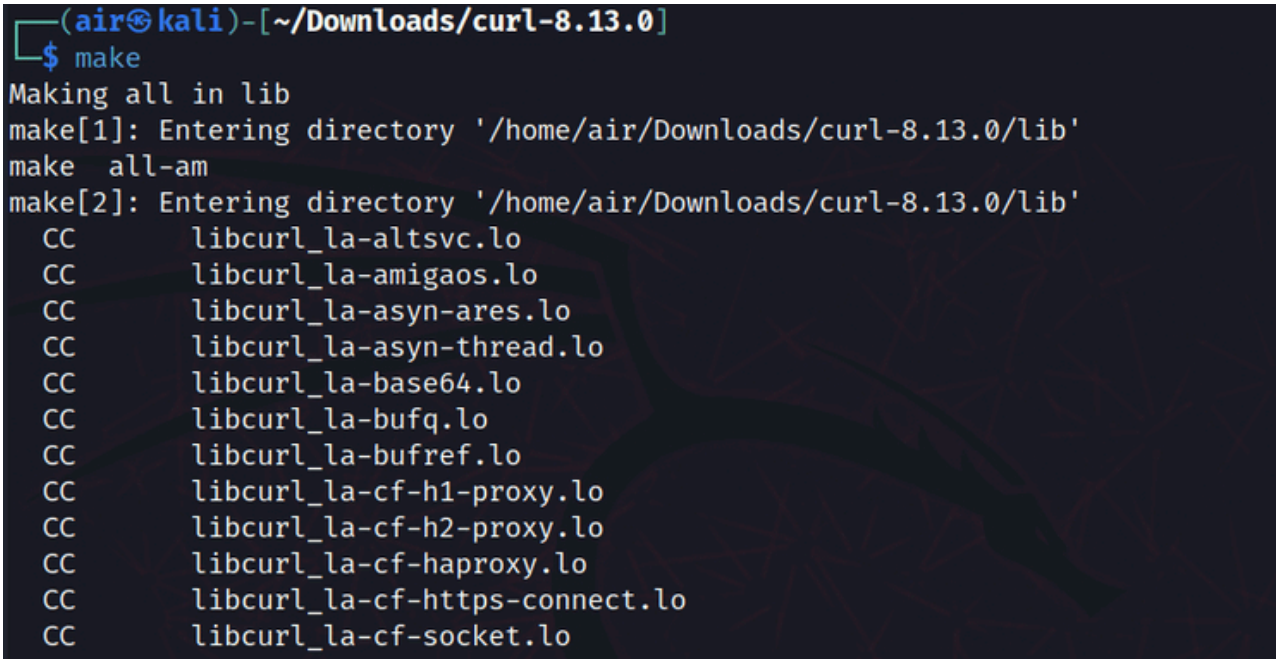
Compiling and Installing Software from Source in Linux – Hackers Arise

hackers-arise.com/linux-basics-for-hackers-the-make-command-compiling-and-installing-software-from-source-in-linux/

aircorridor



Welcome back, aspiring Linux enthusiasts!

In our journey through the Linux ecosystem, we often encounter situations where the software we need isn't available through standard package managers like apt or yum. Or perhaps we need a bleeding-edge version with features not yet available in the repositories. When this happens, knowing how to compile and install software from source becomes an invaluable skill in your Linux toolkit.

Make is a build automation tool that builds executable programs and libraries from source code by reading files called Makefiles. These files specify how to derive the target program from its dependencies. Once you understand this process, you'll have the freedom to install any open-source software on your Linux system.

Some of the key features of the **make** command include:

1. **Dependency Tracking**: Make only rebuilds what needs to be rebuilt, saving time during compilation.
2. **Parallel Execution**: Modern versions can compile multiple source files simultaneously, utilizing multi-core processors.
3. **Cross-Platform**: The make utility works across almost all Unix-like operating systems, making your skills transferable.
4. **Customization**: The compilation process can be customized to your specific needs and system configuration.

Let's download and compile a package from source!

Step #1 Preparing Your System for Compilation

Before we can compile any software, we need to ensure our system has the necessary tools. The compilation process requires a collection of development tools including compilers, linkers, and build utilities.

