

LATEST VIDEOS



TUNNELCRACK: TWO SERIOUS VULNERABILITIES IN VPNS DISCOVERED, HAD BEEN DORMANT SINCE 1996



HOW TO EASILY HACK TP-LINK ARCHER AX21 WI-FI ROUTER



US GOVT WANTS NEW LABEL ON SECURE IOT DEVICES OR WANTS TO DISCOURAGE USE OF CHINESE IOT GADGETS



24,649,096,027 (24.65 BILLION) ACCOUNT USERNAMES AND PASSWORDS HAVE BEEN LEAKED BY CYBER CRIMINALS TILL NOW IN 2022



HOW CHINESE APT HACKERS STOLE LOCKHEED MARTIN F-35 FIGHTER PLANE TO DEVELOP ITS OWN J-20 STEALTH FIGHTER AIRCRAFT [VIDEO]

[VIEW ALL](#)

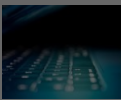
VULNERABILITIES



MASSIVE NVIDIA GPU EXPLOIT FOUND. HOW HACKERS CAN TAKE DOWN 35% OF AI SYSTEMS IN CLOUD!



BLAST-RADIUS ATTACK EXPLOITING CRITICAL RADIUS FLAW COULD COMPROMISE YOUR NETWORK



14 MILLION SERVERS VULNERABLE TO CRITICAL OPENSSSH BUG: BECOME REMOTE ADMIN WITH CVE-2024-6387



HOW SAFE IS YOUR TINYPROXY? STEP-BY-STEP GUIDE TO EXPLOITING TINYPROXY'S ZERO DAY VULNERABILITY



ETERNAL MALWARE: CVE-2024-3400 ROOTKITS PERSIST THROUGH PALO ALTO FIREWALLS UPDATES AND RESETS

[VIEW ALL](#)

## ATTACKERS LEVERAGE EXCEL, POWERSHELL AND DNS IN LATEST NON-MALWARE ATTACK

Share this...



Increasingly, cyberattackers have been leveraging “non-malware” attack methods to target vulnerable organizations. Recently, the Carbon Black Threat Research Team was alerted about such an attack by a partner's incident response (IR) team. The attack ultimately compromised accounts and stole research and intellectual property.

In this specific attack, a malicious Excel document was used to create a PowerShell script, which then used the Domain Name System (DNS) to communicate with an Internet Command and Control (C2) server.

This attack was first reported by a partner of ours, who had been alerted by themselves.

1. How the attacker gained access to the organization's network (e.g., via a phishing email, etc);
2. How the attacker used the network to reach the target (e.g., via a proxy, etc);
3. How the attacker used the target's network to reach the target (e.g., via a proxy, etc).

To answer these questions, we analyzed the attack logs and the network traffic. The malicious Excel document was the key to the attack.

### ATTACK

The Excel document was sent to a small group of approximately 15 engineers. The document's file name was related to the engineers' duties and included the organization's name. At least one recipient opened the document and initiated the attack sequence, which resulted in a PowerShell script communicating with a C2 using a DNS channel.

Following the initial compromise, the PowerShell script was later updated with a different C2. The attackers eventually moved laterally to different servers and continued to use PowerShell as their main tool for conducting their day-to-day activities. The attackers also used this script to download additional tools, which were initially utilized to dump credentials (wce.exe), move laterally (psexec.exe), and conduct network reconnaissance.

(Due to sensitivity, the Excel file detailed below is not the exact file that was submitted, but is almost identical. Only some minor metadata, variable names, and the C2 domain are different.)

### ATTACK TECHNICAL ANALYSIS

#### FIRST STAGE

The metadata for the submitted Excel document is listed in the table below.

File Name	: trill.xls
File Size	: 131,072 bytes
MD5	: 2a462cdbaee3b0340bc6298057d83240
SHA1	: 256e736d7dcb670c6a510b5a7d60a53572acc1e7
SHA256	: 0dc86ad65c90cfc84253c4d7605911aba93b599b1bbd422ce8f597f3fffd59009

Open-source tools (oletools and oledump) were used to initially examine this document. Oletools was used to provide the metadata associated with this Excel file, which provides, among other things, the internal Microsoft dates associated with the file's creation and last saved times.

