Old cat, new tricks, bad habits

pwc.com/gx/en/issues/cybersecurity/cyber-threat-intelligence/old-cat-new-tricks.html

An analysis of Charming Kitten’s new tools and OPSEC errors

Copy link

By Krystle Reid

Executive summary

Yellow Garuda (similar to Charming Kitten, PHOSPHORUS, UNC788) is a threat actor likely to have been active since at least 2012. It is possibly one of the most active and persistent Iran-based threat actors over the last decade and is known primarily for spoofing log-in pages of legitimate webmail services to collect credentials from its targets. The threat actor also has a history of operational security (OPSEC) errors resulting in disclosure of its tools, techniques and procedures (TTPs), including the addition of Android malware to its expanding toolset.

OPSEC mistakes associated with Yellow Garuda operations in late 2021 resulted in the discovery of new tool used to enumerate data from targeted Telegram accounts. We also identified an alias tied to early Iran-based operations and a surveillance report likely written by a Yellow Garuda operator. Additionally, PwC analysts have observed the threat actor’s use of macro-enabled template files as recently as March 2022, a new TTP not previously associated with Yellow Garuda.

Telegram ‘grabber’ tool

Through our regular scanning for Yellow Garuda infrastructure, PwC analysts identified an open directory located at 138.201.145[.]183 containing several compressed archives associated with late 2021 Yellow Garuda activity.

Each of the RAR archives contained a copy of a tool named NewTelegram.LocalGrabber.Sqlite.UI.Win.exe, together with the tool’s component parts, and exfiltrated victim data. In total there were seven sets of victim data on the server, six of which were outputs of the Telegram ‘grabber’ tool, and one of which was almost certainly the result of data exfiltrated by mobile malware. Although it is unclear what malware was used, we note that the type of data captured is in line with the capabilities of PINEFLOWER, an Android malware previously attributed to Yellow Garuda[1].

These archives had filenames referencing Solar Hijri calendar dates indicating that the activity took place between 7th September and 11th October 2021, when converted to the Gregorian calendar. The activity suggests domestic targeting as all victim mobile numbers contained the Iranian country code and Farsi was the main language seen in victim databases (as part of Telegram group names or in exfiltrated messages). From the data exfiltrated, it was also apparent that some of victims were associates of each other, where two pairs of victims were contacts of another victim on Telegram. We also observed that two of the victims likely had links to the Iranian music industry.

Figure 1 - Contents of 138.201.145[.]183

The Telegram grabber tool is written in C++ and uses the open source Telegram Database Library (TDLib), a cross-platform Telegram client typically used to create custom apps for the platform[2]. It has been designed to exfiltrate information from a victim’s Telegram account. This includes messages and associated media, group memberships and contact data.

SHA-256	7709a06467b8a10ccfeed72072a0985e4e459206339adaea3afb0169bace024e
Filename	NewTelegram.LocalGrabber.Sqlite.UI.Win.exe
File type	Win32 EXE
File size	5,423,104 bytes
Compilation timestamp	2062-01-30 02:30:48

In order to access the victim’s account, the threat actor needs to enter a login code which is issued by Telegram as part of its authentication process. The code is sent either to the victim’s Telegram account, or via SMS to the victim’s phone[3]. This means that the threat actor needs to have access to a victim’s active Telegram session, either via a phone or desktop, or otherwise be able to access their SMS messages, for example via mobile malware. As can be seen in Figure 2, the victim’s phone number is required upon opening the tool in order to send the authentication code.

Figure 2 – GUI to enter victim’s phone number

If the victim has enabled two-step verification, an additional password is needed. Where this is unknown, it can be reset using a recovery email if one has been previously setup. The tool has options to view the password hint and send an access code via the victim’s recovery email address, which the threat actor would need to access in order to proceed. The existence of this option indicates that the threat actor, at least in some cases, is likely to have access to the victim’s email account. This aligns with Yellow Garuda’s known tactics, which include extensive credential harvesting via dedicated phishing sites.

Once authenticated, the operator is presented with multiple options to choose the type of data to download. This includes the ability to select a date range for the download of the different types of Telegram chat messages. For groups, the tool attempts to grab details on the participants as well as whether or not the victim is an administrator. The tool is also able to download data relating to the victim’s profile and their contacts, including their names, phone numbers, usernames and profile pictures.

