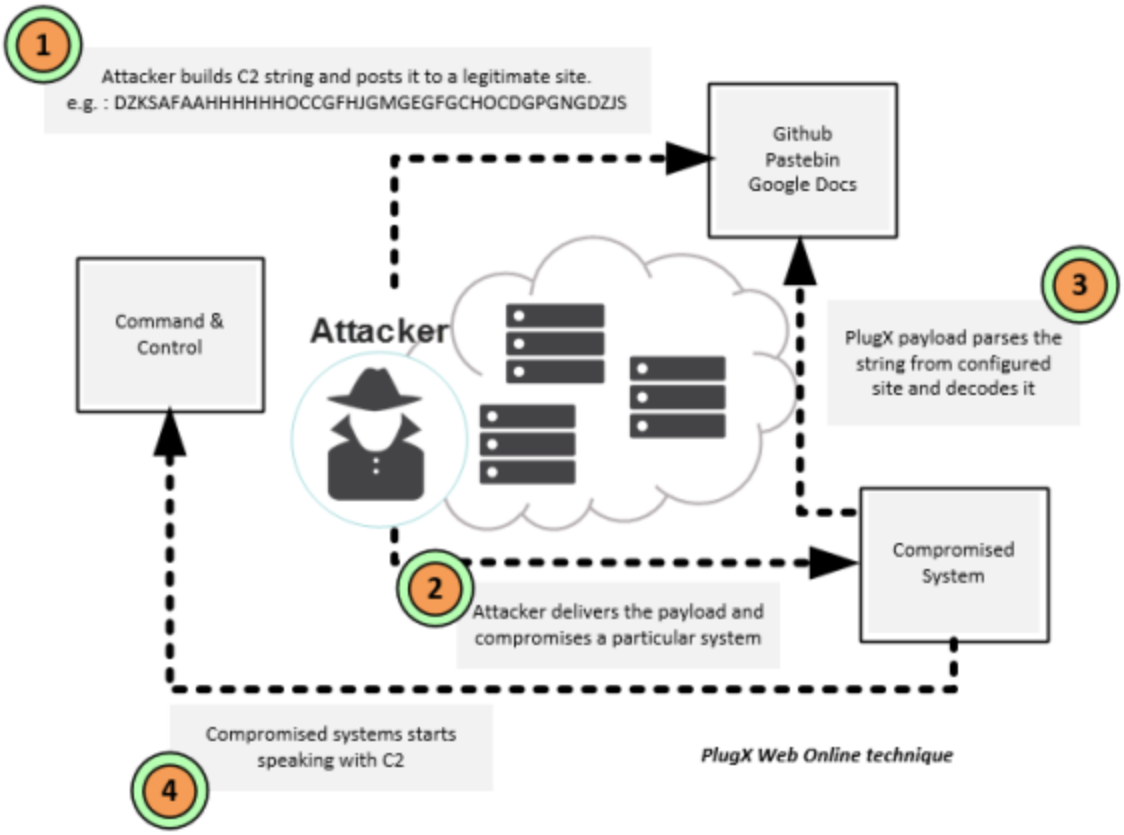
Malware Analysis – PlugX – Part 2

countuponsecurity.com/2018/05/09/malware-analysis-plugx-part-2/

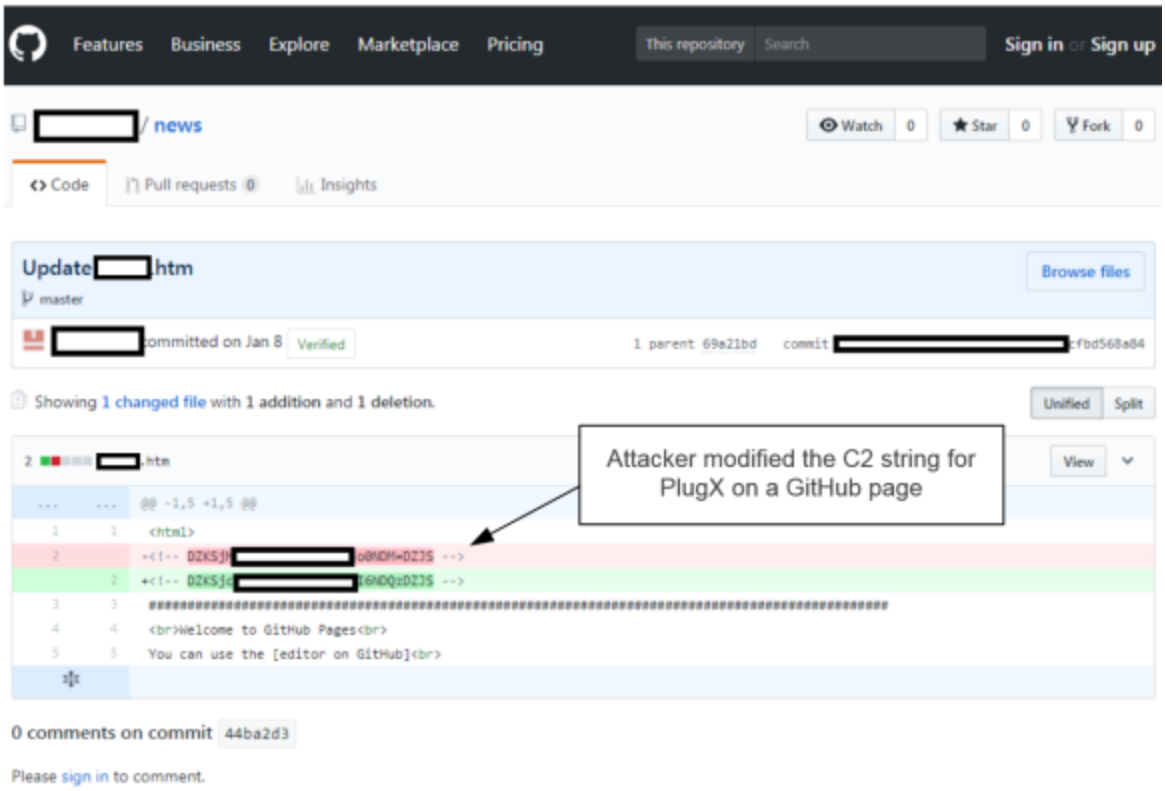
By Luis Rocha May 9, 2018

Following my previous article on PlugX, I would like to continue the analysis but now use the PlugX controller to mimic some of the steps that might be executed by an attacker. As you know the traditional steps of an attack lifecycle follow, normally, a predictable sequence of events i.e., Reconnaissance, initial compromise, establish foothold, escalate privileges, internal reconnaissance, move laterally, maintain persistence, complete mission. For sake of brevity I will skip most of the steps and will focus on the lateral movement.I will use the PlugX controller and C2 functionality to simulate an attacker that established a foothold inside an environment and obtained admin access to a workstation. Following that, the attacker moved laterally to a