TUTORIALS



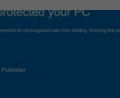
5 TECHNIQUES HACKERS USE TO JAILBREAK CHATGPT, GEMINI, AND COPILOT AI SYSTEMS



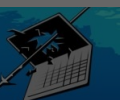
THIS HACKER TOOLKIT CAN BREACH ANY AIR-GAPPED SYSTEM – HERE'S HOW IT WORKS



HACKING PAGERS TO EXPLOSIONS: ISRAEL'S COVERT CYBER-PHYSICAL SABOTAGE OPERATION AGAINST HEZBOLLAH!



FIVE TECHNIQUES FOR BYPASSING MICROSOFT SMARTSCREEN AND SMART APP CONTROL (SAC) TO RUN MALWARE IN WINDOWS



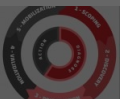
HOW MILLIONS OF PHISHING EMAILS WERE SENT FROM TRUSTED DOMAINS: ECHOSPOOFING EXPLAINED



HOW TO IMPLEMENT PRINCIPLE OF LEAST PRIVILEGE(CLOUD SECURITY) IN AWS, AZURE, AND GCP CLOUD



THE 11 ESSENTIAL FALCO CLOUD SECURITY RULES FOR SECURING CONTAINERIZED APPLICATIONS AT NO COST



HACK-PROOF YOUR CLOUD: THE STEP-BY-STEP CONTINUOUS THREAT EXPOSURE MANAGEMENT CTEM STRATEGY FOR AWS & AZURE



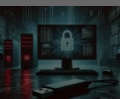
WEB-BASED PLC MALWARE: A NEW TECHNIQUE TO HACK INDUSTRIAL CONTROL SYSTEMS



THE API SECURITY CHECKLIST: 10 STRATEGIES TO KEEP API INTEGRATIONS SECURE

[VIEW ALL](#)

MALWARE



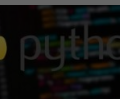
THIS HACKER TOOLKIT CAN BREACH ANY AIR-GAPPED SYSTEM – HERE'S HOW IT WORKS



HACKERS' GUIDE TO ROGUE VM DEPLOYMENT: LESSONS FROM THE MITRE HACK



ETERNAL MALWARE: CVE-2024-3400 ROOTKITS PERSIST THROUGH PALO ALTO FIREWALLS UPDATES AND RESETS



MAJOR PYTHON INFRASTRUCTURE BREACH – OVER 170K USERS COMPROMISED. HOW SAFE IS YOUR CODE?



HOW TO EXPLOIT WINDOWS DEFENDER ANTIVIRUS TO INFECT A DEVICE WITH MALWARE

[VIEW ALL](#)



Oledump was used to detail the streams present in the Excel document. What should be noted here is that two streams (7 and 8) have the letter M (or m) in their entry, meaning the streams contain macros. Stream 8 is highlighted in red and detailed below. Also of interest, highlighted in yellow, is stream 4, which is considerably larger than any of the other streams.





Both of the streams containing macros were dumped and decompressed for further analysis (oledump.py -s 8 -v [path to file])[i]. Stream 8 contained the relevant and malicious macro, and is displayed in the image below. The first thing to observe from the code is the area highlighted in red. The function at lines 8 and 9 will run when the workbook (the document itself) is opened, which serves to call the main function in script (located at line 13).



Welcome

This site asks for consent to use your data

Personalised advertising and content, advertising and content measurement, audience research and services development

Store and/or access information on a device

Your personal data will be processed and information from your device (cookies, unique identifiers, and other device data) may be stored by, accessed by and shared with [134 TCF vendor\(s\)](#) and [63 ad partner\(s\)](#), or used specifically by this site or app.

