Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster

WINTRE: AN ADVERSARY EMULATION TOOL

Functional Specification

Student: Martin Earls / C00227207
Supervisor: Richard Butler

1 Table of CONTENTS

2 INTRODUCTION

The application to be developed is a .NET C# desktop application for Windows operating systems. Users will be able to review and execute security tests, or emulate malicious user behaviour as defined in the MITRE ATT&CK framework. These techniques, coded as individual tests will help generate indicators of compromise in the form of logs created from system events or notable alerts from endpoint security products.

Techniques will be accessible from the GUI of the application and split into separate tactics/categories defined in the ATT&CK framework. They will range from simple command-line based tests to more complex ones utilising WinAPIs.

The user will be able to selectively choose which tests they want to run or save a group of tests as a campaign. Tests can be executed one at a time or added to a queue system, the queue system will simply allow the user to collect and gather a preferred list of tests beforehand and then execute them.

3 GENERAL DESCRIPTION

3.1 GENERAL UI FUNCTIONS

- Display different pages allowing access to functionality across the application.
- Allow user to move between pages with ease.
- Display and track technique queue across pages.
- Display watermark text and then clear this value when a user begins typing in a text box element.
- Display tooltips for relevant UI elements.

3.2 TECHNIQUE FUNCTIONS

- Provide a mechanism for selecting which technique to run.
- Provide a mechanism to compile the relevant source code for the selected technique.
- Provide a mechanism to execute the newly compiled technique.
- Redirect standard output and standard error to the on-screen console log.
- Redirect standard output and standard error to a log file on disk.
- Provide a mechanism to manage a stack-based queue system from the techniques page.
- Allow a user to add a selected technique to the queue.
- Allow a user to delete the next technique on the queue.
- Provide a mechanism that enables a user to run tests both from the queue system and from individually selected tests.
- Retrieve values from matching JSON file of the selected test.
- Display values from matching JSON file.
- Provide a mechanism to pass arguments for complex tests that cannot be ran without arguments.
- Provide a mechanism to distinguish between command line tests and complex tests.
- Provide a mechanism for loading custom tests.
- Provide a mechanism to process JSON data.
- Delete all previously compiled payloads from disk.
- Keep track of any payloads that are detected by Windows Defender by checking if the binary path matches threat protection history.

3.3 CUSTOM TECHNIQUE FUNCTIONS

- Provide a mechanism for the user to input data needed to create a test.
- Create fields for the user to input data containing:
    - Technique name
    - MITRE ATT&CK ID
    - Elevated privileges
    - Category/Tactic
    - Selected source code template/launcher
    - Commands
    - Notes
- Create a relevant JSON file from the specified user input that will be used to load data relevant to the technique.
- Create a relevant source code file based on the command input of the custom test.
- Provide a mechanism to escape a command input, to a string literal and perform processing on the string so that the correct syntax is maintained when passed to the source code file.

3.4 Campaign Functions

- Provide a mechanism to select a group of tests.
- Allow the user to add these tests into a JSON file so that they're easily referenceable.
- Provide a mechanism for loading and deleting tests based on saved campaigns.
- Allow the user to define metadata for the campaigns.
    - Campaign name
    - Focus of campaign
    - Results efficiency percentage
    - Results notes
- Allow the user to run multiple techniques in one session.
- Redirect standard output and standard error to the on-screen console log.
- Redirect standard output and standard error to a log file on disk.
- View previous campaigns and a brief description of their results.

3.5 Report Functions

- Allow a user to enable or disable reporting functionality.
- When enabled, keep track of which tests are ran from either the techniques page or the queue.
- Create a template summarising data of techniques that have been ran.
    - Tactic
    - Test ID
    - Test name
    - Detected
    - Time and date of execution
- Create sections in the template where the tester can fill in their own details.
- Allow users to export, save or delete generated templates.
- Users must be able to preview their report that will be generated.

4 PROJECT SCOPE

4.1 GOAL

- The goal is to make a user friendly, GUI based C# application that supports running small individual security tests, which are repeatable and easily configured.
- Develop a series of relevant and modern techniques based off of the MITRE ATT&CK Framework.
- Integrate as many techniques as possible and have a more or less even spread of the number of tests across categories.

