitimate venture capital firm and putting it into PDF format.



It should be noted here that earlier, the stage-one dropper reached out to cloud[.]dnx[.]capital, thus keeping on theme with the disguise of a venture capital firm.

This PDF viewer technique used by the attacker is a clever one. At this point, in order to perform analysis, not only do we need the stage-two malware but we also require the correct PDF file that operates as a key in order to execute the malicious code within the application.

So, how is the malware displaying a different PDF than the one loaded by the user?

To answer this, we take a closer look into some of the functions within the app. Most notably, we see one titled viewPDFas part of the PEPWindow class. This function seeks to a specific offset within the loaded PDF to check for a specific blob of data. If the expected data is present, a function called _encrypt_data is invoked, which, ironically runs code to decrypt the blob and produce a new PDF. It does this using a hardcoded 100-byte XOR key which can be found in the __CONSTdata of the executable.

• • •	-	Internal PDF Viewer		
RAW RVA	-		٩	
 Executable (X86_64) Mach64 Header Load Commands Section64 (_TEXT,_text) Section64 (_TEXT,_stubs) Section64 (_TEXT,_objc_methname) Section64 (_TEXT,_objc_dissname) Section64 (_TEXT,_objc_methtype) Section64 (_TEXT,_objc_methtype) Section64 (_TEXT,_costing) Section64 (_TEXT,_const) Section64 (_TEXT,_unwind_info) Section64 (_DATA,_nl_symbol_ptr) Section64 (_DATA,_ot) 	pFile 00007E20 00007E30 00007E40 00007E50 00007E60 00007E70 00007E80 00007E80 00007E80	Data LO 00 00 00 00 00 08 40 00 00 00 00 00 00 40 8F 40 00 00 00 00 00 00 00 24 40 29 6C E1 9D D6 39 13 C0 68 4C B4 47 0B A0 D0 D6 B3 2B D9 FF 8D 28 19 21 A9 39 A6 94 9E 19 24 C7 8D AA 17 98 8D CC 0C DE 1F 3F 7D 15 FC 09 AB 33 FB 76 BD 4A 00 00 00 00 00	Data HI 00 00 00 00 00 00 3F 00 00 00 00 00 00 00 00 00 00	Value @? @@)l9^D hL.Gu\ .+(.! .9\$.3.~.u" MM_K.n .?}3?@

This newly decrypted PDF is then displayed to the user in the application, providing the illusion that this app was truly necessary in order to view the full details of the PDF.

Since the embedded PDF file is loaded directly into the viewer, it is never written to the disk. Using a disassembler — such as Hopper — we can extract it by placing a breakpoint on the return in the encrypt_data function.



If analyzing the ARM executable (as opposed to the Intel executable), we can print the \$x0 register which gives us all the bytes of the decrypted blob. Saving these bytes into a file will also reveal the inner PDF file.

