Pre-Compliance Accreditation Tool for Python

Justin Dierking
Hardbit Solutions Ltd.

Abstract

I am demonstrating a Python based program for scanning and re-mediating security findings on Windows and Linux based information systems. PCAT2PY (Pre-Compliance Accreditation Tool for Python) is capable of running against remote systems as a trusted user or running locally against a system with administrative privileges. Following a scan or re-mediation using PCAT2PY, a report, formatted as HTML, is produced indicating the compliance, standard output, identification, and discussion of every finding that was scanned or re-mediated.

Categories and Subjects: Introduction – Theory of Operation – Design – Requirements – Conclusion

Introduction

The purpose of this project is to develop a small footprint capability to detect and mitigate security findings against Microsoft Windows and Linux based information systems. PCAT2PY was developed as an agent-less, trusted tool that does not require installation. This tool is used as a copy-and-run application for local and remote information systems.

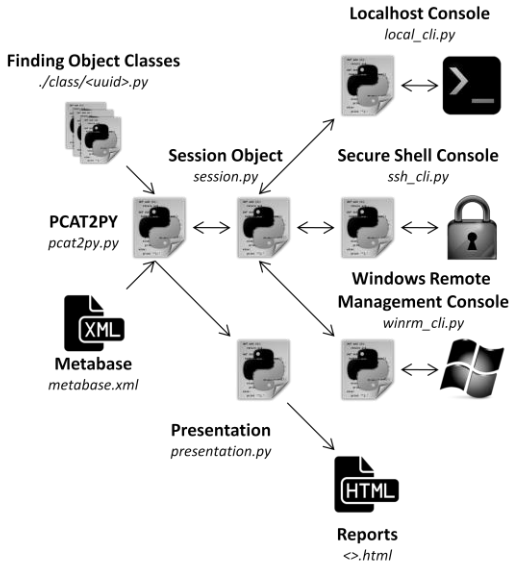


Figure 1: *PCAT2PY general architecture. This figure illustrates the logical connections between PCAT2PY's components.*

This paper is intended to provide the reader with an understanding of the development and operation of this program. Specifically, theory of operation, design, and operational requirements are discussed.

Theory of Operation

PCAT2PY is written in the Python programming language. It is a command prompt based application that is controlled by using switches and arguments to control its behavior prior to execution. To simplify portability, PCAT2PY has also been bundled as self-contained, 64 bit binary executables for Linux and Windows.

PCAT2PY has two modes of operation. First PCAT2PY can be run in scan mode to evaluate the compliance of information systems against selected security findings. The security findings that are evaluated can be selected individually or by posture. Postures exist for scanning operating systems as well as applications.

Second, PCAT2PY can be run in re-mediation mode where non-compliant configuration settings are corrected. Great care must be taken when running PCAT2PY in remediation mode. The current implementation has no functionality for reversing changes made while correcting non-compliant settings.

By default PCAT2PY generates no standard output or files of any kind. To generate standard output, PCAT2PY can be run in 2 debugging levels. Debug level 1 simply indicates compliance of a finding or findings. Debug level 2 indicates compliance and standard output from the commands executed by PCAT2PY to evaluate a finding. Finally, PCAT2PY can be called with an HTML switch to generate a report formatted as a HTML file.

Design

The design of PCAT2PY uses an object oriented approach that is based around a metabase, session object, finding objects, and console objects. The metabase consists of extensible markup language (XML) that defines the findings. Console objects encapsulate methods for interacting with local or remote hosts. The session object encapsulates the instantiated console object as well as the instantiated finding objects. The session object has methods for generating debug standard output, scanning, and re-mediation. Finding objects encapsulate methods for checking and fixing configuration settings.

PCAT2PY currently defines through console object classes. There is an implementation for Secure Shell (SSH), Windows Remote Management (WINRM), and local host. These console object classes expose the operating system to PCAT2PY. For Microsoft Windows based hosts, high level methods have been implemented for accessing the registry, security database, event log auditing settings, and Powershell. For Linux based hosts, a single method for executing shell commands has been implemented.

Every time PCAT2PY is executed, a session object is instantiated and populated. A session object contains a console object as well as a dictionary of all the finding objects that have been selected for use for the impending scan or re-mediation. In addition, the methods for scanning, re-mediation, and debugging are implemented within the session object. The session object's purpose is to encapsulate the data that PCAT2PY collects and the console object used to gather that data.