4.2 DELIVERABLES

- Each tactic of the MITRE ATT&CK framework that is applicable to post exploitation must be covered, in that it has relevant techniques available for testing that can be applied to WINTRE's purpose.
- Each technique must be reliable.
- Each technique must be relevant to the MITRE ATT&CK framework.
- Each technique must not cause permanent damage to the operating system or hardware.
- The tool must run on all modern Windows operating systems that have the .NET framework installed.
- The tool must be easily portable between OS editions and function the same, e.g. Windows Server, Education and Enterprise editions.

4.3 RISKS

There is a potential risk for misuse of this product by end users who may seek to cause damage or attack systems using some of the tests in this project. In general, the majority of tests developed for the project will not be employing heavy obfuscation or stealth techniques, nor will there be any auto-exploitation functionality implemented.

Most tests are also designed to run locally on a single system, automatic command and control functionality is not provided. Some tests that are particularly old and common e.g. process read of LSASS to dump process memory, should always result in detection with any anti-virus product. These deliberate measures should be sufficient in preventing damage by a malicious user who may seek to run these tests.

5 TARGET MARKET

5.1 BROAD

The target market for this tool is applicable for any organisation that has a sizable dependence on IT infrastructure. Broadly speaking this tool could be used in millions of various companies, even ones where employees do not utilise computers in day to day operations. Organisations still need some infrastructure for storing company, employee and customer data.

5.2 SPECIFIC

Penetration Tester: Pentesters may utilise the tool during purple team engagements to test the effectiveness of a security operations centre, rather than writing and testing many techniques manually they can efficiently test in a more results oriented manner, not worrying about the validity of the test results. Purple Team engagements could also be conducted with the pentester as the main tester while relaying information to a SOC. Engagements are usually very expensive and take a great deal of preparation and planning to conduct. The tool reduces time spent in this process meaning more time can be spent performing adversary emulation rather than preparing for an engagement.

Security Operations Centres: SOCs looking to carry out fast, efficient adversary simulation testing in line with the MITRE ATT&CK framework would highly benefit from such a tool. WINTRE would be exponentially more cost effective due to the high cost of hiring a penetration tester/red team to perform a purple team engagement. Time is also saved by relying on in house setup and installation rather than working with an external company every simulation. SOCs are usually in house and would have a say in how their security is spent, typically however this only applies to medium to large sized companies that can afford a team of analysts that make up the SOC.

6 FUNCTIONAL REQUIREMENTS

6.1 FUNCTIONAL

#	Function	Description	Criticality	Obstacles	Dependencies
1	Run test	Simulate technique	HIGH	Local binary compilation	GetTests
2	Complex test	Check if a test is complex	HIGH		-
3	Add test		HIGH	JSON	-
4	Get tests	Load all tests from file system on startup	HIGH	-	-
5	Escape command	Convert command to string literal	HIGH	Filtering commands with quotes and symbols	AddTest
6	Create campaign	Define a new campaign	MEDIUM	-	RunTest
7	Load campaign	Search, store campaign details for display	MEDIUM	-	CreateCampaign
8	Review campaign	View details of a previous campaign	MEDIUM	-	LoadCampaign
9	Add to queue	Add a test to the queue	MEDIUM	-	-
10	Remove from queue	Remove a test from the test queue	MEDIUM	-	AddToQueue
11	Add note	Add notes during campaign	LOW	-	-
12	Enable reporting	Start generating template	LOW	-	-
13	Export report	Export a report from preview	LOW	-	Enable Reporting
14	Delete report	Delete an existing report in progress	LOW	-	Enable Reporting

6.2 Usability

- The application must be convenient to use.
- The user interface must be well designed, responsive and easy to navigate.
- The interface must have built in tool tips and watermarks for textbox values, in order to further enhance legibility and ease of use.
- Creating custom techniques must be accessible for simple tests.
- Report generation must be of a quality compatible with a manual report, such that the user may rely on both dynamic report generation and their own post editing.

6.3 Reliability

- The application must not crash or create vulnerabilities on the system.
- Malicious techniques should be able to undo any potential system changes made, e.g. decrypt user files in ransomware testing.
- The application must have the ability to clear any malicious payloads compiled on the system.
- The application must maintain verbose logs with the capability to present these to the user or store them for external analysis.
- Logs generated by the application must record all relevant steps and actions undergone when executing techniques.
- Accurate timestamps of potential detections must be maintained for post engagement correlation.

6.4 Performance

- The application must not hog system resources.
- The application must use minimal resources to accomplish all tasks.
- The application must not be prone to memory leaks.