Windows Domain Controller. I will use the PlugX controller to accomplish this scenario and observe how an attacker would operate within a compromised environment.

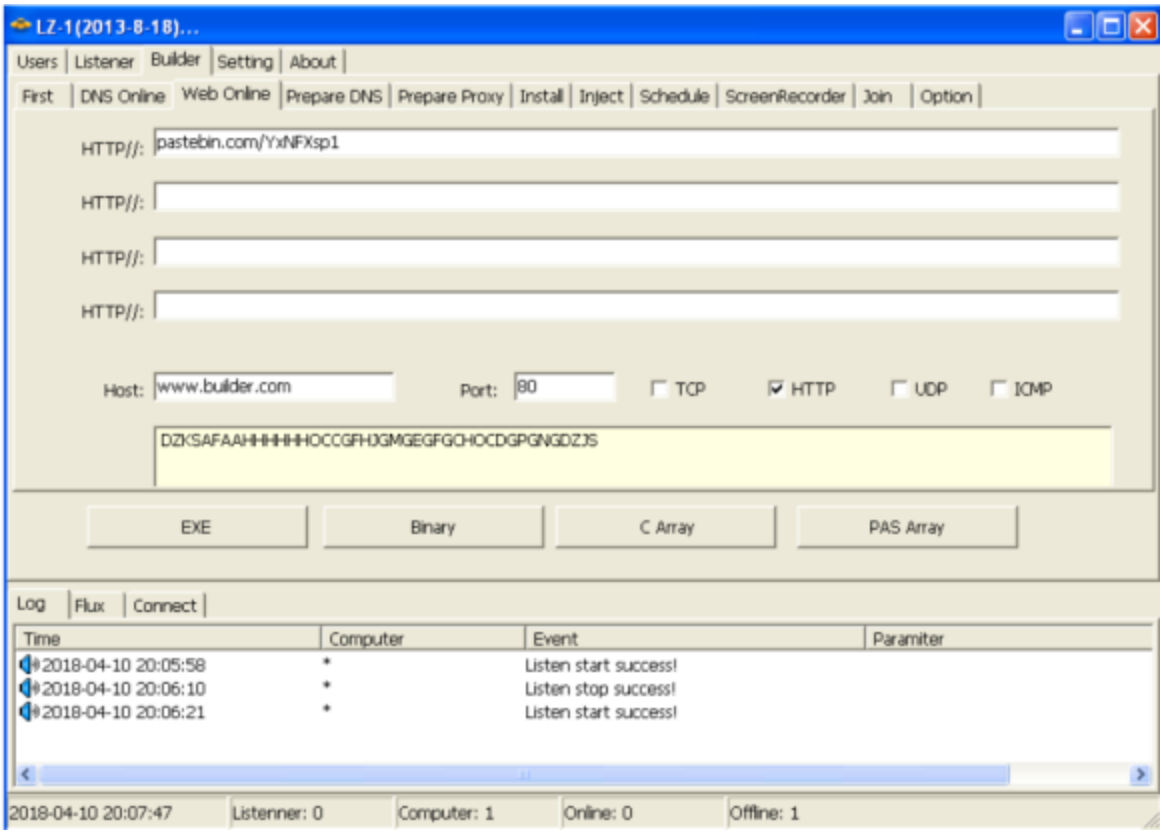
As we saw previously, the PlugX controller interface allows an operator to build payloads, set campaigns and define the preferred method for the compromised hosts to check-in and communicate with the controller. In the PlugX controller, English version from Q3 2013, an operator can build the payload using two techniques. One is using the “DNS Online” technique which allows the operator to define the C2 address e.g, an URL or IP address, that will be used by the payload to speak with the C2. The other method, is the “Web Online”, which allows the operator to tell the payload from where it should fetch the C2 address. This method allows the operator to have more control over the campaign. The following diagram illustrates how the “Web Online” technique works.