Dentrol	
Executable file: Innvata/Impdinte	arnal PDF Viewer 2.app/Contents/MacOS/internal PDF Viewer
Working directory:	
Arguments:	
Controls	
• • • • • •	🙃 🔹 🔹 Signaled (Signal 5 = SIGTRAP) Attach to Process
Threads	Calitack
Thread 69350: Breakpoint Thread 69397: None	0x100001874 - Internal PDF Viewer encrypt data 0x100001a40 - Internal PDF Viewer - PSPWindow view90511
Thread 69495: None	0x100001f48 - Internal PDF Viewer - (PEFWindow initialize:)
Thread 69496: None	0x1000016cc = Internal PDF Viewer 28-[AppDelegate openDocument:] block invoke
Thread 69553: None Thread 69554: None	0xlbcfc4760 - AppKit - [NSSaveFanel didEndFanelWithReturnCode:]
Thread 69555: None	0x1befc4db4 = AppKit =[NSSavePanel completeWithReturnCode:url:urls:]
Thread 69556: None	<pre>0xlbefc6bec = AppKit =[NSSavePanel observeValueForKeyPath:ofObject:change:context:]</pre>
	Oxlbc8eal3c - Foundation NSKeyValueNotifyObserver
	0x1bcs69770 = Foundation N3KeyValueDidChangeNithPerThreadFendingNotifications
	0x1c2d2e564 - ViewBridge 41-[NSViewBridge setObject:forKey:withKVO:] block invoke
	0x1c2db6970 - ViewBridge withHintInProgress
	0x1c2d257c8 - ViewBridge -INSViewBridge metObject:forKev:withKV0;1
	GPR Memory Debugger Console Application Output
no \$x0	
<25584446 2d312e35 @d@a25b5 b5b5b	5584 04312888 20616254 04881c3c 21547978 65214381 74615c6f 67215861 6765728 3220020 52214c61 66572865 66205553 20202153 74727563 0673 7464173 6406465 4415452 4415459 4651758 45164174 375464 645646665728 3220020 52214c61 66572865 66205553
67657321 4361756e 74283921 4b6964	473 56283328 38285228 31332838 28522833 33283828 52283336 28382852 28343428 38265228 34372838 28522835 38283828 52283536 28382852
6563743c 3c2f496d 61676535 203526	194 01510406 68334088 20016204 80683621 2134/574 05213061 61626120 617403268 1440326 3468527 3405/81 73724380 7382527 3416244 082 2052249 66616752 31820831 2020102 5530427 4578474 5374614 6551528 47531668 36260398 52247453 32080398 32247528
69614261 78552838 28382836 313228	632 20313120 3020523e 3e215072 61635365 74562150 44462154 65787421 49666167 65422149 66616765 43214966 61676549 56208563 21466564 037 32305420 2143616e 74656e74 73203420 30205221 47726175 703c3c21 54797065 21477261 75702153 21547261 6e737061 72656e63 79214353
2f446576 69636552 47423e3e 2f5463 65636f64 652f4c65 6e677468 203335	162 73215321 53747275 63745861 72656674 7328383e 3e8d8a65 6e646162 6a8d8a34 28382861 626a8d8a 3c3c2146 696c7465 7221466c 61746544 933 3e3e8d8a 73747265 616d8d8a 789cb594 414b8331 18c5e181 7c8739aa 87d99949 2671a878 88d552a1 a8b66841 3c88d49e aaa8d111 cc6eab5d
da5a2bae 97dd6593 ccfbbd97 49a0bd a3c5c37c a6307801 6b6be07c dc0766	s82 5eaflaf7 4783a0d3 53381bf4 alla4e82 ccdfad79 b5868090 88c47bf6 1858200a cldbcc9a db1378b6 26b04299 824e85eb 2f8ea9bc 1f17d654 869 f14aab96 a8858872 294e3b24 ceada5d8 05837b94 a8387db2 861b4d06 17901442 24f46568 d1947285 939043a1 297f33bc cdcbf2e1 24378676
8cdf@cad b93b82e3 7b985e5a 735ec4 02119deb 0cf60b8c 83d421ac c05c12	476 e14a17b8 aa8a2142 50c22cfb 780fc5fc 134dc819 937648b3 159a5b87 b6dc20f5 8x9448a9 5166cee8 e2af76fa ff58fd01 ace93856 918c3196 22c 677808d6 36a59ba6 7e58bbaf 774b1c65 ae8fa18b 475ba664 37cf55d8 9430c56f 3c85ed8d dab7f4d3 92a2ecea 28d684be 4b476d4d d8466d12
9475a87c 1b4adad9 73168cch ch3483	7Fa ab5997ea 89ebeb94 e9bbfb34 EdcEe763 Eeb4ca41 17745b89 c87a887E 74acEca1 Eef958f7 Ef578dfb 81dfE43h bb8d8aE5 Ee64737E 776561E4
Debugger.command	

Stage-Two Communication

So far we've decoded the PDF file that is embedded within the original PDF file, but as we stated earlier, this is the point where the malware will also phone home to the attacker. Much like the inner PDF document, the attacker's C2 is also XOR encoded within the original PDF. This is why we see the encrypt_data function run a second time. The following bytes are passed to it which can be found towards the bottom of the original PDF document.