6.5 Supportability

- The accompanying usage manual must cover the necessary technical aspects while also presenting usage steps at a high level for ease of use.
- Installation of the tool must be simplified in all possible areas with minimal user interaction required.
- The application must support all applicable modern Windows operating systems without error.
- The application must support logging detections that come as a result of Windows Defender.
- If the .NET framework is not present on the system, an installation script must detect this and inform the user or download and perform the installation for them.

7 TECHNIQUE FUNCTIONS

Main Techniques:

1. Code Execution
    a. Execute a disguised PE - run an executable file that may have a masked extension using whitespace. Example: "testfile.pdf   .exe". A simple test to see if anti-virus picks up on files using whitespace in names to hide the real extension.
    b. Rundll32 - run an arbitrary payload, typically from a DLL. Attackers can use this instead of directly running executables.
    c. JavaScript - run shell commands from a JavaScript context.
2. Persistence
    a. Create local backdoor administrator - creating a local admin account on the machine.
    b. Create hidden local backdoor administrator - creating a local admin account on the machine that is hidden using string manipulation. Example: "net user $ Hidden! /add /active:yes".
    c. Extension hijacking - make it so that when a user opens a certain type of file (e.g. text files), they actually launch a script that will execute a pre-defined payload in the script.
3. Privilege Escalation
    a. Get unquoted service paths - enumerate all local service paths, filter them for missing quotes which can be used to change the intended binary path, allowing an attacker to execute their own PE in place of the original.
    b. Search for sensitive strings (e.g. "password") - simply get a list of all files, search for matching strings that may involve passwords.
    c. Search for credential related files - get a list of all files and search for files that may store credentials. Even if these files don't exist, it could be a good indicator of compromise to check for if someone is noticed to be looking for these files.
4. Defence Evasion
    a. Crypter - encrypt an executable, bypass Windows Defender static analysis and (hopefully) runtime analysis.
    b. Create Windows Defender folder exclusions - add folder exclusions from which files could be executed.
    c. Turn off Windows Defender - temporarily disable runtime scanning and other protections if possible.
    d. Basic Powershell/Windows command line obfuscation - execute standard shell commands but with obfuscation, such as string concatenation.
5. Credential Theft
    a. Keylogger - log and store user keystrokes for a specified time interval.
    b. Decrypt vault credentials - decrypt windows vault credentials.
    c. Check for auto logon credentials - check the registry for these credentials, even if they don't exist this could be indicator of compromise when this anomalous behaviour is discovered.
    d. Dump LSASS process memory - create a process dump of LSASS to extract local NTLM hashes.
    e. Dump registry hives to extract NTLM hashes (SAM / SYSTEM)
    f. Create a dump of SECURITY hive to extract MS Cache V2 credentials/domain account credentials.

    g. Credential prompt - use PowerShell to prompt a user into entering their password via GUI. Note: tester simply prompts themselves and enters any value to fulfil the test.
    h. Decrypt Wi-Fi credentials - credentials related to access points may be stored on a system and can be decrypted in the context of the user's session.
    i. Custom network provider - add a custom credential manager that simply saves the user's password in plaintext.
6. Discovery
    a. Local account discovery - list local user accounts.
    b. Local permissions discovery - enumerate local user permissions.
    c. Local process discovery - enumerate processes and installed software.
    d. Get system information - retrieve system / OS related information.
    e. Internal site discovery - check the registry for a trusted site list which may expose intranet sites.
    f. File and directory discovery - gather a list of files on the system, determine programs installed and potential attack surface.
7. Data Exfiltration
    a. Archive files for exfiltration over the internet - e.g. extracting any retrieved credential files, send to remote server for analysis.
    b. Encode files for exfiltration over the internet.
    c. Encrypt files for exfiltration over the internet.
    d. Use steganography for exfiltration over the internet.