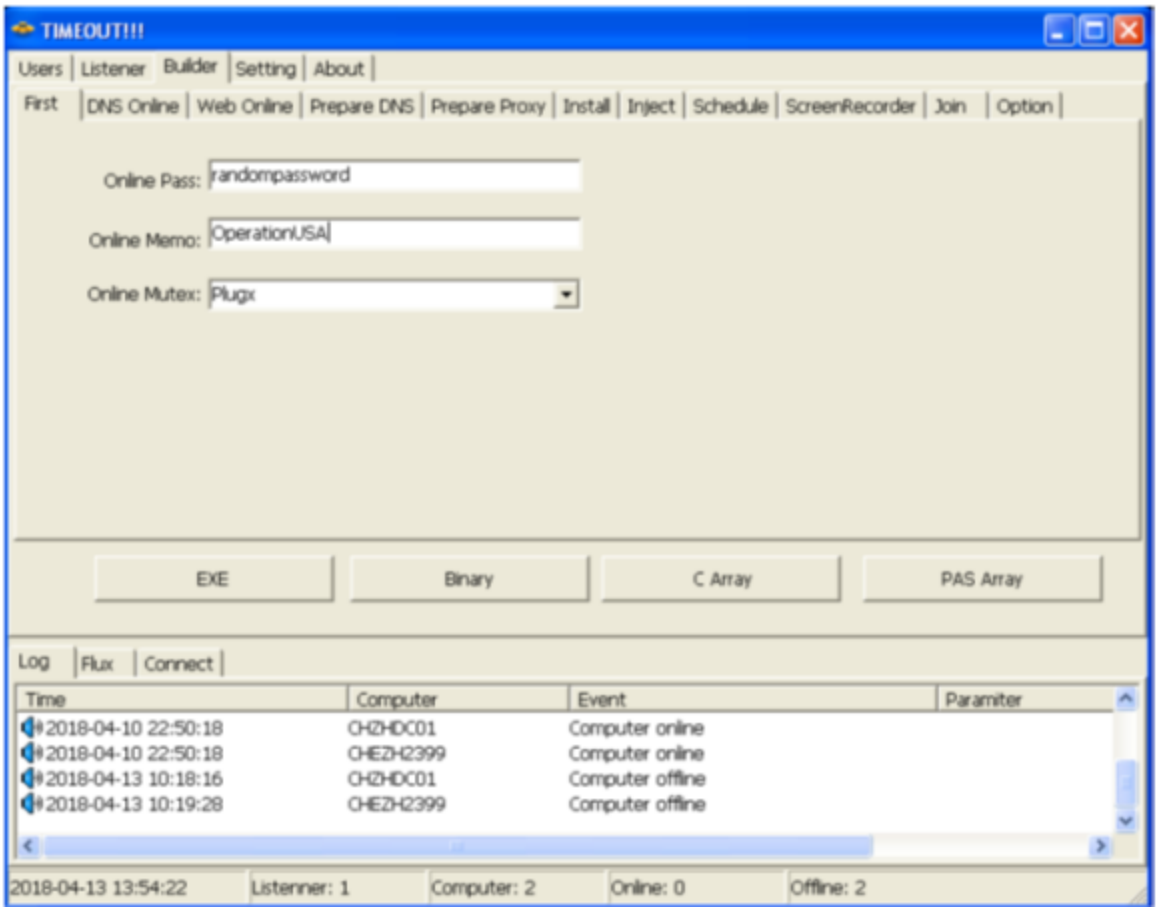
Why do I say this technique would allow an attacker to have more control? Consider the case that an organization was compromised by a threat actor that use this PlugX technique. In case the C2 is discovered, the impacted organization could block the IP or URL on the existing boundary security controls as a normal reaction to the concerns of having an attacker inside the network. However, the attacker could just change the C2 string and point it to a different system. In case the organization was not able to scope the incident and understand the TTP's (Tools, Tactics and Procedures) then the attacker would still maintain persistence in the environment. This is an example that when conducting incident response, among other things, you need to have visibility into the tools and techniques the attacker is using so you could properly scope the incident and perform effective and efficient containment and eradication steps. As an example of this technique, below is a print screen from a GitHub page that has been used by an unknown threat actor to leverage this method.



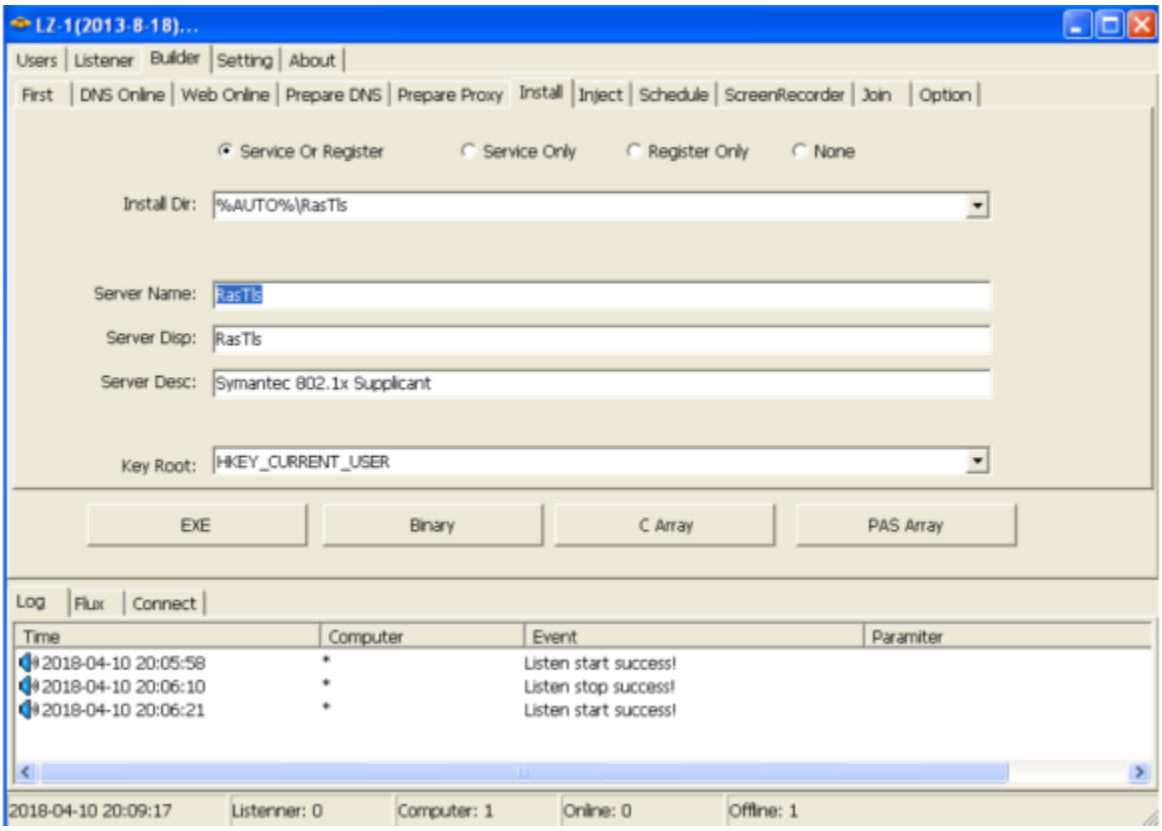
So, how to leverage this technique on the PlugX builder? The picture below shows how the operator could create a payload that uses the “Web Online” technique. The C2 address would be fetched from a specified site e.g. a Pastebin address, which on its turn would redirect the payload to the real C2 address. The string “DZKSAFAAHHHHHHOCCGFHJGMGEGFGCHOCDGPGNGDZJS” in this case is used to decode to the real C2 address which is “www.builder.com”. On the “PlugX: some uncovered points” article, Fabien Perigaud writes about how to decode this string. Palo Alto Unit42 gives another example of this technique on the “Paranoid PlugX” article. The article “Winnti Abuses GitHub for C&C Communications” from Cedric Pernet ilustrates an APT group leveraging this technique using GitHub.