In Debian-based systems like Kali, you can install these tools with:

kali > sudo apt-get install build-essential

```
┌──(air㉿kali)-[~]
└─$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  build-essential
0 upgraded, 1 newly installed, 0 to remove and 910 not upgraded.
Need to get 4624 B of archives.
After this operation, 17.4 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 build-essential amd64 12.12 [
4624 B]
Fetched 4624 B in 1s (7212 B/s)
Selecting previously unselected package build-essential.
(Reading database ... 391444 files and directories currently installed.)
```

These packages include the gcc/g++ compilers and libraries required to compile software from source. Without these tools, you won't be able to proceed with compilation.

Once you have the development tools installed, you'll need to check if the software you're trying to compile has any dependencies. When using package managers, dependencies are handled automatically, but when compiling from source, you need to ensure all required libraries are available on your system.

For our example, we'll be compiling curl, a command-line tool for transferring data with URLs. While most systems come with curl pre-installed, compiling it ourselves will demonstrate the process that applies to most software packages.

Step #2 Downloading and Extracting the Source Code

Let's download the curl source code:

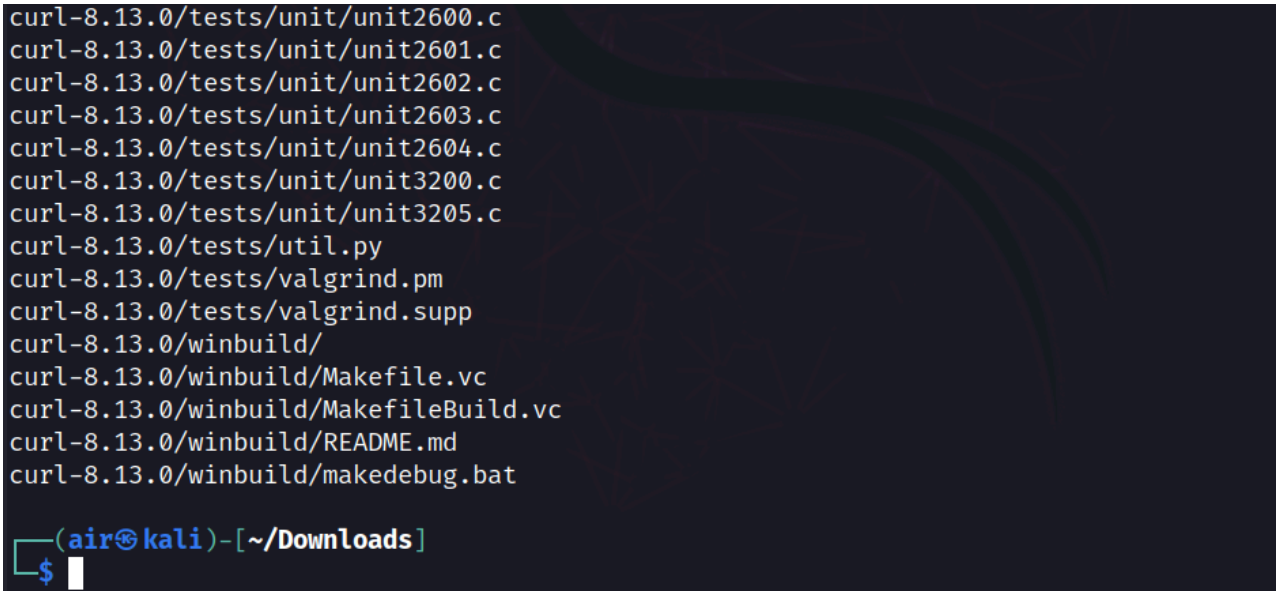
kali > wget -O curl.tar.gz https://curl.se/download/curl-8.13.0.tar.gz

```
┌──(air㉿kali)-[~/Downloads]
└─$ wget -O curl.tar.gz https://curl.se/download/curl-8.13.0.tar.gz
--2025-04-22 12:01:44--  https://curl.se/download/curl-8.13.0.tar.gz
Resolving curl.se (curl.se)... 151.101.193.91, 151.101.1.91, 151.101.129.91, ...
Connecting to curl.se (curl.se)|151.101.193.91|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4188717 (4.0M) [application/x-gzip]
Saving to: 'curl.tar.gz'

curl.tar.gz           100%[=====================>]   3.99M  10.0MB/s    in 0.4s

2025-04-22 12:01:44 (10.0 MB/s) - 'curl.tar.gz' saved [4188717/4188717]
```

This command downloads the curl source code and saves it as curl.tar.gz in your current directory.