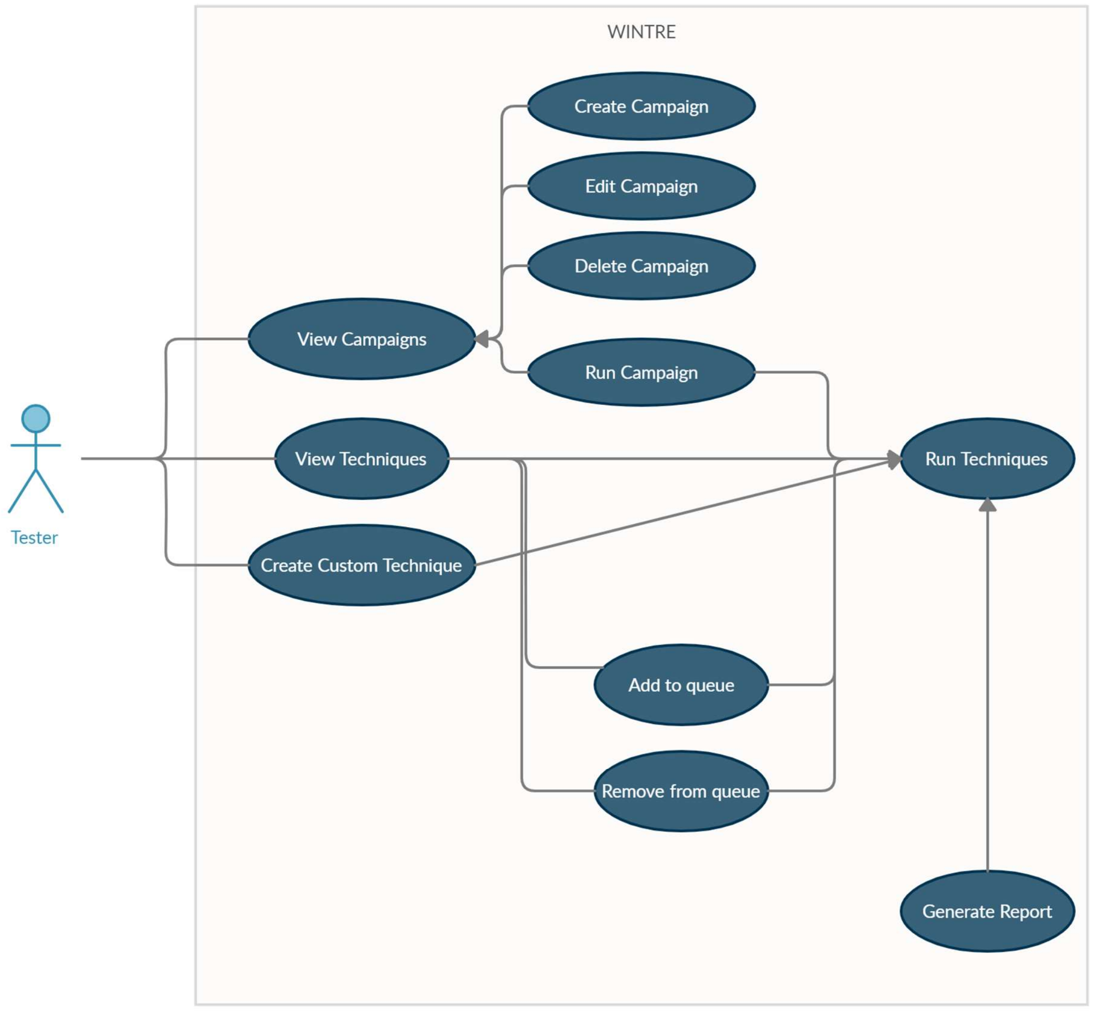
Discretionary Techniques:

1. Privilege Escalation
    a. Spawn SYSTEM Shell (PsExec) - Copy a service executable to a local file share, use this service executable to launch an arbitrary service that could spawn a local SYSTEM shell with maximum privileges.
    b. Service binaries - Get a list of service binaries and determine if a user has permissions to append or modify data.
2. Lateral Movement
    a. PsExec Lateral Movement - This lateral movement would only take place between the 2 designated machines. Copies a service executable to a remote file share, executes it using named pipes to create an interactive SMB shell, equivalent to a reverse shell connection. Attacker in this case authenticates with a username and password of a local administrator.
    b. Pass-The-Hash (PsExec) - Use NTLM hashes as opposed to a password for configuring a remote service on a file share to get an interactive SMB shell.
3. Impact
    a. Ransomware - A controlled ransomware program, which would encrypt all files in a user's directory and then decrypt them to simulate a ransomware attack.
4. Command and Control
    a. Execute C# (PS) reverse shell back to Kali - Create a reverse shell to an external server listening for a callback via TCP Socket WinAPIs.
    b. Execute PowerShell reverse shell back to Kali - Create a reverse shell to an external server listening for a callback via PowerShell.
    c. Downloaders - Use built in Windows binaries to download external payloads as referenced from the "Living off the Land Binaries (and Scripts)" project. E.g. CertUtil.exe.
    d. Powershell download utility, utilising WinAPIs through .NET framework.