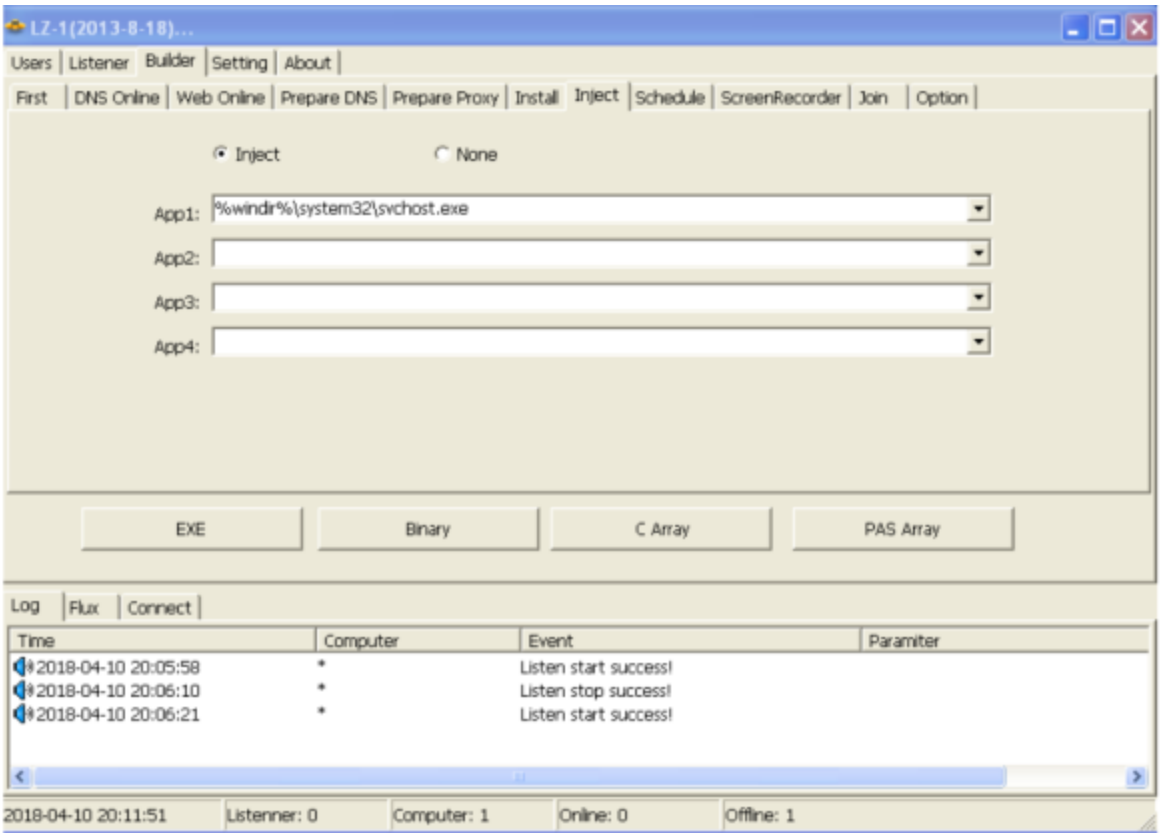
For sake of simplicity, in this article, I'm going to use the DNS Online technique using "www.builder.com" as C2 address. Next, on the "First" tab I specify the campaing ID and the password used by the payload to connect to the C2.



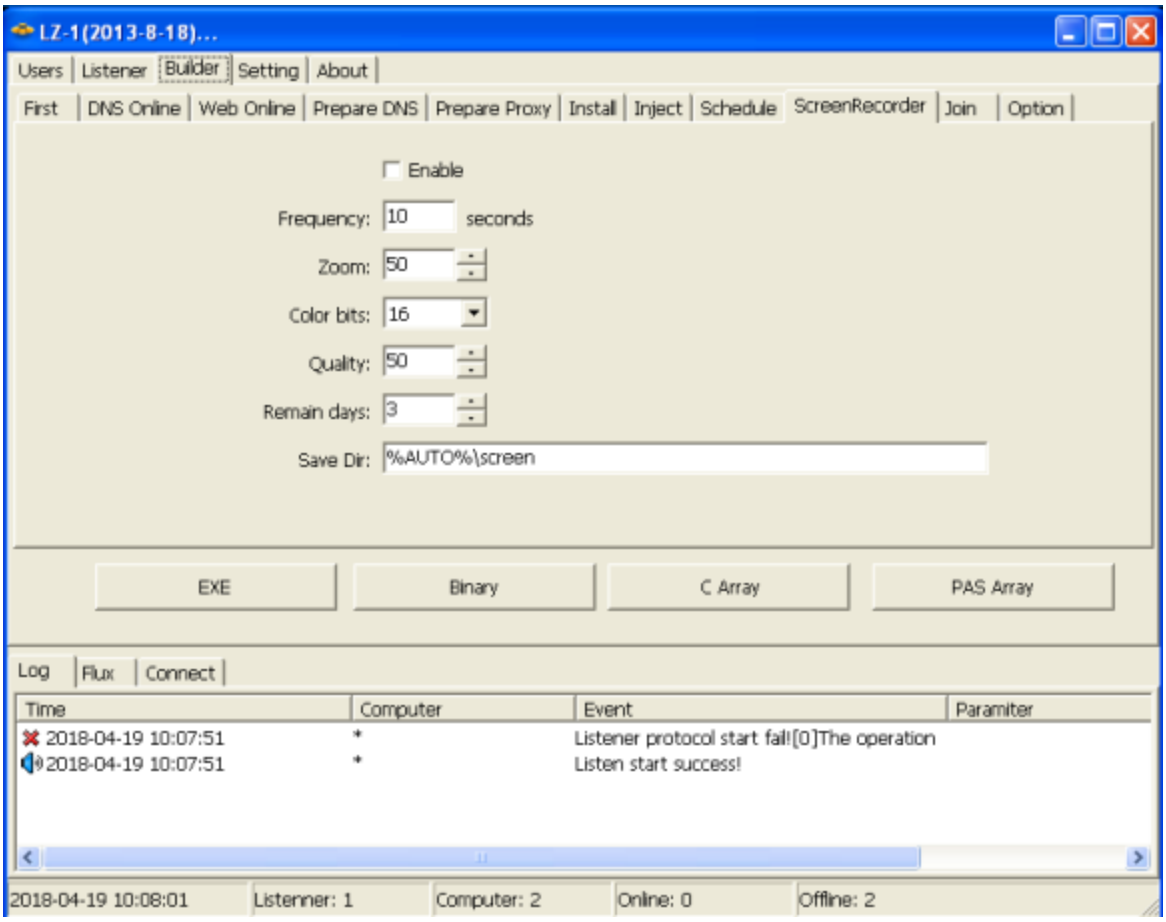
Next, on the Install tab I specify the persistence settings, in this case I'm telling the payload to install a service and I can specify different settings including where to deploy the binaries, the service name and service description. In addition, I can specify that if the Service persistence mechanism fails due to some reason the payload should install the persistence mechanism using the Registry and I can specify which HIVE should be used.