Some vendors may process your personal data on the basis of legitimate interest, which you can object to by managing your options below. Look for a link at the bottom of this page to manage or withdraw consent in privacy and cookie settings.

The next area of importance is the area highlighted in yellow. This series of lightly obfuscated commands sets string variables and objects used later in the script. The section highlighted in green is responsible for extracting the malicious PowerShell script from the Excel document. Line 19 sets the target file, which varies per attacker campaign, in the user's temporary folder[iii]. Line 20 will then get the contents of the cell located in worksheet 1 at row 37 and column 27 and write that data to the target file (U1848931.TMP). The content of that file is the malicious PowerShell script and is detailed later in this report.

The area highlighted in blue is the second part of this attack and is responsible for executing the PowerShell script and entrenching it on the system. Again, at line 24 the contents of a cell are located, specifically in the worksheet at row 40 and column 27 that is loaded into a buffer.

Line 25 will then concatenate an obfuscated string, and the buffer is appended to that string in line 26. These two lines will be used in line 27 to call PowerShell, which invokes a script that is encoded[iv] in the buffer.

To manually extract the contents that are being used in the different buffers, stream 4 will first need to be dumped. This stream contains numerous records that track cell, row, and column states, as well as the data being stored in these locations.

Reading column and row tables can be tedious (there are open source projects that can parse this data and, often times, while reviewing the data present in the stream, the suspect data is typically obvious.)

In this particular stream, it is fairly obvious that there is Base64 encoded data beginning at offset 0x124BA, highlighted in red below. The word (two-byte) value, highlighted in yellow below (before the data starts), is the length of that data 0x6CFC (or 27,900 decimal) in bytes.

Now that the length of the data is known, this portion can be extracted and Base64 decoded. It should be noted that cells in a worksheet can only hold a certain amount of data, specifically 8,228 bytes; anything larger requires the use of continue blocks, which tracks additional data and works similar to data runs in the NTFS file system.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00012490	00	00	08	40	00	1E	F1	10	00	00	00	0D	00	00	08	0C @ ñ
000124A0	00	00	08	17	00	00	08	F7	00	00	10	FC	00	20	20	27 ÷
000124B0	00	00	00	09	00	00	00	FC	6C	00	4A	41	42	7A	41	47
000124C0	4D	41	63	67	42	70	41	48	41	41	64	41	42	6B	41	47
000124D0	6B	41	63	67	41	67	41	44	30	41	49	41	41	69	41	43
000124E0	51	41	5A	51	42	75	41	48	59	41	4F	67	42	31	41	48
000124F0	4D	41	5A	51	42	79	41	48	41	41	63	67	42	76	41	47
00012500	59	41	61	51	42	73	41	47	55	41	58	41	42	68	41	48

The table below is an example of one of the continue blocks. This block is typically 5 bytes in length. The first byte (highlighted in yellow) is the BIFF record number or “type.” The last three bytes highlighted in blue represent the length of data, in bytes, following the record, which in this case, is 0x002020 (or 8224 decimal).

Understanding continue blocks will allow you to properly extract the correct amount of data. To calculate the size with the continuation blocks, the provided size (27,900 in this example) needs to be divided by maximum length of each block size (8228), the quotient (dropping the remainder, in this case the quotient is 3 can then be multiplied by 5 (the size of the record) resulting in the number of bytes to add to the original provide size (27900 + (3 x 5) = 27915).

The result will be the size of the data with the continue blocks in place. After extracting the relevant data, the continuation blocks should be removed so that actual data, in this case Base64 encoded string, can be handled properly. This specific data is the buffer that was saved by as U1848931.TMP.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00016000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000160F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000161F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016200	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016210	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016250	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016270	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016290	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000162F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016300	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016310	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016330	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016340	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016350	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016360	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016370	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016380	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016390	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000163F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016400	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016410	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016420	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000164F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016500	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016510	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00016520	00	00	00													

The next section from the above image, highlighted in red, is responsible for reading in the buffer that was written to disk as U1848931.TMP. This file will be Base64 decoded and the ID placeholder (line 16) will be replaced with a pseudorandom number. The buffer will then be saved to disk as mc.ps1 in the same directory as the VB script detailed above.