Next, we need to extract the contents of this file:

kali > tar -xvzf curl.tar.gz



The tar command with these options (-xvzf) extracts (-x) the contents of the archive, verbosely (-v) showing the files being extracted, from a gzipped (-z) file (-f).

After extraction, you'll have a new directory containing the source code. Let's navigate into it:

kali > cd curl-8.13.0

```
┌──(air㉿kali)-[~/Downloads]
└─$ cd curl-8.13.0

┌──(air㉿kali)-[~/Downloads/curl-8.13.0]
└─$ ls -l
total 2272
-rw-r--r-- 1 air air     438 Apr  2 01:47 CHANGES.md
drwxr-xr-x 2 air air    4096 Apr  2 01:47 CMake
-rw-r--r-- 1 air air   92232 Apr  2 01:47 CMakeLists.txt
-rw-r--r-- 1 air air    1088 Apr  2 01:47 COPYING
-rw-r--r-- 1 air air    1711 Apr  2 01:47 Dockerfile
-rw-r--r-- 1 air air    7167 Apr  2 01:47 Makefile.am
-rw-r--r-- 1 air air   39872 Apr  2 01:47 Makefile.in
-rw-r--r-- 1 air air    1664 Apr  2 01:47 README
-rw-r--r-- 1 air air   32291 Apr  2 01:47 RELEASE-NOTES
-rw-r--r-- 1 air air   44533 Apr  2 01:47 acinclude.m4
-rw-r--r-- 1 air air   45906 Apr  2 01:47 aclocal.m4
-rwxr-xr-x 1 air air    7400 Apr  2 01:47 compile
-rwxr-xr-x 1 air air   49482 Apr  2 01:47 config.guess
-rwxr-xr-x 1 air air   35406 Apr  2 01:47 config.sub
-rwxr-xr-x 1 air air 1320462 Apr  2 01:47 configure
-rw-r--r-- 1 air air  161240 Apr  2 01:47 configure.ac
-rw-r--r-- 1 air air    4781 Apr  2 01:47 curl-config.in
-rwxr-xr-x 1 air air   23568 Apr  2 01:47 depcomp
drwxr-xr-x 6 air air    4096 Apr  2 01:47 docs
drwxr-xr-x 3 air air    4096 Apr  2 01:47 include
-rwxr-xr-x 1 air air   15358 Apr  2 01:47 install-sh
drwxr-xr-x 6 air air   12288 Apr  2 01:47 lib
-rw-r--r-- 1 air air    1586 Apr  2 01:47 libcurl.pc.in
-rwxr-xr-x 1 air air  333057 Apr  2 01:47 ltmain.sh
```

Inside this directory, you'll find numerous files and subdirectories containing the source code and build instructions for curl.