The finding object is the most basic unit to PCAT2PY. Every finding object is referenced by a universally unique identifier (UUID). Within a finding object, there are members for storing compliance, standard output, and UUID. Every finding object has a check method; and optionally, a fix method. Fix methods are not implemented in cases where re-mediating a finding is highly subjective and dependent on the information system's environment.

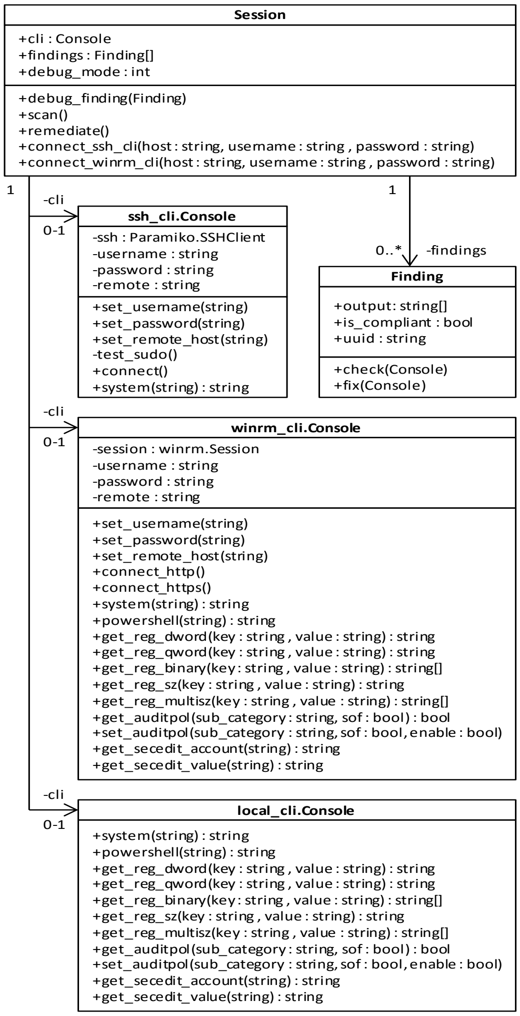
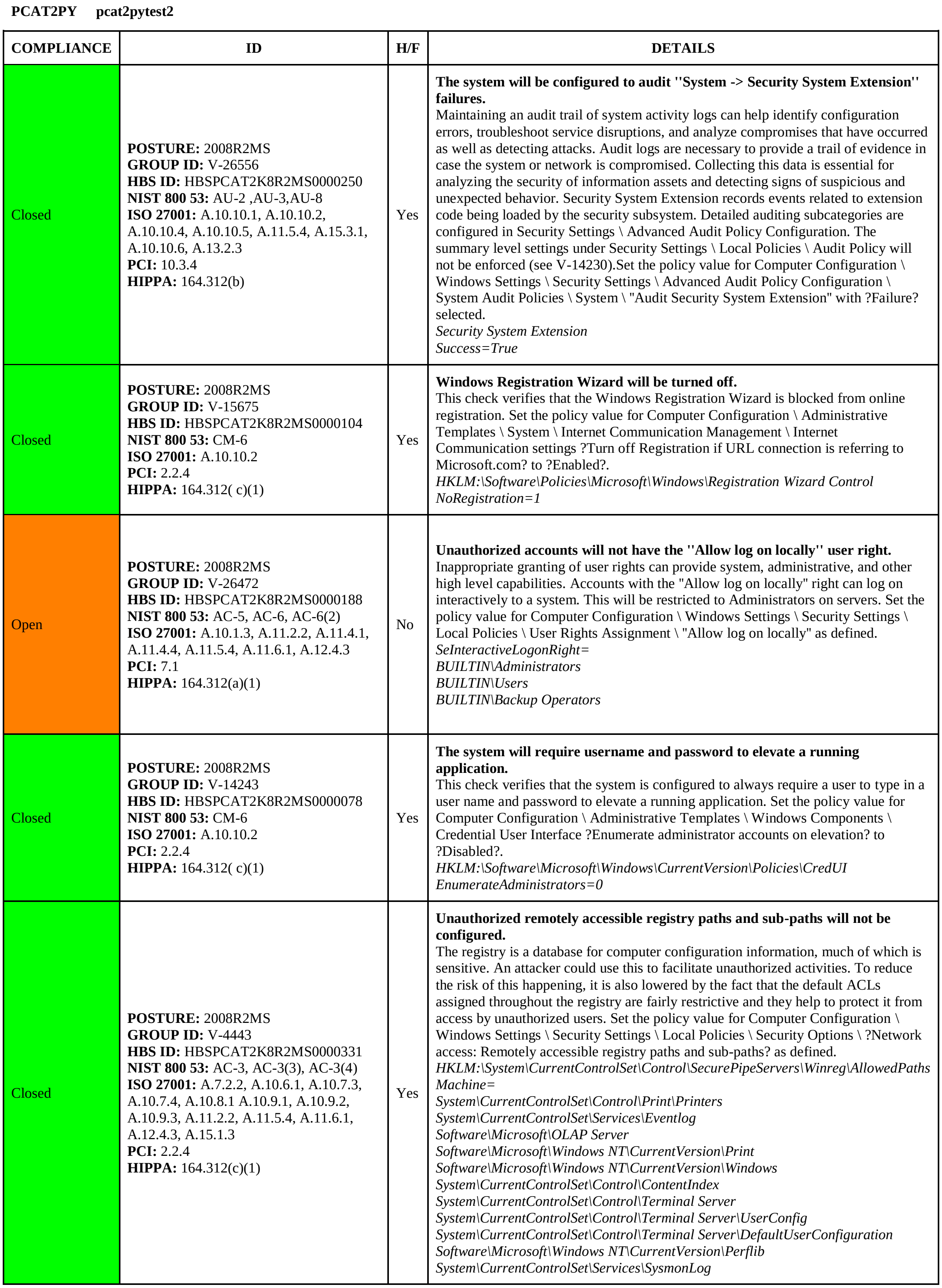