This time when the encrypt_data function runs using the same hardcoded XOR key as before, it returns the following:

68747470 733A2F2F 646563	8 25333176 65657475 72657325	696E666F 2F69354F 7644455F	https://deck.31ventures.info/i50vDE_
52422F72 5548536E 6C3372	5 752F5639 516A307A 66526A6C	2F687A32 64687751 4047652F	RB/rUHSnl3rUu/V9Qj@zfRj1/hzZdhwQMGe/
36347556 41375865 714259	6 65396744 2144	AND 1000 (77) (2000) (50)	64uVA7Peq8Yfe9g0/D

After the embedded PDF has been displayed to the user and the URL has been deobfuscated, the malware then calls a function titled <u>_downAndExecute</u> and makes a POST request to a C2 server to presumably retrieve and execute a stage-three payload.



In the <u>_downAndExecute</u> function shown below, we can see the various parameters being set in order to initiate an HTTPrequest.



The malware also creates a new thread and sleeps before making the **POST** request again in a loop until an HTTP 200 response is returned.

Unfortunately, at the time of our analysis, the server was not responding with the necessary message.



We have however managed to discover a new URL on the same domain that is hosting a Mach-O executable that we believe to be the new location of the final payload.

If the stage-two dropper succeeds in downloading the stage-three payload, we can view the next actions within the downAndExecute_block_invoke.



The aforementioned image shows the following steps taking place if the C2 responds:

1. The malware creates a temporary directory and writes the received file to that temporary directory. The name of that malicious file will be the current mach timestamp (the number of seconds since midnight January 1st, 2001). An example file path would look like this:

/var/folders/g6/w3s4hg8n57sgfjl4xgrhjs_w0000gn/T/7035176 04263

- 2. Executable permissions are assigned to the new file.
- 3. The program arguments are set and the file is executed. The set argument is that of the attacker C2 decoded from this stage two payload. The stage-three will go on to use this value.

Stage-Three

The stage-three payload (182760cbe11fa0316abfb8b7b00b63f83159f5aa) is an ad-hoc signed trojan written in Rust and weighing in at a sizable 11.2MB. It's a universal binary that holds both ARM and x86 architectures. Upon initial execution, it performs a handful of system recon commands.

One of the earliest used modules is titled webT::getinfo. Within this module is the ability to look at the basic info about the system, process listing, current time and whether or not it's running within a VM. The functions are named accordingly.

Address	Туре	Name
0x10000a9d4	Р	webT::make_status_string::h7a82ba076c0dc67f
0x10000ac04	Р	<pre>webT::send_request::hfcdd5dd674401a33</pre>
0x10000b1f4	Р	webT::main::h3abdbc4821fd6bb0
0x10000c1ec	Р	_\$LT\$webTCustomError\$u20\$as\$u20\$corefmtDebug\$GT\$::fmt::h8ce892e1dc1c74;
0x10000d8d8	Р	<pre>webT::getinfo::get_comname::h493dc40b2a15d42e</pre>
0x10000d9d4	Р	<pre>webT::getinfo::get_osinfo::hd4634312e1544d62</pre>
0x10000dac8	Р	<pre>webT::getinfo::get_installtime::ha0426d17132a18a3</pre>
0x10000ded8	Р	<pre>webT::getinfo::get_boottime::h460919080572949c</pre>
0x10000e160	Р	webT::getinfo::get_currenttime::h7781115e5374bc3d
0x10000e2f4	Р	<pre>webT::getinfo::get_vmcheck::hc393d0a579c3a132</pre>
0x10000e7a8	Р	<pre>webT::getinfo::get_processlist::h78c5b10552853da6</pre>

Running this malware results in communication to the URL provided as the first argument passed at execution time. The WebT::send_request function is responsible for sending the initial message to the C2 server. When placing a breakpoint on it, we can step over it resulting in a call to the server.

add x add x bl	<pre>o, sp, #exce 1, sp, #0x78 2, sp, #0x270 _2N4webT12send_reques</pre>	t17hfcdd5dd674	(01a33E 1 webTilsend			
add x	2, sp, #0x270 _2N4webT12send_reques	t17hfcdd5dd674	Mala33E ; webTijsond			
exec	2N4webT12send_reques	t17hfcdd5dd674	Mela33E ; webT::send	and a second state of the state of the local		
exec				requestsshifedd	5dd674481a33	
exec			LuLu Alert			
	= 182760	cbellfa03	16abfb8b7b00b6	53f83159f	iaa 🔽	Q
	is trying to	connect to	104.255.172.56		virus total	ancestry
rocess Info	44825					
rocess args: rocess path:	https://deck.3lve /tmp/182760cbe11f	ntures.info/i50 a0316abfb8b7b00	JvDE_RB/rUHSnl3rUu/V9 Jb63f83159f5aa	Qj@zfRjl/hz2dh	wQMGe/64u/VA7PeqBYfe	9gD/D
etwork Info						
p address:	104.255.172.56					
ort & pretoco	1: 443 (TCP)					
everse das na	se: unknown					
			rule scope:			
			Process	Θ	Block	Allow
inestamp: 16:	44:56				temporarily	(pid: 44025
Enter a Pytho	n Command					