8 APPLICATION USERS

Systems Administrator/Tester - The primary user for this application will be a system admin/tester. This is based on the fact that the majority of organisations do not perform adversary emulation due to the high cost and expertise required to do so, having a tool that can achieve this gives much more power to smaller organisations and systems administrator teams that do not have a dedicated team of security analysts to rely upon. This user is the most common, expected general application user.

The system admin serves as a general user/tester, that may also be fulfilling a security role and seeks to perform adversary emulation.



9 ABSTRACT USE CASES

The following use cases are based on the three most common scenarios of a user interacting with the application, running tests, adding a custom test or generating a report based on tests that are being run.

Running a series of techniques:

Primary Actor:

- Tester/Sys admin

Preconditions:

- None

Success Guarantee:

- The user can select techniques to run.

Main Success Scenario:

- User starts the application.
- User selects the techniques page.
- User selects their preferred category.
- User can then select tests and review the test information.
- User can add the test to the queue or simply run the test.
- User can review the output of the test in the log output or run it from the queue on the sidebar where the output will also be send to the logs.

Creating a custom technique:

Primary Actor:

- Tester/Sys admin

Preconditions:

- User has tested their custom technique's syntax is correct and functioning.

Success Guarantee:

- The user can create a custom test based on a command of their choosing to fulfil a specific MITRE technique.

Main Success Scenario:

- User enters a technique name.
- User enters a related MITRE ATT&CK ID.
- User selects whether their technique runs with elevated privileges or not.
- User select the relevant category/tactic of their technique.
- User chooses a template, either cmd.exe or powershell.exe to launch their technique.
- User enters their command(s) to execute their technique.
- User enters notes related to the technique.

Generating a report:

Primary Actor:

- Tester/Sys admin

Preconditions:

- User has started the application and has selected tests to run.

Success Guarantee:

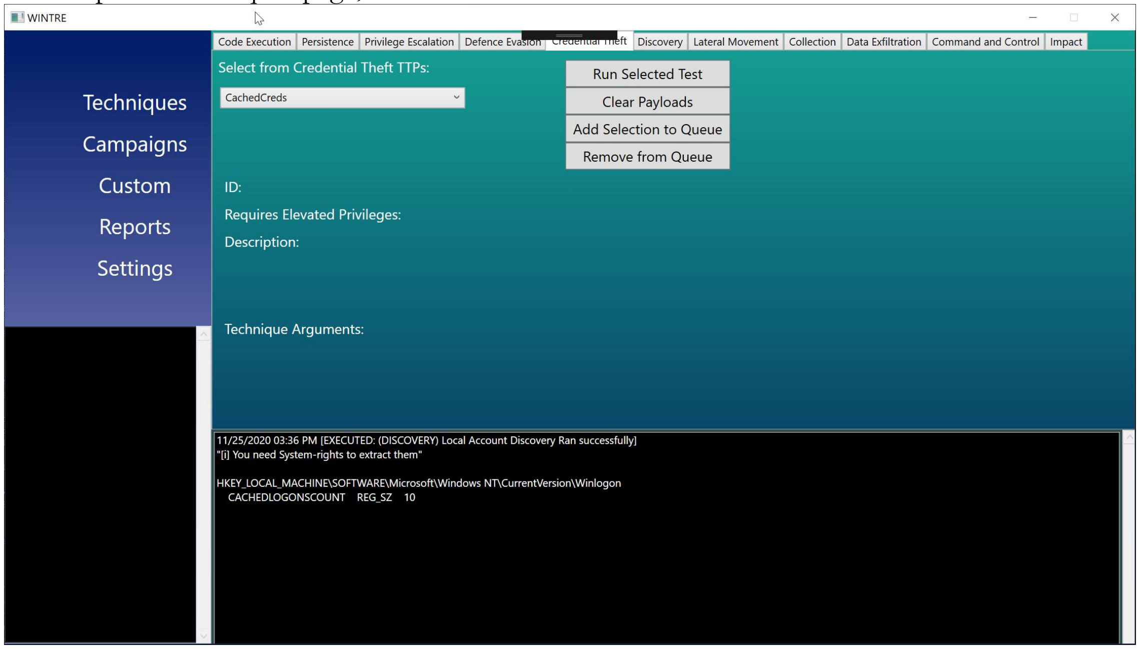
- The user is able to successfully enable report generation and the report is dynamically generated as tests are ran. This means a table containing the tests and its details are generated along with additional fields relevant to the testing (such as time / date) for the tester to fill in.