Session
+cli : Console
+findings : Finding[]
+debug_mode : int
+debug_finding(Finding)
+scan()
+remediate()
+connect_ssh_cli(host : string, username : string , password : string)
+connect_winrm_cli(host : string, username : string , password : string)
1
1
-cli
0-1
ssh_cli.Console
-ssh : Paramiko.SSHClient
-username : string
-password : string
-remote : string
+set_username(string)
+set_password(string)
+set_remote_host(string)
-test_sudo()
+connect()
+system(string) : string
0..*
-findings
Finding
+output: string[]
+is_compliant : bool
+uuid : string
+check(Console)
+fix(Console)
-cli
0-1
winrm_cli.Console
-session : winrm.Session
-username : string
-password : string
-remote : string
+set_username(string)
+set_password(string)
+set_remote_host(string)
+connect_http()
+connect_https()
+system(string) : string
+powershell(string) : string
+get_reg_dword(key : string , value : string) : string
+get_reg_qword(key : string , value : string) : string
+get_reg_binary(key : string , value : string) : string[]
+get_reg_sz(key : string , value : string) : string
+get_reg_multisz(key : string , value : string) : string[]
+get_auditpol(sub_category : string, sof : bool) : bool
+set_auditpol(sub_category : string, sof : bool, enable : bool)
+get_secedit_account(string) : string
+get_secedit_value(string) : string
-cli
0-1
local_cli.Console
+system(string) : string
+powershell(string) : string
+get_reg_dword(key : string , value : string) : string
+get_reg_qword(key : string , value : string) : string
+get_reg_binary(key : string , value : string) : string[]
+get_reg_sz(key : string , value : string) : string
+get_reg_multisz(key : string , value : string) : string[]
+get_auditpol(sub_category : string, sof : bool) : bool
+set_auditpol(sub_category : string, sof : bool, enable : bool)
+get_secedit_account(string) : string
+get_secedit_value(string) : string

Figure 2: *UML diagram of PCAT2PY's class structure. The remote access console classes are built as wrappers using the paramiko[3] and pywinrm[4] Python modules.*