This payload allows the attacker to carry out further objectives on the system, but perhaps a deep dive on stage-three is best saved for another blog post.

At a High Level

We dove fairly deeply into some of the different actions of this malware. At a higher level, the workflow looks like the following:



Connections to BlueNoroff

There are a few signs that this malware is tied to BlueNoroff. First and foremost is the domain used in the stage-one dropper: cloud[.]dnx[.]capital. This domain was reported as being used by the attackers in a <u>writeup done by Proofpoint</u>. In the previously mentioned Kaspersky blog, it was reported that the attackers had created numerous fake domains impersonating venture capital firms and banks in a campaign Kaspersky titled 'SnatchCrypto'. This aligns with the social engineering schemes discovered in the PDF document. The Windows malware also used the "decoy document" approach which clearly worked well for the attacker. The earliest submission of the "Internal PDF Viewer" we could find on VirusTotal was uploaded in January 2023 and we've observed the attackers continuing to host it.

While many different PDF payloads exist that work on Windows, so far only one PDF has been discovered that will result in a call to the attacker on macOS. We do suspect more than just this one PDF exists. It's worth noting that the XOR key found within the malware can also be found within a variety of malicious PDF files. However, when loaded into the Viewer application, these files do not result in a properly decoded URL. We suspect a different variant of the malicious viewer (or perhaps a different platform) is capable of loading the XOR key from within the PDF instead of the attackers hardcoding it in the malicious app.

Conclusion

The malware used here shows that as macOS grows in market share, attackers realize that a number of victims will be immune if their tooling is not updated to include the Apple ecosystem. Lazarus group, which has strong ties to BlueNoroff, has a long history of attacking macOS and it's likely we'll see more APT groups start doing the same.

Jamf Protect defends against the malicious components of this malware and blocks the malicious domains. Jamf Threat Labs will continue to monitor BlueNoroff's activity on this campaign.

Thre	eat view					
Malw	are network tr	affic				
Wh	at does this th	reat mean? I How to	remediate			>
(c)	unu open threats	Change are blocked through	Either for	Backed threats (2) +		Benet all
	Severity	Threat name	Devicer	s Läst seen	Events	
		Malware network traffic cloud.dnx.capital	1 device	21 seconds ago	3 events	Manage policy Archive
	User	Device	Days active	Last seen		Events
4	Nathan	Mac	0	Apr 20, 2023 (08:16:53	3 events
-		Malware network traffic deck.31ventures.info	1 device	a 39 seconds ago	3 events	Manage policy Archive
	User	Device	Days active	Last seen		Events
Ц	Nathan	Mac	0	Apr 20, 2023 0	08:16:36	3 events

A shout out to <u>Patrick Wardle</u> for his collaboration on some of the analysis here. If you're looking to learn more about the analysis of macOS malware, check out the free online book: <u>The Art of Mac Malware</u>.

Indicators of Compromise

References:

Ensure your macOS endpoints are protected from current and novel Maccentric threats.

Don't just take Jamf's word for it, put Jamf Protect to the test today.

Request Trial

Jamf Threat Labs Jamf

Jamf Threat Labs is a global team of experienced threat researchers, cybersecurity experts and data scientists with skills that span penetration testing, network monitoring, malware research and app risk assessment. Jamf Threat Labs primarily monitors and explores emerging threats affecting Mac and mobile devices. The team's research is published with the aim of raising awareness of specific threats while also improving awareness and advocacy of security practices to protect the modern workforce.

Subscribe to the Jamf Blog

Have market trends, Apple updates and Jamf news delivered directly to your inbox.

To learn more about how we collect, use, disclose, transfer, and store your information, please visit our <u>Privacy Policy</u>.