Figure 3 – Metadata accessed for each contact

The exfiltrated data is stored within a SQLite database and also in JSON format. For attachments sent or received through chats, there are options to choose specific file formats to download. These pertain to common video, audio, document, binary and compressed file extensions. In addition to being able to exfiltrate data, the threat actor also has the ability to delete messages from the victim’s account.

We found 15 additional samples of the tool on an online multi-antivirus scanner which share the same filename (NewTelegram.LocalGrabber.Sqlite.UI.Win.exe) and TypeLib ID (7bb2c20c-740e-498b-8dd6-9c2ff8ad9572) as the sample we analysed. The TypeLib ID is a unique GUID created by Visual Studio when a new project is created[4], thus indicating that all of the samples originated from the same project. The additional samples were all uploaded within a 31 day window between January and February 2021 and contained similar functionality to the one we analysed. The main difference was the presence of a web request function that appeared to be used in a testing capacity. Given the clustered times of submission and the presence of the web request test method, we assess it is likely these samples were submitted by the threat actor itself in a testing capacity.

Insights into Yellow Garuda’s operations

The following Microsoft Word document was found in the directory corresponding to one of the victims whose data was highly likely exfiltrated via mobile malware and not through the Telegram ‘grabber’ tool. Its filename translates to ‘01Report’ and its contents detail the status of the surveillance on that victim.

Filename	01گزارش.docx
SHA-256	36c12ff1b62f4579d64926f5a26c4a1806235859a8f71c8754d5b257716be538
File type	DOCX
Author	ll_invisible_ll
Last modified user	ll_invisible_ll
Creation date	2021-10-11 06:48:00
Last modifed date	2021-10-12 12:54:00
File size	13,736 bytes

Figure 4 – Threat actor report (left) and translation from Farsi to English (right)

The report gives us insight into the threat actor's specific data collection objectives. It references the surveillance of audio and video conversations of the victim's phone and confirms the victim's name, national identity number, mobile number and phone model. It indicates the surveillance was completed on 11th October 2021 and is being sent for review by the 'relevant expert' on the orders of the manager of '2000' and '2300', numbers which could represent individual operators or departments within Yellow Garuda's operations. The report ends with a warning that if the surveillance is detected, it will not be possible to re-access the phone.

These numbers align with additional observations from the output log files of the Telegram 'grabber' tool which contained local file paths likely belonging to the threat actor. An example is as follows:

Figure 5 – Example of file path found in output log files. The date converts to 22nd September 2021 in the Gregorian calendar

We observed values of 1500, 2700 and 3500 being used as part of the local directory structure. This indicates that at least three operators or teams may have contributed to the Telegram 'grabber' activity observed and a further two separate operators or teams worked on the victim referenced in the report.

A previously leaked organisational chart associated with Iran's Islamic Revolutionary Guard Corps (IRGC) shows individual departments with similar numerical referencing (Figure 6)[5]. From the English translation in Table 1, we can see that several of the observed values are present, overlapping with departments related to cyber, security and counterintelligence. Although we are unable to independently verify the validity of this chart, the overlap in naming convention, and our understanding that Yellow Garuda is likely associated with the IRGC[6], aligns with our assessment that these are operator/team names.

The author name of the threat actor report, “Il_invisible_Il” is fairly unique and gives us insight into a potential individual operator. This alias was also in use between 2010 and 2016 on the Ashiyane forum, a now defunct Iranian hacking forum originally started by the Ashiyane Digital Security Team. The Ashiyane Digital Security Team has previously been linked to IRGC activity[7] and several of its members appeared in a US Department of Justice (DOJ) indictment for distributed denial of service (DDoS) attacks against organisations in the US financial sector and other US-based companies between 2011 and 2013[8].

Figure 6 – Diagram purportedly showing IRGC-related departments and leads (Farsi language) [9]

Name	Number
Head of the Intelligence Unit of the IRGC, Hossein Taeb	20
The Head of Department, General Yar Ali Sabzi	100
Deputy of Readiness and Support	200
Deputy of HR and Recruitment	300
Deputy of Plan, Program and Budget, General Gholipour	400
Deputy of Military Intelligence	500
Deputy of Psychological Operations, Haj Abdullah Mushfeq	600
Special cases	700
Deputy of Equipping, General Sadeghian	900
Deputy of Information Protection, Izadi	1000
Deputy of Inspector, General Karimi	1100
Deputy of Information Collection	1200
Thematic	1300
Deputy of Counterintelligence, General Taha	**1500**
Deputy of Arrest and Surveillance Operation, Detention Center, General Qajavand	1600
Legal Deputy, Dr. Mahdavi	1700
Deputy of Civil Engineering (Structure and Building)	1800
Representative of the Supreme Leader, Haj Qasimi and Mr. Elahi	1900