The metabase is central to the operation of PCAT2PY. Each finding is defined as an XML key with a UUID assigned as an attribute. Within each finding's key, elements of severity, discussion, posture, and cross referenced compliance standards are present. If a method exists to check compliance or additionally fix a non-compliant configuration setting, a class tag is present in the findings key. Every class tag is given another UUID attribute referencing the finding class definition to use for scanning and re-mediation. If no class tag is present in the finding's key, the finding is treated as a manual finding.

```
<?xml version="1.0"?>
<root>
 <finding uuid="20b3240a-5cc5-11e4-af55-00155d01fe08">
  <class uuid="20b32bb2-5cc5-11e4-af55-00155d01fe08"/>
  <severity>CAT III</severity>
  <group_id>V-1090</group_id>
  <rule_title>Caching of logon credentials will be limited.</rule_title>
  <vulnerability_discussion>The default Windows configuration caches the last
logon credentials for users who log on interactively to a system. This feature is
provided for system availability reasons such as the user''s machine is
disconnected from the network or domain controllers are not available. Even though
the credential cache is well-protected, storing encrypted copies of users
passwords on systems do not always have the same physical protection required for
domain controllers. If a system is attacked, the unauthorized individual may
isolate the password to a domain user account using a password-cracking program,
and gain access to the domain. Set the policy value for Computer Configuration \
Windows Settings \ Security Settings \ Local Policies \ Security Options \
?Interactive Logon: Number of previous logons to cache (in case Domain Controller
is not available)? to ?2? logons or less.</vulnerability_discussion>
  <posture>2008R2DC</posture>
  <hippa>164.312(c)(1)</hippa>
  <pci>2.2.4</pci>
  <hbs_id>HBSPCAT2K8R2DC0000014</hbs_id>
  <nist_800_53>CM-6</nist_800_53>
  <iso_27001>A.10.10.2</iso_27001>
 </finding>
 <finding uuid="20b3632a-5cc5-11e4-af55-00155d01fe08">
  <class uuid="20b36852-5cc5-11e4-af55-00155d01fe08"/>
  <severity>CAT II</severity>
  <group_id>V-3383</group_id>
  <rule_title>The system must be configured to use FIPS-compliant algorithms for
encryption, hashing, and signing.</rule_title>
  <vulnerability_discussion>This setting ensures that the system uses algorithms
that are FIPS-compliant for encryption, hashing, and signing. FIPS-compliant
algorithms meet specific standards established by the U.S. Government and must be
the algorithms used for all OS encryption functions. Set the policy value for
Computer Configuration \ Windows Settings \ Security Settings \ Local Policies \
Security Options \ ''System cryptography: Use FIPS compliant algorithms for
encryption, hashing, and signing'' to ''Enabled''.</vulnerability_discussion>
  <posture>2012MS</posture>
  <hippa>164.312(a)(2)(iv),164.312(e)(2)(ii)</hippa>
  <pci>8.3,8.5,4.1,3.6,2.3</pci>
  <hbs_id>HBSPCAT12MS0000298</hbs_id>
  <nist_800_53>AC-17(2),AC-18(1),SC-9,SC-9(1),SC-13,SC-13(1)</nist_800_53>
  <iso_27001>A.10.6.1, A.10.8.1, A.10.8.5, A.11.4.1, A.11.4.2, A.11.4.6, A.11.7.1,
A.11.7.2, A.10.9.1, A.10.9.2, A.15.1.6</iso_27001>
 </finding>
 <finding uuid="20b38602-5cc5-11e4-af55-00155d01fe08">
  <severity>CAT II</severity>
  <group_id>V-14225</group_id>
  <rule_title>Passwords for the built-in Administrator account must be changed at
least annually or when a member of the administrative team leaves the
organization.</rule_title>
  <vulnerability_discussion>This check verifies that the passwords for the default
and backup administrator accounts are changed at least annually or when any member
of the administrative team leaves the organization. Define a policy for required
password changes for the default and backup admin
account.</vulnerability_discussion>
  <posture>Win8</posture>
  <hippa>164.312(a)(2)(i),164.312(a)(2)(ii)</hippa>
  <pci>7.1.1</pci>
  <hbs_id>HBSPCATWIN80000055</hbs_id>
  <nist_800_53>AC-2(7)</nist_800_53>
  <iso_27001>A.8.3.3, A.11.2.1, A.11.2.2, A.11.2.4, A.11.5.2, A.11.5.5,
A.11.5.6</iso_27001>
```

Figure 3: *Leading section of the metabase showing three finding definitions. Each finding key is referenced by a UUID attribute. If a definition has a finding object associated with it, a class tag will be embedded in the finding's key with a UUID attribute linking it to the finding class definition that can instantiate the finding object. The rest of the elements in each finding key are arbitrary elements that indicate severity, discussion, group id, rule title, discussion, posture, and cross referenced compliance standards.*