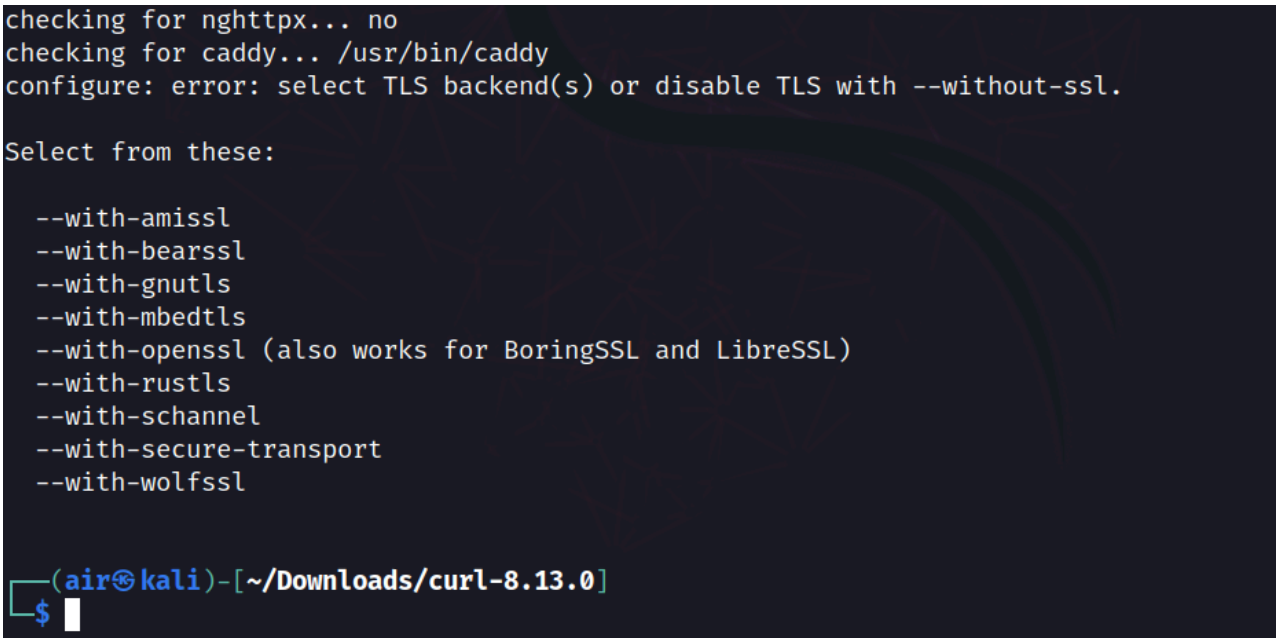
Step #3 Configuring the Build Environment

Before compiling, we need to configure the build environment. This is done using the configure script, which is designed to help programs run on a wide variety of different computer systems.

The configure script checks your system for required dependencies, sets up the build environment, and creates a Makefile tailored to your specific system. This is a critical step that ensures the software will compile correctly on your particular Linux distribution.

Let's run the configure script:

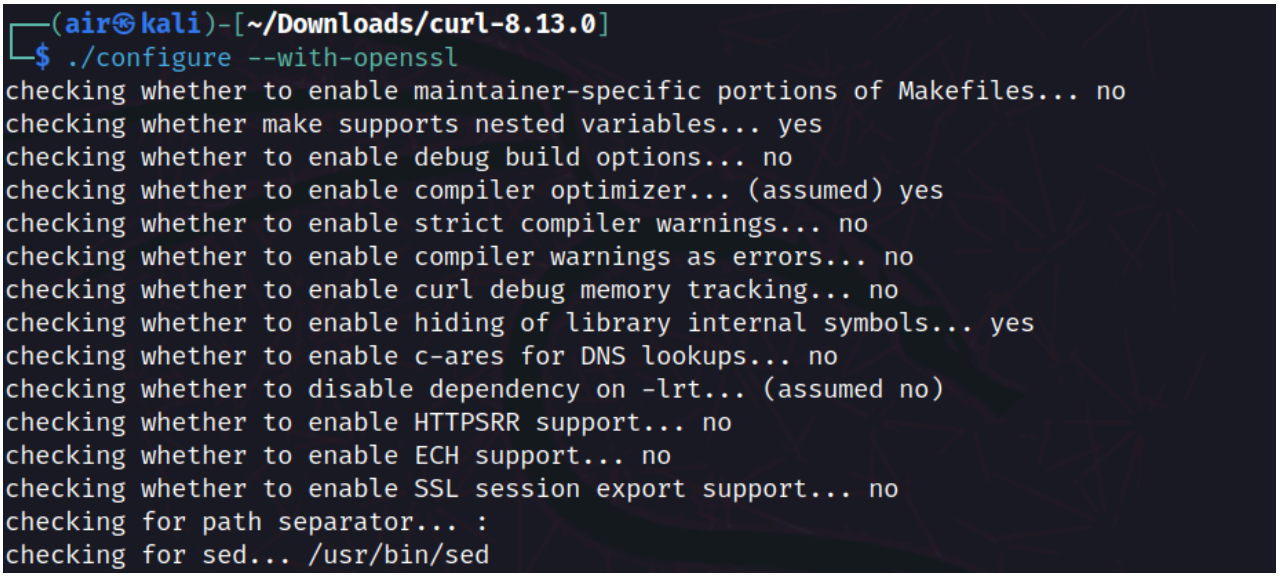
kali > ./configure



This command runs the configure script in the current directory. The script will perform numerous checks on your system, looking for required libraries and tools, and setting up the build environment accordingly.

As you can see, we received an SSL error. We can choose any option from the list, I will use OpenSSL:

kali> ./configure –with-openssl



The configure script essentially matches the libraries required by the program with the ones installed on your system. By doing this, the compiler knows where to look for the libraries required by curl. When it's done, it generates a file called Makefile with all the necessary information for the compilation process.

During this process, you may see an error about a missing library, to continue you just need to install it through apt and repeat the script.

The configure script often accepts various options that allow you to customize the build. For example, you might want to install the software in a different location or enable/disable certain features (just like we did with SSL). You can usually see these options by running:

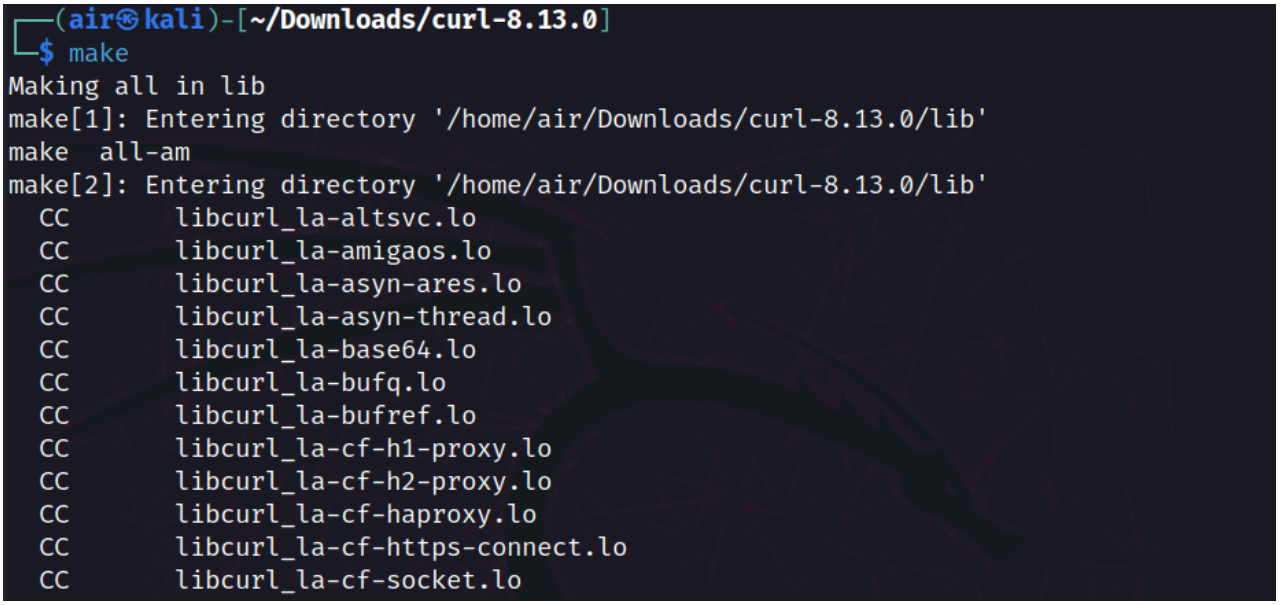
kali > ./configure –help

```
┌──(air㉿kali)-[~/Downloads/curl-8.13.0]
└─$ ./configure --help
`configure' configures curl - to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking ...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']

Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]
```

Step #4 Compiling the Source Code with Make

Now that we have configured the build environment and generated the Makefile, we can compile the source code using the make command:

kali > make



The make command reads the instructions in the Makefile and executes the commands necessary to compile the software. This process can take anywhere from a few seconds to several hours, depending on the size and complexity of the software.