Main Success Scenario:

- The user starts the application.
- The user runs some tests or adds some tests to a queue.
- The user enables the report generation feature.
- The user runs several tests.
- The user can then view the results generated from tests and manually mark off if a technique was detected.

9.1 Mock-up UI Design Functionality

Mock-up for Techniques page, where a user can select and run tests.



Select from Credential Theft TTPs: CachedCreds

ID:

Requires Elevated Privileges:

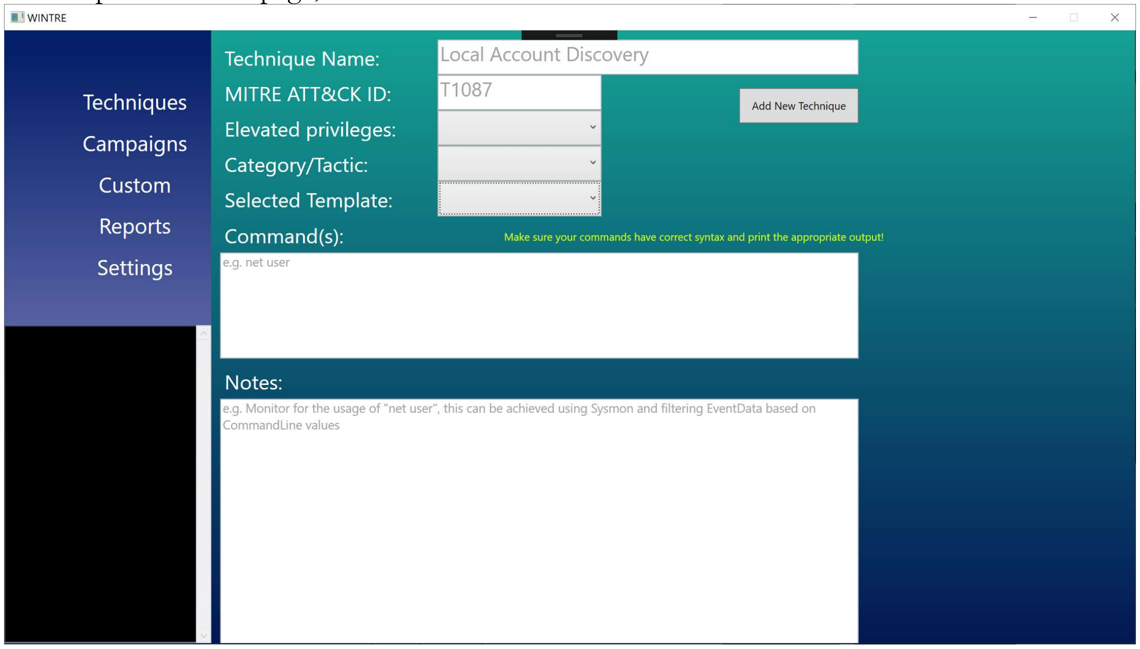
Description:

Technique Arguments:

```
11/25/2020 03:36 PM [EXECUTED: (DISCOVERY) Local Account Discovery Ran successfully]
"[i] You need System-rights to extract them"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    CACHEDLOGONSCOUNT    REG_SZ    10
```

Mock-up for Custom page, where a user can create custom tests.



Technique Name: Local Account Discovery

MITRE ATT&CK ID: T1087

Elevated privileges:

Category/Tactic:

Selected Template:

Command(s): Make sure your commands have correct syntax and print the appropriate output!

e.g. net user

Notes:

e.g. Monitor for the usage of "net user", this can be achieved using Sysmon and filtering EventData based on CommandLine values

10 Constraints

10.1 Delivery Date

The delivery date for the finalised product is April 2021, this may be a short time frame depending on the number of techniques to be implemented. To balance out the deliverables proposed techniques for development are to be split into mandatory and discretionary categories. Mandatory ones will take precedence as well as ensuring that each category has at least 3 techniques to choose from. In an ideal situation where all discretionary functionality is completed, each category is to have 5+ techniques.

10.2 Platforms

The tool will be developed using the .NET framework which is compatible with most modern Windows operation systems. Testing will take place on Windows Education.