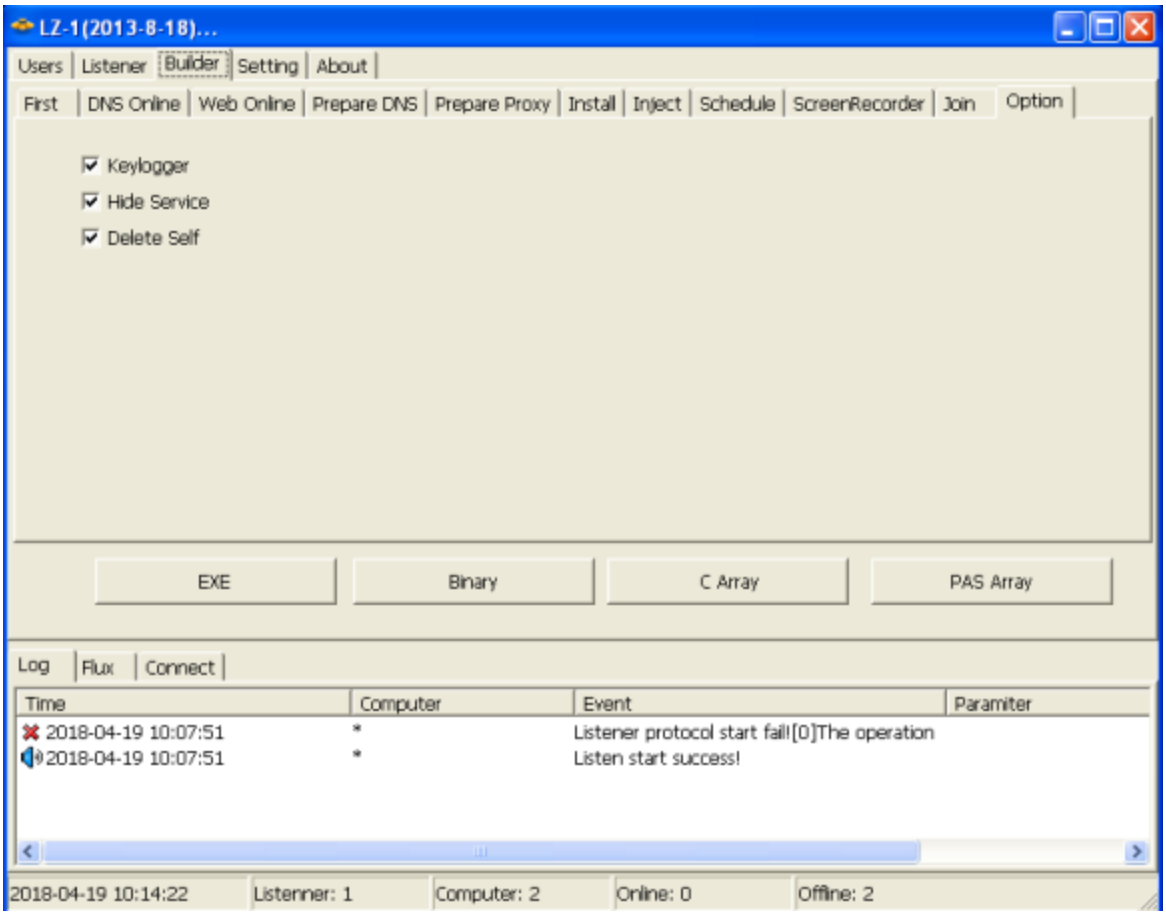
Then, In the inject tab I specify which process will be injected with the malicious payload. In this case I choose “svchost.exe”. This will make PlugX start a new instance of “svchost.exe” and then inject the malicious code into svchost.exe process address space using process hollowing technique.



Other than that, the operator could define a schedule and determine which time of the week the payload should communicate with the C2. Also the operator could define the Screen Recording capability that will take screenshots at a specific frequency and will save them encrypted in a specific folder.