The last section from the above image, displayed in yellow, is responsible for creating the entrenchment. The combination of the lines 19 and 20 will create a scheduled task named “GoogleServiceUpdate” that will run every two minutes and execute the L69742.vbs file, which is displayed below.

### FINAL PAYLOAD

The mc.ps1 PowerShell script is a remote-access tool, with the following metadata:

File Name	: mc.ps1
File Size	: 20,924 bytes
MD5	: bebb6238a9b858386cef07328f4470e3
SHA1	: ba9e9c8d36b88b6a8cbe3fa116bfb8c8e1c6c9ad
SHA256	: 1d5d31994

The script is a remote-access tool, with the following metadata: The script is a remote-access tool, with the following metadata:

The image shows a screenshot of a website with a consent dialog. The dialog has a title "Welcome" and a main heading "This site asks for consent to use your data". It lists two categories of data usage: "Personalised advertising and content, advertising and content measurement, audience research and services development" and "Store and/or access information on a device". Below the list, it states: "Your personal data will be processed and information from your device (cookies, unique identifiers, and other device data) may be stored by, accessed by and shared with 134 TCF vendor(s) and 63 ad partner(s), or used specifically by this site or app." At the bottom, it says: "Some vendors may process your personal data on the basis of legitimate interest, which you can object to by managing your options below. Look for a link at the bottom of this page to manage or withdraw consent in privacy and cookie settings."

The script's main functions, which have been collapsed and highlighted in yellow, are responsible for crafting the DNS request and communicating with the C2. These will be detailed more in the next section. The other important function, highlighted in blue, is responsible for encoding the file upload portions of the DNS request. The file and batch-file download responses will be Base64 encoded, with a mapping of the characters “\_” and “-” instead of the traditional “+” and “/”. The output from batch scripts will be uploaded and processed encoded by the “base32data” function before being transmitted to the C2. This function, as its named, is an implementation of Base32. However, instead of using the standard alphabet, the code uses custom alphabet mapping, listed below. It should be noted that other variants used only Base64 (with and without custom alphabet mappings).

Standard mapping	- ABCDEFGHIJKLMNOPQRSTUVWXYZ234567
Custom mapping	- abcdefghijklmnopqrstuvwxyz012345

The script is fairly flat and will simply execute commands from top to bottom. In several of the samples reviewed, the download-files function was first executed, followed by the download-batch file function, and finally the upload-file function. Each function will make a request to see if there is a command to complete the specific task.


Depending on the response from the C2, the script will either proceed with the command or continue onto the next function. Due to the size limitations of data that can be transmitted in a DNS record query, the script will need to make numerous requests in

order to completely download or upload data. The DNS queries performed for each of these functions have their own identifier, listed in the table below.


Identifier	Notes
Download File	
rne_	Initial request for file download. If the C2 returns a response beginning with <i>OK</i> , then the next script will continue with the file download. If the C2 returns a response beginning with <i>NO</i> , then the script will continue to the download batch file function.
rd_	Secondary request to initiate the file download. The script will continue to make request and appending the responses to a buffer until the C2 returns a response beginning with <i>EOFEOF</i> . This data is Base64 encoded when transmitted.
Download Batch File	
bne_	Initial request for batch file download. If the C2 returns a response beginning with <i>OK</i> , then the next script will continue with the batch file download. If the C2 returns a response beginning with <i>NO</i> , then the script will continue to the upload file function.
bd_	Secondary request to initiate the batch file download. The script will continue to make request and appending the responses to a buffer until the C2 returns a response beginning with <i>EOFEOF</i> . The script will then append the .bat file extension and execute the file. The output of the batch files that are executed are then written to the upload folder with the original name of the batch file prepended with <i>_bat</i> . This data is Base64 encoded when transmitted.
Upload File	
u_	Any files in the upload folder, which are typically output from batch files, are uploaded to the C2. The script will upload the files in segments that are <i>31</i> bytes in length (this parameter is set in one of the initial variables and can be altered). The C2 will continue to receive the DNS request from the script until the request contains <i>EOFEOF</i> . This data is Base32 encoded when transmitted.