Deputy of Cyber, Hamid Naeem	**2000**
Deputy of Security (Counter-Terrorism and Fight against Armed Groups), General Nouhi	**2300**
Deputy of Crimes	2400
Centre of Documents	N/A

Table 1 – English translation of Figure 6 showing IRGC-related departments and leads; numbers overlapping with those observed in our analysis have been highlighted in bold

Macro-enabled Word document templates

Between January and March 2022, we observed Yellow Garuda using Microsoft Word document droppers which use remote template injection to obtain and execute a malicious macro. This is the first time we have observed the threat actor deploying macros or using remote template injection as part of its attack sequence.

SHA-256	Filename
41b37de3256a5d1577bbed4a04a61bd7bc119258266d2b8f10a9bb7ae7c0d4ec	Turkey_inj.docx[10]
725bdf594baa21edf1f3820b0daf393267066717832452598c617552a004e5da	Turkey.docx
01ca3f6dc5da4b98915dd8d6c19289dcb21b0691df1bb320650c3eb0db3f214c	Iran- Taliban relations.docx
57cc5e44fd84d98942c45799f367db78adc36a5424b7f8d9319346f945f64a72	NY.docx
a8c062846411d3fb8ceb0b2fe34389c4910a4887cd39552d30e6a03a02f4cc78	Details-of-Complaint.docx

Table 2 – Details of Microsoft Word document droppers (DOCX)

The document lures we observed covered a variety of themes including nuclear energy and weapons related to Turkey, US shipping ports and Iran's relationship to the Taliban and as such, we assess they were likely used to target a variety of unrelated entities. Many of these lures, used material sourced from legitimate English-language websites, including news and media sites. It is not unusual for threat actors to make use of current affairs as a means to catch the attention of potential victims and the themes are not necessarily indicative of specific targeting.

Figure 7 – Example of lure content

The initial Microsoft Word document (DOCX) is hosted on a third party service such as Dropbox or Amazon Web Services (AWS). Yellow Garuda is known to extensively employ social engineering as part of its attacks, therefore it is highly likely phishing was used to coerce a potential victim to download and open the document.

Once opened, a form of remote template injection takes place where the document reaches out to a URL to download a file with a DOTM extension (a macro-enabled template file). The URL is specified within the relationship component word/_rels/settings.xml.rels of the initial document as indicated by the ‘Target’ value in Figure 8. The documents we analysed reached out to files hosted on either Microsoft OneDrive or on dedicated threat actor-controlled infrastructure, as can be seen below.

Figure 8 – URL visible in word/_rels/settings.xml.rels

We were able to access several examples of this second stage macro-enabled template file. These differed in functionality and form, however they all maintained persistence by replacing the victim’s default Microsoft Word template, meaning that the malicious template (and macro) will open whenever Microsoft Word is opened by the victim.

SHA-256	Filename
c45bffb5fe7056075b966608e6b6bf82102f722b5c5d8a9c55631e155819d995	DocTemplate.dotm
dd28806d63f628dbc670caaa67379da368e62fa9edfbdfd37d3a91354df08e1c	DocTemplate.dotm
c0d5043b57a96ec00debd3f24e09612bcbc38a7fb5255ff905411459e70a6bb4	Details.dotm
28de2ccff30a4f198670b66b6f9a0ce5f5f9b7f889c2f5e6a4e365dea1c89d53	Arabic.dotm

Table 3 – Details of malicious template files (DOTM)

In some cases[11], we observed the macro code creating a reverse shell using code almost identical to that found in open source on a GitHub repository[12]. In other cases[13], the template files were password protected meaning that the victim is required to specify the password in order for the attack sequence to proceed. This would need to be passed over to the victim via a phishing email or some other form of social engineering. The template files also contained RC4-encrypted strings (both within the macro and lure document) for which the decryption key needed to be obtained as the response to a HTTP GET request to an Amazon S3 bucket. These steps were likely designed to thwart analysis attempts if the password cannot be obtained to open the document, or the infrastructure hosting the decryption key is no longer active.