PCAT2PY **pcat2pytest2**

COMPLIANCE	ID	H/F	DETAILS
Closed	**POSTURE:** 2008R2MS **GROUP ID:** V-26556 **HBS ID:** HBSPCAT2K8R2MS0000250 **NIST 800 53:** AU-2 ,AU-3,AU-8 **ISO 27001:** A.10.10.1, A.10.10.2, A.10.10.4, A.10.10.5, A.11.5.4, A.15.3.1, A.10.10.6, A.13.2.3 **PCI:** 10.3.4 **HIPPA:** 164.312(b)	Yes	**The system will be configured to audit ''System -> Security System Extension'' failures.** Maintaining an audit trail of system activity logs can help identify configuration errors, troubleshoot service disruptions, and analyze compromises that have occurred as well as detecting attacks. Audit logs are necessary to provide a trail of evidence in case the system or network is compromised. Collecting this data is essential for analyzing the security of information assets and detecting signs of suspicious and unexpected behavior. Security System Extension records events related to extension code being loaded by the security subsystem. Detailed auditing subcategories are configured in Security Settings \ Advanced Audit Policy Configuration. The summary level settings under Security Settings \ Local Policies \ Audit Policy will not be enforced (see V-14230).Set the policy value for Computer Configuration \ Windows Settings \ Security Settings \ Advanced Audit Policy Configuration \ System Audit Policies \ System \ "Audit Security System Extension" with ?Failure? selected. *Security System Extension* *Success=True*
Closed	**POSTURE:** 2008R2MS **GROUP ID:** V-15675 **HBS ID:** HBSPCAT2K8R2MS0000104 **NIST 800 53:** CM-6 **ISO 27001:** A.10.10.2 **PCI:** 2.2.4 **HIPPA:** 164.312(c)(1)	Yes	**Windows Registration Wizard will be turned off.** This check verifies that the Windows Registration Wizard is blocked from online registration. Set the policy value for Computer Configuration \ Administrative Templates \ System \ Internet Communication Management \ Internet Communication settings ?Turn off Registration if URL connection is referring to Microsoft.com? to ?Enabled?. *HKLM:\Software\Policies\Microsoft\Windows\Registration Wizard Control* *NoRegistration=1*
Open	**POSTURE:** 2008R2MS **GROUP ID:** V-26472 **HBS ID:** HBSPCAT2K8R2MS0000188 **NIST 800 53:** AC-5, AC-6, AC-6(2) **ISO 27001:** A.10.1.3, A.11.2.2, A.11.4.1, A.11.4.4, A.11.5.4, A.11.6.1, A.12.4.3 **PCI:** 7.1 **HIPPA:** 164.312(a)(1)	No	**Unauthorized accounts will not have the ''Allow log on locally'' user right.** Inappropriate granting of user rights can provide system, administrative, and other high level capabilities. Accounts with the "Allow log on locally" right can log on interactively to a system. This will be restricted to Administrators on servers. Set the policy value for Computer Configuration \ Windows Settings \ Security Settings \ Local Policies \ User Rights Assignment \ "Allow log on locally" as defined. *SeInteractiveLogonRight=* *BUILTIN\Administrators* *BUILTIN\Users* *BUILTIN\Backup Operators*
Closed	**POSTURE:** 2008R2MS **GROUP ID:** V-14243 **HBS ID:** HBSPCAT2K8R2MS0000078 **NIST 800 53:** CM-6 **ISO 27001:** A.10.10.2 **PCI:** 2.2.4 **HIPPA:** 164.312(c)(1)	Yes	**The system will require username and password to elevate a running application.** This check verifies that the system is configured to always require a user to type in a user name and password to elevate a running application. Set the policy value for Computer Configuration \ Administrative Templates \ Windows Components \ Credential User Interface ?Enumerate administrator accounts on elevation? to ?Disabled?. *HKLM:\Software\Microsoft\Windows\CurrentVersion\Policies\CredUI* *EnumerateAdministrators=0*
Closed	**POSTURE:** 2008R2MS **GROUP ID:** V-4443 **HBS ID:** HBSPCAT2K8R2MS0000331 **NIST 800 53:** AC-3, AC-3(3), AC-3(4) **ISO 27001:** A.7.2.2, A.10.6.1, A.10.7.3, A.10.7.4, A.10.8.1 A.10.9.1, A.10.9.2, A.10.9.3, A.11.2.2, A.11.5.4, A.11.6.1, A.12.4.3, A.15.1.3 **PCI:** 2.2.4 **HIPPA:** 164.312(c)(1)	Yes	**Unauthorized remotely accessible registry paths and sub-paths will not be configured.** The registry is a database for computer configuration information, much of which is sensitive. An attacker could use this to facilitate unauthorized activities. To reduce the risk of this happening, it is also lowered by the fact that the default ACLs assigned throughout the registry are fairly restrictive and they help to protect it from access by unauthorized users. Set the policy value for Computer Configuration \ Windows Settings \ Security Settings \ Local Policies \ Security Options \ ?Network access: Remotely accessible registry paths and sub-paths? as defined. *HKLM:\System\CurrentControlSet\Control\SecurePipeServers\Winreg\AllowedPaths* *Machine=* *System\CurrentControlSet\Control\Print\Printers* *System\CurrentControlSet\Services\Eventlog* *Software\Microsoft\OLAP Server* *Software\Microsoft\Windows NT\CurrentVersion\Print* *Software\Microsoft\Windows NT\CurrentVersion\Windows* *System\CurrentControlSet\Control\ContentIndex* *System\CurrentControlSet\Control\Terminal Server* *System\CurrentControlSet\Control\Terminal Server\UserConfig* *System\CurrentControlSet\Control\Terminal Server\DefaultUserConfiguration* *Software\Microsoft\Windows NT\CurrentVersion\Perflib* *System\CurrentControlSet\Services\SysmonLog*