Target framework:

- .NET Framework 4.7.2
- 64-bit architecture, product may be compiled to also run on 32-bit subsystems depending on feasibility of the developed techniques

10.3 Signatured tests

Some techniques developed may have to be updated or removed during testing, despite having previously functioned with no issues. This may happen if the technique in question is signatured and Windows Defender starts detecting it unexpectantly. This may happen to any technique eventually if the test begins to be considered malicious when anti-virus signatures are updated.

It also depends on the validity of testing a certain technique if you know definitively that technique will generate a notable alert. To prevent tests from being uploaded and signatured, Automatic Sample Submission and Cloud Based Protection features of Windows Defender will be disabled during development.

11 Project Plan

Sprint	Plan	Due Date	Deliverable
1	Custom Tests	30/11/2020	Finish custom tests functionality, test out more commands to be added, finish the PowerShell command template.
2	Techniques/Standard Tests	11/12/2020	Complete standard test functionality, load test details from relevant JSON files.
3	Complex Tests	23/12/2020	Technically "standard" tests but ones that will require additional arguments, update UI and simulation mechanism to reflect this, e.g. arguments for reverse shell.
4	Further develop techniques	7/1/2021	Further expand on techniques in various categories.
5	Further develop techniques	21/1/2021	Further expand on techniques in various categories.
6	Report Functionality	7/2/2021	Develop report generation and export functionality, preferably to word documents as a template.
7	User Interface	21/2/2021	Enhance UI aspects, add graphics and icons.
8	Performance Testing	7/3/2021	Test performance of the application, benchmark the application on various Windows systems and seek to improve performance if possible.
9	Unit testing	21/3/2021	Create unit tests for specific parts of the application to test for redundancy, e.g. an invalid custom test etc.
10	Security testing	28/3/2021	Test the application for any potential security issues, potentially encrypt logs?
11	Reserved	19/4/2021	Complete any outstanding work by project due date.

12 METRICS

Criteria	**Description**
Usability	The product must be easy to use and conform to usability standards. The product is easy to setup and the user can quickly display and choose tests to execute.
Platform	The target platforms are modern Windows desktop operation systems, the application should also be compatible with Windows Server and similar OS.
Security	The application must be secure and not create any unintended vulnerabilities or alter a system's configuration beyond repair.
Efficiency	The application must be able to produce quantifiable results, i.e. an efficiency rating of the existing security controls that could be recorded in the report generation manually during testing. This efficiency rating serves as the central benchmark to how an organisation will improve testing over time based on the techniques that are being ran.
Supportability	Sufficient documentation is provided to the user in the form of a user manual that goes into detail on techniques. This detail is also provided in the application itself where possible. Complex tests that require additional user interaction or setup are also accounted for in supplementary documentation.

13 PRECEDENT

The major precedent for this project was during my internship. My job role as a security analyst heavily focussed on creating and facilitating the testing of the MITRE ATT&CK framework against existing endpoint security controls. During this time, I was working with several Anti-Virus and Endpoint Detection and Response products, often with the end goal of analysing which product was most viable for the protection of company assets. This meant the efficiency of each product had to be tested regularly and required many iterations of testing to produce a comprehensive report with repeatable, relevant techniques that were based on the MITRE framework.

I was testing many techniques meaning there was a heavy time investment spent researching and programming the techniques rather than actually testing. As well as that, the tests would have to be manually chosen beforehand meaning the tester had to note down exactly which scripts were being ran at what times, and note down when they were executed. This was necessary information for the security operations centre to be able to correlate their log data and perform threat hunting or analysis of the specific tests being ran.

13.1 SIMILAR APPLICATIONS

Scythe is the primary similar application this product is based on. Scythe is a closed source adversary emulation platform. During my work placement I worked with a penetration tester using Scythe to conduct a purple team engagement. I discovered some of the disadvantages of a server-client component when using it, as Scythe incentivises the user to run many techniques together in a single executable, that executable is also compiled on the server.

There were two major disadvantages to this, firstly it made the security operation centre's job of analysing the results of the tests more difficult given that they were all being executed within seconds of each other. This also meant each test had to be configured, compiled and then downloaded manually on to the target endpoint if they were to be separated.

Secondly, as the tests were compiled externally this required manual patching of each binary as existing security controls would block execution of any binaries not compiled by the local user or ones that did not have a valid certificate to prove their origin. This was bypassed using signature spoofing but by compiling the executables locally with WINTRE, we avoid this unnecessary complication.