Files dropped by the macros shared similar filenames to recent Yellow Garuda activity observed by Check Point[14]. The PowerShell backdoor known as CharmPower was observed to read data from a file called ni.txt, located in %AppData%, whose contents are sent to the command and control server along with basic information about the victim's machine. This aligns with our observations that ni.txt is used to house a hardcoded identifier and could indicate that a version of CharmPower is deployed at a later stage of the attack sequence.

Conclusion

Over the past year, we have seen Yellow Garuda continue to add tools to its arsenal. In its use of macro-enabled template files, we can see that the threat actor has made efforts to stage various parts of the infection chain remotely, disrupting analysis efforts where these are not accessible. The threat actor has also continued to make OPSEC mistakes exposing its tools and targeting through open servers. The Telegram 'grabber' tool we observed appears to be a tool that the threat actor has had access to since at least January 2021, and used against domestic targets to obtain specific access to Telegram messages and contacts alongside mobile malware.

The threat actor's operational report has given us further insight into its analysis process, indicating that there is an internal structure to its operations denoted by numerical call signs. It also highlights the alias of an individual which has previously been linked to Iran-based activity over several years.

MITRE ATT&CK

More detailed information on each of the techniques used in this blog, along with mitigations, can be found on the following MITRE pages:

Valid Accounts - https://attack.mitre.org/techniques/T1078/

Two-Factor Authentication Interception - https://attack.mitre.org/techniques/T1111/

Obfuscated Files or Information - https://attack.mitre.org/techniques/T1027/

System Information Discovery - https://attack.mitre.org/techniques/T1082/

System Network Configuration Discovery - https://attack.mitre.org/techniques/T1016/

Data Staged - https://attack.mitre.org/techniques/T1074/

Exfiltration Over Web Service - https://attack.mitre.org/techniques/T1567/

Acquire Infrastructure: Web Services - https://attack.mitre.org/techniques/T1583/006/

Acquire Infrastructure: Domains - https://attack.mitre.org/techniques/T1583/001/

Phishing: Spearphishing Link - https://attack.mitre.org/techniques/T1566/002/

Indicators of Compromise

Telegram 'grabber' tool:

Indicator	Type
7709a06467b8a10ccfeed72072a0985e4e459206339adaea3afb0169bace024e	SHA-256
f09fa790f8b3bf59f44093ae18e8c9ec95b54fb8dab5039e9bfd09b12b815950	SHA-256
6710d037801471826817596fa71637eecda4f58cddf47bbb48b3984b21582721	SHA-256
141ae6d29118b099d5ef8ee0daa7a4714447d5aa13ce43563e21900014f1db7d	SHA-256
ada1e14da19338f2fa009254a993c6b6607e9a328499c3a762d6652ca8edee5e	SHA-256
49218f19e3dc89ab2698f9e23f37d16a97b410de91226bb24e65c8392b74de93	SHA-256
4cddb6a4fbf8771ee3180b974fc12c8261880a213a4bf36b1e910e1c1df847cf	SHA-256
5987f958d758866ccea33437c53276382f9c362fc33e81d342b616dc70aeb78f	SHA-256
7ea6cb74238d3f0099d4b9c42dd7301b9fb903b62f1f2e06ef73ade533691a69	SHA-256
6e4e195c2d60aec5a75f287f2b27ade3204390ace9ad4dec07753234fb148b57	SHA-256
6b84eebded654d29b63f931a28e5fc4318aaf32604d1ad2f14e4a87b7a499206	SHA-256
f1651ffda0d45e6c37cd31c0ed83d9bd08c33acbd3647cbdd8b22b804ce8d6a3	SHA-256
009df256bce5971edaab72c19c4ebcc9296e203a2ef447557c0796d86217d1d3	SHA-256
5a9b1bf53e47cbecf41259f31d06f86dcf62b7858debd680c0a232de3577669a	SHA-256
4f85a533e6d25fb281639f9fb4b4f817faab2b291a7835c267f29c27728247f9	SHA-256
435f61ad26b729e1d7813454ff8279c52ebd928a3d1dd824cb9267189991565d	SHA-256
a81d2c633e938a04f486dea3b245e87dc498bc02	SHA-1
9f9a5e7c24f8f2ab030ce875736d80e541156003	SHA-1