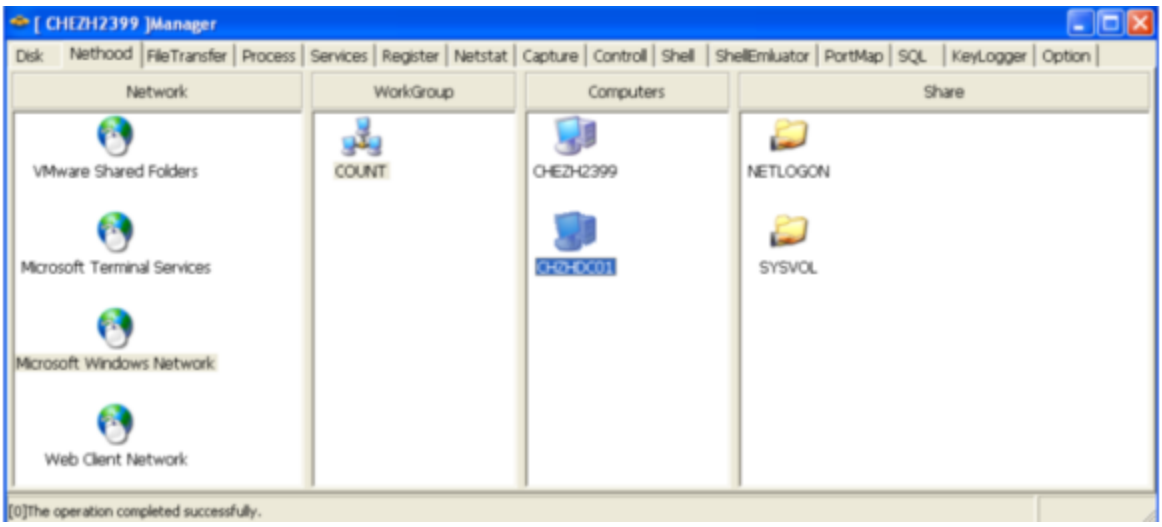


Last settings on the “option” tab allow the operator to enable the keylogger functionality and specify if the payload should hide it self and also delete itself after execution.

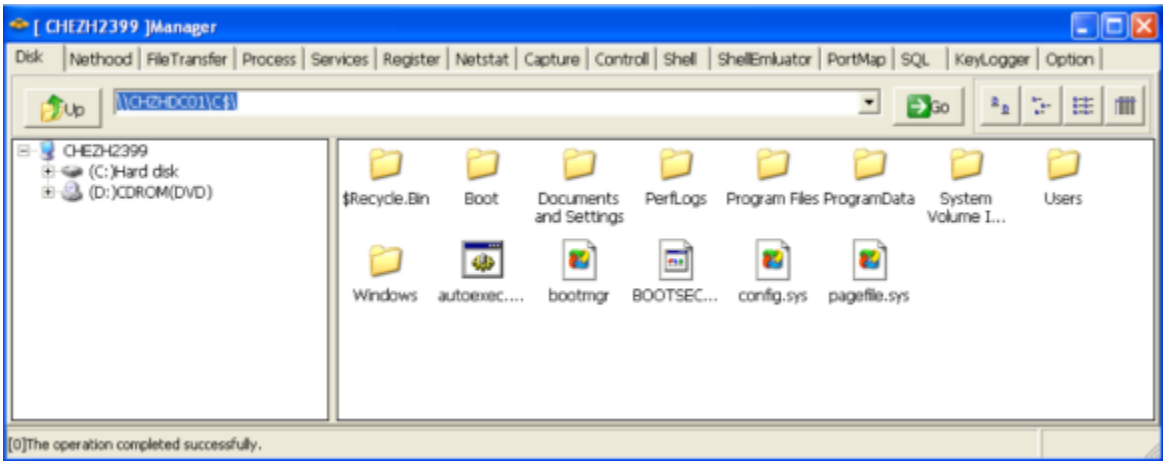


Finally, after all the settings defined, the operator can create/download the payload in different formats. An executable, binary form (Shellcode), or an array in C that can then be plugged in another delivery mechanism e.g, PowerShell or MsBuild. After deploying and installing the payload on a system, that system will check-in into the PlugX controller and an operator can call the “Manager” to perform the different actions. In this example I show how an attacker, after having compromised a system, uses the C2 interface to:

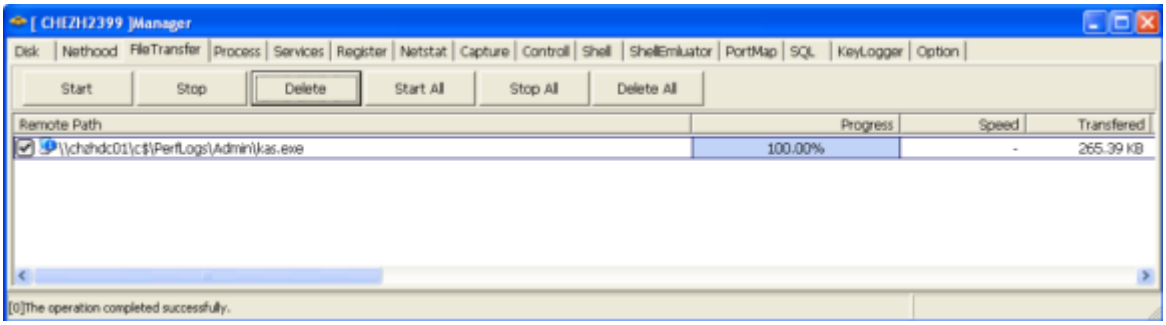
Browse the network



Access remote systems via UNC path

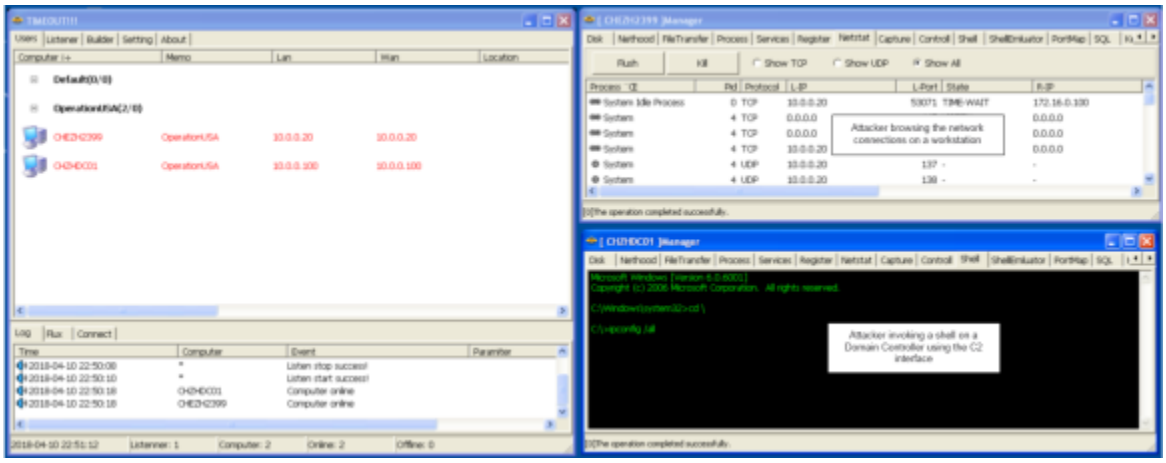


Upload and execute a file e.g., upload PlugX binary

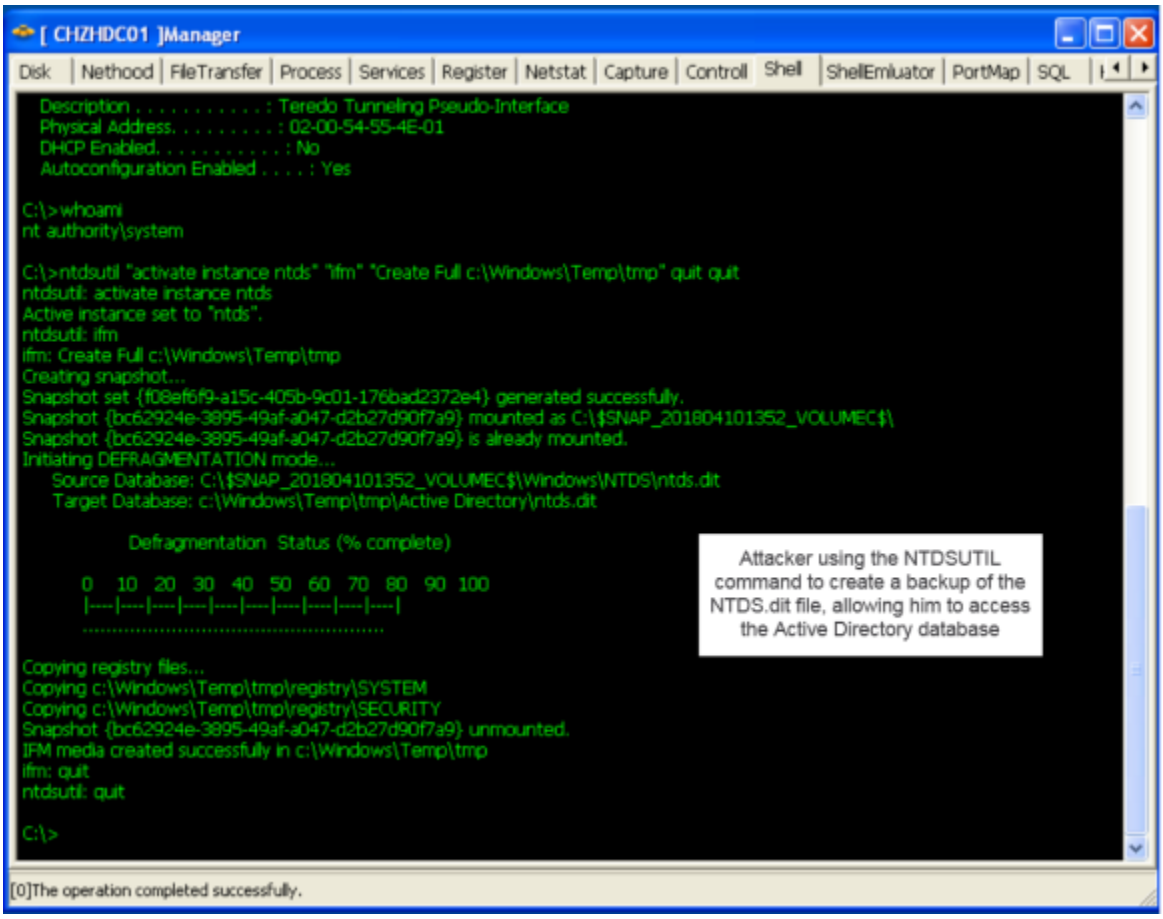


Invoke a command shell and perform remote commands e.g., execute PlugX binary on a remote system