During compilation, you'll see a lot of output as the compiler processes each source file. Don't worry if you don't understand all of this output – it's primarily useful for debugging if something goes wrong.

The make command is incredibly powerful because it only rebuilds what needs to be rebuilt. If you make changes to the source code and run make again, it will only recompile the files that have changed, saving considerable time during development.

Step #5 Installing the Compiled Software

After successfully compiling the software, the final step is to install it on your system. This is done using the make install command, which typically needs to be run with root privileges:

kali > sudo make install

```
┌──(air㉿kali)-[~/Downloads/curl-8.13.0]
└─$ sudo make install
Making install in lib
make[1]: Entering directory '/home/air/Downloads/curl-8.13.0/lib'
make[2]: Entering directory '/home/air/Downloads/curl-8.13.0/lib'
 /usr/bin/mkdir -p '/usr/local/lib'
 /bin/bash ../libtool   --mode=install /usr/bin/install -c   libcurl.la '/usr/local/l
ib'
libtool: install: /usr/bin/install -c .libs/libcurl.so.4.8.0 /usr/local/lib/libcurl.s
o.4.8.0
libtool: install: (cd /usr/local/lib && { ln -s -f libcurl.so.4.8.0 libcurl.so.4 || {
 rm -f libcurl.so.4 && ln -s libcurl.so.4.8.0 libcurl.so.4; }; })
libtool: install: (cd /usr/local/lib && { ln -s -f libcurl.so.4.8.0 libcurl.so || { r
m -f libcurl.so && ln -s libcurl.so.4.8.0 libcurl.so; }; })
libtool: install: /usr/bin/install -c .libs/libcurl.lai /usr/local/lib/libcurl.la
libtool: install: /usr/bin/install -c .libs/libcurl.a /usr/local/lib/libcurl.a
libtool: install: chmod 644 /usr/local/lib/libcurl.a
libtool: install: ranlib /usr/local/lib/libcurl.a
```

This command copies the compiled binaries, libraries, and documentation to their appropriate locations on your system, as specified in the Makefile. The installation locations are typically determined during the configure step.

After installation, you should be able to run the software. For curl, you can verify the installation by checking its version:

kali > curl -V

```
┌──(air㉿kali)-[~/Downloads/curl-8.13.0]
└─$ curl -V
curl 8.13.0 (x86_64-pc-linux-gnu) libcurl/8.12.1 OpenSSL/3.4.1 zlib/1.3.1 brotli/1.1.
0 zstd/1.5.6 libidn2/2.3.7 libpsl/0.21.2 libssh2/1.11.1 nghttp2/1.64.0 nghttp3/1.6.0
librtmp/2.3 OpenLDAP/2.6.9
Release-Date: 2025-04-02
Protocols: dict file ftp ftps gopher gophers http https imap imaps ipfs ipns ldap lda
ps mqtt pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp ws wss
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTP3 HTTPS-proxy IDN IPv6 Kerb
eros Largefile libz NTLM PSL SPNEGO SSL threadsafe TLS-SRP UnixSockets zstd
WARNING: curl and libcurl versions do not match. Functionality may be affected.
```

This should display the version of curl you just compiled and installed, along with information about the supported features and protocols.

Summary

Remember, while package managers offer convenience, knowing how to compile from source gives you flexibility and control that package managers can't match. It's an essential skill for any serious Linux user, administrator or hacker.

To learn more about Linux check out Linux Basics for Hackers training.

aircorridor
AirCorridor is a seasoned cybersecurity specialist with a Masters in Cybersecurity of Information Technologies and Systems and expertise in Physical Security. With experience in secure government platforms, advanced Linux administration and

professional cyber operations, he provides expert training and consultancy. As a Hackers-Arise contributor, he stays ahead of emerging threats and empowers individuals and businesses with cutting-edge security strategies and privacy-focused solutions.