85f1e02cb5f5c38b848c282187c3ceee7d544e13	SHA-1
b3adc3d81853185f65dbd278fbba7f795e4a3259	SHA-1
914a8da21feaab56fecbdc997710566775850617	SHA-1
affe20def567eb63447f2a3aad3927d52384db59	SHA-1
26ed903a997d8f9dfee10435e8930a9b24bd46f9	SHA-1
2b5056c31ca2a54e6bccc1912eee522dcf16cd94	SHA-1
71028a08ec0d64dff36cf5405997501278b949f9	SHA-1
78b4ba41d2de822061d1f3e0c43d13d564f10871	SHA-1
b785169c5fbaff8e205d6d58783706fc07208d59	SHA-1
48b110b088d4fd8381990dbd6cbb23abeb87b422	SHA-1
6df60e871d14996c4826a8c2355d64d3aabbfab6	SHA-1
82a0d684a1e144a7f9f874e652597155bb12ae92	SHA-1
a8e7784df801cea9cb6278762437314bb42d1966	SHA-1
72c4fe68520c0307367b0865b29215d1fc6e2c32	SHA-1
1d64ddd5a2c0fae5817235ab9ddf334f	MD5
e66136da3bb11795da64f038ec4610b8	MD5
eb51402e73a86800cdce3a50c9c804fe	MD5
f7b0da0dca597f3e61f53000814f8148	MD5
96be653e085046ed518ad3ce48fc4190	MD5

d16f4bf877445e9fca422dc736db64cf	MD5
949cc35be1b366eaad94ea03cf862d6e	MD5
88fd6260d23f01213d3e2abee74db4a2	MD5
5816f687ce49588aae2584bb5e9f652f	MD5
12a172b74d0c080217bf0b883c109a6b	MD5
b78483179f85d3c8e23733ebd114e10e	MD5
bddebaea4bf45f6b464d68a7b8e07b92	MD5
aba932b87072f479445a323b183cc29b	MD5
381bb58655a194e75763fb01a36e5c7b	MD5
b8045bebc39a8fff666803a5163173d8	MD5
6a1dca07dafd2eebd99aba7c31ace928	MD5
NewTelegram.LocalGrabber.Sqlite.UI.Win.exe	Filename
138.201.145[.]183	IPv4 address

Macro-enabled template activity:

Indicator	**Type**
57cc5e44fd84d98942c45799f367db78adc36a5424b7f8d9319346f945f64a72	SHA-256
725bdf594baa21edf1f3820b0daf393267066717832452598c617552a004e5da	SHA-256
41b37de3256a5d1577bbed4a04a61bd7bc119258266d2b8f10a9bb7ae7c0d4ec	SHA-256

01ca3f6dc5da4b98915dd8d6c19289dcb21b0691df1bb320650c3eb0db3f214c	SHA-256
c45bffb5fe7056075b966608e6b6bf82102f722b5c5d8a9c55631e155819d995	SHA-256
dd28806d63f628dbc670caaa67379da368e62fa9edfbdfd37d3a91354df08e1c	SHA-256
a8c062846411d3fb8ceb0b2fe34389c4910a4887cd39552d30e6a03a02f4cc78	SHA-256
c0d5043b57a96ec00debd3f24e09612bcbc38a7fb5255ff905411459e70a6bb4	SHA-256
28de2ccff30a4f198670b66b6f9a0ce5f5f9b7f889c2f5e6a4e365dea1c89d53	SHA-256
b98a24144067ec3605e84158e12d6498222295ae	SHA-1
f39c5689887f5b94741e285cd867e1475111499e	SHA-1
5c0e8bd70e2dd49d45937ccc3f38de61d356384c	SHA-1
40dc7101e1991672b5f60523e69ed5787a9dc4fa	SHA-1
cc9f460e593522e57b66fed9a34d3ba332391165	SHA-1
930e4757740aaefd9cb567faf301816fbe37c1c3	SHA-1
e3712e3d818e63060e30aec2a6db3598cbf0db92	SHA-1
e45aeccb798f5cf6cb5d877821d1f4aa7f55cf6f	SHA-1
aba938bf8dc5445df3d5b77a42db4d6643db4383	SHA-1
45b50d42e8d827ca0373c12533211c33	MD5
55748b22a52823a3ccb5d8b106826cec	MD5