Welcome

This site asks for consent to use your data



Personalised advertising and content, advertising and content measurement, audience research and services development



Store and/or access information on a device

Your personal data will be processed and information from your device (cookies, unique identifiers, and other device data) may be stored by, accessed by and shared with [134 TCF vendor\(s\)](#) and [63 ad partner\(s\)](#), or used specifically by this site or app.

Some vendors may process your personal data on the basis of legitimate interest, which you can object to by managing your options below. Look for a link at the bottom of this page to manage or withdraw consent in privacy and cookie settings.

bd\_42306\_1\_-\_txt\_0\_314159.yjksdr1.tk

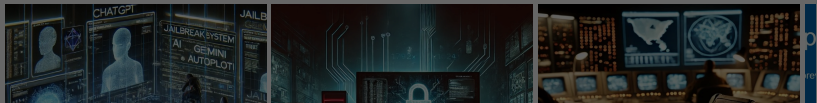
The response will be a Base64 (with a custom mapping of the symbols) encoded string, which will be decoded and written to disk with the original file name (in this case 1.txt). This process will continue with the script making another request and the writing the decoded response to the target file until the response contains *EOFEOF*.

Once the end of file marker has been received, the script will rename the file with a .bat extension and then execute the file. The output of the file will be written to the upload directory as, in this scenario, 1.txt-bat which is ultimately Base32 coded (using the custom alphabet) and transmitted to the C2 as a DNS request.

A python script was written to parse the traffic that is created by this malware, which is available to members of the Carbon Black User Exchange.

Source:https://www.carbonblack.com/

Share this...



5 TECHNIQUES HACKERS USE TO JAILBREAK CHATGPT, GEMINI, AND COPILOT-AI SYSTEMS

THIS HACKER TOOLKIT CAN BREACH ANY AIR-GAPPED SYSTEM – HERE'S HOW IT WORKS

HACKING PAGERS TO EXPLOSIONS: ISRAEL'S COVERT CYBER-PHYSICAL SABOTAGE OPERATION AGAINST HEZBOLLAH!

Alisa Esage G

Working as a cyber security solutions architect, Alisa focuses on application and network security. Before joining us she held a cyber security researcher positions within a variety of cyber security start-ups. She also experience in different industry domains like finance, healthcare and consumer products.

ON: MARCH 20, 2017 / IN: INCIDENTS, MALWARE, VULNERABILITIES / TAGGED: DOMAIN NAME SYSTEM (DNS), MALWARE, POWERSHELL

CYBER SECURITY CHANNEL

US GOVT WANTS NEW LABEL ON SECURE IOT DEVICES OR WANTS TO DISCOURAGE USE OF CHINESE IOT GADGETS

24,649,096,027 (24.65 BILLION) ACCOUNT USERNAMES AND PASSWORDS HAVE BEEN LEAKED BY CYBER CRIMINALS TILL NOW IN 2022

HOW CHINESE APT HACKERS STOLE LOCKHEED MARTIN F-35 FIGHTER PLANE TO DEVELOP ITS OWN J-20 STEALTH FIGHTER AIRCRAFT [VIDEO]

Welcome

This site asks for consent to use your data

Personalised advertising and content, advertising and content measurement, audience research and services development

Store and/or access information on a device

Your personal data will be processed and information from your device (cookies, unique identifiers, and other device data) may be stored by, accessed by and shared with [134 TCF vendor\(s\)](#) and [63 ad partner\(s\)](#), or used specifically by this site or app.

Some vendors may process your personal data on the basis of legitimate interest, which you can object to by managing your options below. Look for a link at the bottom of this page to manage or withdraw consent in privacy and cookie settings.

Page 6 of 6