Figure 4: *HTML report of a scan made by PCAT2PY against a selection of findings demonstrating findings being evaluated against the auditing settings, registry, and security settings.*

Requirements

To successfully use PCAT2PY, there are requirements that need to be met. These requirements vary depending on whether or not PCAT2PY will be used against remote hosts. For localhost use, the only requirement is that PCAT2PY must be run with administrative privileges. For use against remote information systems, there are further requirements that need to be satisfied for using SSH and WINRM.

PCAT2PY was developed and tested on Python 2.7.8. When operating PCAT2PY through the Python interpreter as a script, the following Python modules must be present: pycrypto[1], ecdsa[2], paramiko[3], and pywinrm[4]. These requirements as well as having a Python interpreter installed can be ignored when using the bundled implementations of PCAT2PY. Currently, there are three builds of PCAT2PY available. There are stand-alone, x86_64, binary releases for Red Hat and Windows. Finally there is a release of the original Python scripts composing PCAT2PY.

The following steps are taken to setup a Windows environment capable of running PCAT2PY through the Python interpreter.

1. Install python-2.7.8.amd64
2. Install cx_Freeze-4.3.3.win-amd64-py2.7
3. Install pycrypto-2.6.win-amd64-py2.7
4. Run python get-pip.py
5. Run python –m pip install ecdsa
6. Run python –m pip install paramiko
7. Run python –m pip install pywinrm

This will allow a Windows based system to run PCAT2PY directly or bundle it into a self contained executable using cx_Freeze[5]. If PCAT2PY is bundled, the metabase.xml file and the class directory must be copied into the bundled PCAT2PY’s working directory.

There are two requirements when scanning remote hosts over SSH. First, PCAT2PY must be connecting as root or preferably as a user with sudoers rights. Second, the “requiretty” default must be disabled in the remote host’s sudoers file. If PCAT2PY detects that it is connecting as a non-root user, it will attempt to elevate privileges through sudo. If PCAT2PY fails to do so, a custom exception is thrown and the execution is terminated.

There are two requirements when scanning remote hosts over WINRM. First, basic authentication must be enabled on the remote hosts. Second, WINRM must be configured to allow unencrypted connections. The current PCAT2PY implementation is limited to using basic authentication over unencrypted connections only.

Conclusion

The goal of this project was to develop and demonstrate a Python based, trusted pre-compliance tool. PCAT2PY is designed to be agent-less and require no installation to run. This application can be run locally or remotely against information systems that have WINRM or SSH configured and enabled according to the requirements guidance from this paper.

References

[1] D. C. Litzenberger. (2014, June 20). *pycrypto* [Online]. Available: https://github.com/dlitz/pycrypto, http://www.pycrypto.org/, https://www.dlitz.net/software/pycrypto/

[2] B. Warner. (2014, July 14). *python-ecdsa* [Online]. Available: https://github.com/warner/python-ecdsa

[3] J. Forcier. (2014, June 20). *paramiko* [Online]. Available: https://github.com/paramiko/paramiko/, http://docs.paramiko.org

[4] A. Diyan. (2014, Nov 2). *pywinrm* [Online]. Available: https://github.com/diyan/pywinrm

[5] A. Tuininga. (2014, May 3). *cx_Freeze* [Online]. Available: https://github.com/GreatFruitOmsk/cx_freeze, http://cx_freeze.readthedocs.org