4ae177a37658c82adad3265ad3cce662	MD5
14c095de9da5fbba5548d9fea65c8b2d	MD5
db998d8182f4afd9f42bb289c508a1f3	MD5
c711036ef1805fea9dc2c8e633b961fd	MD5
b7bc6a853f160df2cc64371467ed866d	MD5
651d72776c0394693c25b1e3c9ec55d0	MD5
bdf188b3d0939ec837987b4936b19570	MD5
official-updates[.]info	Domain
office-updates[.]info	Domain
51.38.87[.]253	IPv4 address
hxxp://official-updates[.]info/office/Default.dotm	URL
hxxps://dl[.]dropboxusercontent[.]com/s/psmt483ybusajvy/Turkey.docx?dl=0	URL
hxxps://u1ndk6f4nf[.]execute-api[.]us-east-1[.]amazonaws[.]com/page/EdPEtAGapngkNtLLFCee	URL
hxxps://u1ndk6f4nf[.]execute-api[.]us-east-1[.]amazonaws[.]com/page/zhUezQeFqaDRmxWaHfVz	URL
hxxps://s3[.]amazonaws[.]com/2v63r9egi46/mvhg5dhdbsolshpq	URL
hxxps://s3[.]amazonaws[.]com/2v63r9egi46/hgn8fdsf512fsc5	URL
hxxps://drive[.]google[.]com/uc?export=download&id=13_PT71n8Ujl2lSTcQcyFJ4TNetI-wvDf&dID=1645099370036&linkName=Download%20File	URL

hxxp://office-updates[.]info/2022/Details.dotm	URL
hxxp://office-updates[.]info/static/admin/storage/Arabic.dotm	URL
hxxp://office-updates[.]info/static/admin/storage/Details.dotm	URL

Footnotes

[1] ‘UNC788: IRAN’S DECADE OF CREDENTIAL HARVESTING AND SURVEILLANCE OPERATIONS’, VB2021 localhost, https://vblocalhost.com/uploads/VB2021-Haeghebaert.pdf (October 2021)

[2] Telegram, ‘Telegram Database Library’, https://core.telegram.org/tdlib

[3] Telegram, ‘FAQ’, https://telegram.org/faq#login-and-sms

[4] ‘Using .NET GUIDs to help hunt for malware’, Virus Bulletin, https://www.virusbulletin.com/virusbulletin/2015/06/using-net-guids-help-hunt-malware/ (25th June 2015)

[5] ‘BLACK BOX|جعبه سیاه’, Telegram, https://t.me/jabeh_siah/4232 (26th February 2019)

[6] ‘BadBlood: TA453 Targets US and Israeli Medical Research Personnel in Credential Phishing Campaigns’, Proofpoint, https://www.proofpoint.com/uk/blog/threat-insight/badblood-ta453-targets-us-and-israeli-medical-research-personnel-credential (30th March 2021)

[7] ‘Decision 2011/235/CFSP’, European Union, https://eurlex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:267:0013:0018:EN:PDF (12th October 2011)

[8] ‘Indictment 834996’, US Department of Justice, https://www.justice.gov/opa/file/834996/download

[9] ‘BLACK BOX|جعبه سیاه’, Telegram, https://t.me/jabeh_siah/4232 (26th February 2019)

[10] It is likely this is not the original file name and it has been altered by the submitter.

[11] SHA-256: 28de2ccff30a4f198670b66b6f9a0ce5f5f9b7f889c2f5e6a4e365dea1c89d53 and c0d5043b57a96ec00debd3f24e09612bcbc38a7fb5255ff905411459e70a6bb4

[12] ‘JohnWoodman/VBA Macro Reverse Shell’, GitHub, https://github.com/JohnWoodman/VBA-Macro-Reverse-Shell/blob/main/VBA-Reverse-Shell.vba#L69 (13th February 2021)

[13] SHA-256: c45bffb5fe7056075b966608e6b6bf82102f722b5c5d8a9c55631e155819d995 and dd28806d63f628dbc670caaa67379da368e62fa9edfbdfd37d3a91354df08e1c

[14] ‘APT35 exploits Log4j vulnerability to distribute new modular PowerShell toolkit’, Check Point, https://research.checkpoint.com/2022/apt35-exploits-log4j-vulnerability-to-distribute-new-modular-powershell-toolkit/ (11th January 2022)

{{filterContent.facetedTitle}}

{{contentList.loadingText}}

{{item.videoDuration}}

{{item.title}}

{{item.thumbnailText}}

{{item.videoDuration}}

{{item.publishDate}}

{{item.title}}

{{item.thumbnailText}}

{{contentList.loadingText}}

{{contentList.loadMoreLabel}} {{contentList.viewAllLabel}}