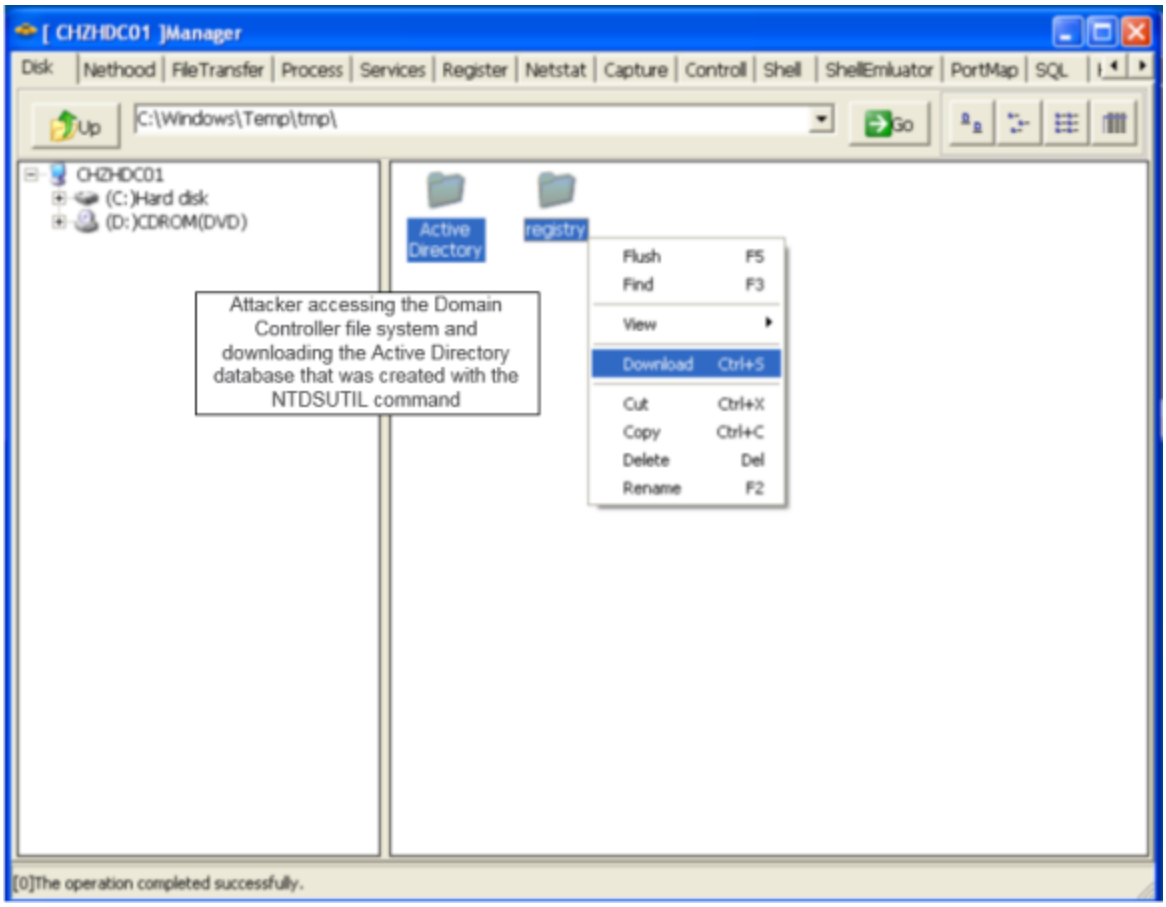
Previous pictures illustrate actions that the attacker could perform to move laterally and, for example, at some point in time, access a domain controller via UNC path, upload the PlugX payload to a directory of its choice and execute it. In this case the pictures show that the PlugX payload was dropped into c:\PerfLogs\Admin folder and then was executed using WMI. Below example shows the view from the attacker with two C2 sessions. One for one workstation and another for a domain controller.



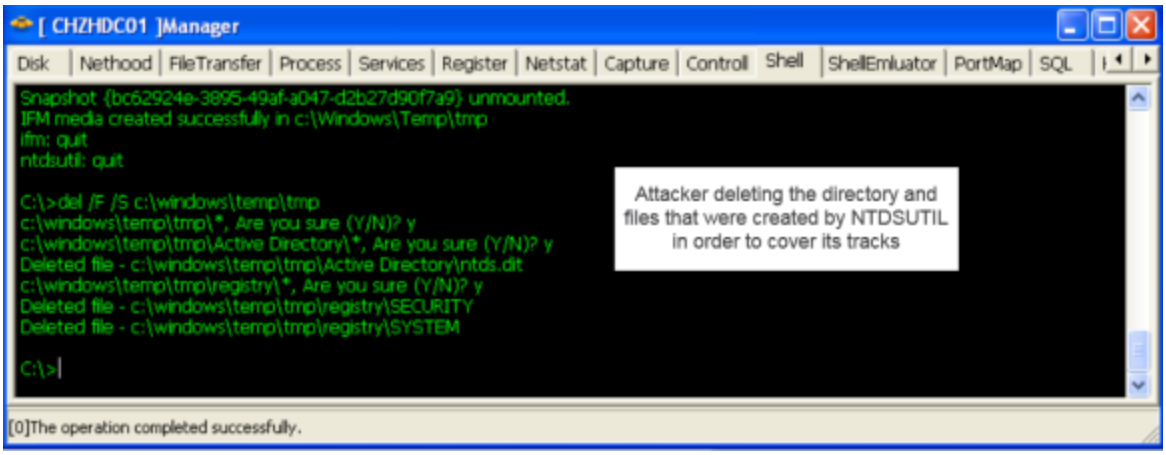
Having access to a domain controller is likely one of the goals of the attacker so he can obtain all the information he needs about an organization from the Active Directory database.



To access the Active Directory database, the attacker could, for example, run the “ntdsutil.exe” command to create a copy of the “NTDS.dit” file using Volume Shadow Copy technique. Then, the attacker can access the database and download it to a system under his control using the PlugX controller interface. The picture below illustrates an attacker obtained the relevant data that was produced using the “ntdsutil.exe” command.



Finally, the attacker might delete the artifacts that were left behind on the file system as consequence of running “ntdsutil.exe”.



So, in summary, we briefly looked at the different techniques a PlugX payload could be configured to speak with a Command and Controller. We built, deploy and install a payload. Compromised a system and obtain a perspective from PlugX operator. We move laterally to a domain controller and installed the PlugX payload and then used a command shell to obtain the Active Directory database. Of course, as you noted, the scenario was accomplished with an old version of the PlugX controller. Newer versions likely have many new features and capabilities. For example, the print screen below is from a PlugX builder

from 2014 (MD5: 534d28ad55831c04f4a7a8ace6dd76c3) which can create different payloads that perform DLL Search order hijacking using Lenovo's RGB LCD Display Utility for ThinkPad (tplcdclr.exe) or Steve Gibson's Domain Name System Benchmarking Utility (sep_NE.exe). The article from Kaspersky "PlugX malware: A good hacker is an apologetic hacker" outlines a summary about it.

That's it! With this article we set the table for the next article focusing on artifacts that might helps us uncover the hidden traits that were left behind by the attacker actions performed during this scenario. Stay tuned and have fun!