Lynis

Lynis Installation and Usage

Introduction

Lynis is an open source security tool. It helps with auditing systems running UNIX-alike systems (Linux, macOS, BSD), and providing guidance for system hardening and compliance testing. This document contains the basics to use the software.

Installation

The installation of Lynis is explained in the Get Started guide.

Using Lynis

Basics

By running 'lynis' the program is started and will provide the basic parameters available. If you manually extracted Lynis (or used Git), then use './lynis' to start the program from the local directory.

The most common command to start Lynis is using **audit system** command. This still start the security scan.

To run Lynis you should meet one requirement: have write access to **/tmp** (temporary files)

Commands, Options, and Arguments

Introduction

The behavior of programs can influenced with commands, arguments, and options. These terms are often mixed up, so we start with a quick introduction.

Commands tell the program *what* to do. An option tells the program *how* to do it. If an argument is used, it tell *on what* it applies. Arguments usually follow an option, like a filename, or a target.

Example

```
$ ./lynis audit system --quick --auditor "The Auditor"
```

In this example we tell Lynis to **audit** (command), with the target **system** (argument). By using the **--quick** (option), we tell it not to wait. We used **--auditor** (option) and defined it as **"The Auditor"** (argument).

Lynis Commands

The Lynis tool requires a minimum amount of parameters to run. If you are using it for the first time, just run lynis and see what output it provides.

```
$ ./lynis
```

Without any commands, Lynis will display its status, together with suggestions on how to start.

Audit

The *audit* command tells Lynis to perform an audit.

Targets include:

- **system** - audit the host system
- **dockerfile** - audit a dockerfile

Show

The *show* command informs Lynis to share information, like help or the value of something.

Examples:

- **help** - show help and tips
- **profiles** - show discovered audit profiles
- **settings** - show active settings
- **version** - show Lynis version

Parameters

In the table below, the most commonly used parameters are listed.

Parameter	Abbreviated	Description
--auditor "Name"		Assign an auditor name to the audit (report)
--checkall	-c	Start the check
--check-update		Check if Lynis is up-to-date
--cronjob		Run Lynis as cronjob (includes -c -Q)
--help	-h	Shows valid parameters
--manpage		View man page
--nocolors		Do not use any colors
--pentest		Perform a penetration test scan (non-privileged)
--quick	-Q	Don't wait for user input, except on errors
--quiet		Only show warnings (includes --quick, but doesn't wait)
--reverse-colors		Use a different color scheme for light backgrounds
--version	-V	Check program version (and quit)

Tips

- If Lynis is not installed as package (with included man page), use **--man** or **nroff -man ./lynis.8**
- For systems where the shell background is light, use **--nocolors** or **--reverse-colors**
- Use **lynis show options** to see all available parameters of Lynis

Profiles

Lynis uses profiles to have a set of predefined options for your operating system and preferences. If you don't provide a profile (--profile <name>), the default profile (default.prf) will be used. You are advised to copy the default.prf and adjust it to your needs.

With the usage of profiles, you can make a template/baseline for different types of systems.

Examples:

- Profile per operating system (Debian Linux, RedHat Linux, OpenBSD)
- Profile per system roles (mail server, web server)
- Profile per security level (low, medium, high level)

HostIDs

During the security audit, Lynis attempts to assign two identifiers to the system. They can be compared as fingerprints and can be used in other tools and to link data to an existing system.

Identifiers: hostid and hostid2

The first identifier is named **hostid** and has a length of 40 characters. The MAC address of the system is typically used its data input. The second identifier is **hostid2**. It is 64 characters long and typically uses a public SSH key a data input.

```
lynis show hostids
```

Overriding the identifiers

In case your system can not generate the host identifiers automatically, then you can specify them in your custom profile (custom.prf). This can also be useful when systems are short-lived, yet you want to link the same data to such instance.

```
lynis configure settings hostid=$(head -c 64 /dev/random | sha1sum | awk '{print $1}'):hostid2=$(head -c 64 /dev/random | sha256sum | awk '{print $1}')
```

One-liner to generate both IDs and add them to the configuration

Important notes

- Only override the host identifiers when really needed
- Make sure that the identifiers are unique for every individual system
- When using this option, **both** values for *hostid* and *hostid2* need to be set in the profile

Screen output

While Lynis scans a system it will perform single target tests and output the result of every (performed) test to the screen. Every scan result has to be interpreted by the auditor and (re)checked what it means.

Behind most tests, it will output [**OK**] or [**WARNING**], where the first one is considered an expected (good) result, the second one unexpected. However, keep in mind that a result saying "[**OK**]" does NOT always mean the scanned target is correctly configured, safe (security wise) or a best practice.

On the opposite, every "[**WARNING**]" doesn't have to be 'bad', since systems (and their requirements) are different. However, as auditor you are adviced to pay attention to them and check what influence the test has on your system or policy.
Actions you can take after getting a warning:

- Fix the problem
Read the log file about the technical background (often it contains a suggestion at the test), consult internet sources and documentation about what the impact of the change can be.

- Disable the test (whitelisting)
Within the scan profile, tests can be completely disabled (option **test_skip_always**). When you

have a test which gives a warning and you are not interested in the result of that particular test, you can ingore it.

For example:
You have only one DNS server configured on your workstation. A test shows a warning and reveals that it expects at least two working name servers. In such case you can choose not to get informed about it and disable the test. Extend the option *test_skip_always* in your scanning profile with the test number (which can be found in the log file or at the end of the Lynis screen output).

After every scan, the auditor should consult the log file (/var/log/lynis.log) and interpreter the results. If tests are displayed as a "[**WARNING**]", the log file will give the reason why a warning was displayed. In most cases a "Suggestion:" line will be present, to assist in resolving the issue or give more information what was tested (or expected).

Suggestions and Warnings

The screen output, as outlined in previous section, will provide the status of most tests on screen. During the audit proces, Lynis will gather any possible suggestion or warning. These results will be grouped and displayed at the bottom of screen output. Usually warnings are events which really need an action.

Suggestions on the other hand could indicate room for improvement. It's common to find much more suggestions than warnings. This does not imply that because there are many suggestions (and no warnings) that a system is properly secured!

To determine what has been checked together with the related suggestion/warning, the test identifier is displayed on the same line (between brackets). Open the lynis log file (/var/log/lynis.log) and search for this identifier.

Plugins

Lynis plugins are extensions to the Lynis core. Where normal Lynis controls perform individual tests and share the outcome, plugins will usually just gather information. This information is then collected and processed in bulk. The big benefit is that is quicker and more powerful. For example security intelligence can be applied by collecting data and correlating it on the central node.

Plugin phases

Lynis has modular support to extend basic functionality by using plugins. Plugins are executed in several phases:

Plugins: Phase 1
Plugins which do use hooks into existing tests, or gather data for later processing, will use phase 1 to initialize. Some tests which are part of the plugin will then finish in phase 2.

Plugins: Phase 2

After running all tests, plugins get a last chance to do their job. For example parse discovered elements on the system, like a virtual host within Apache.
Plugins which can be used standalone (e.g. no hooks, no input from existing tests), can be executed in phase 1. No need for a phase 2 component.

Enabling plugins:
Plugins can be enabled by using the plugin option within the profile.
Example: plugin=<custom_myplugin>

Plugin directory
The directory in which plugins can be stored is determined by Lynis. By default it tries a few paths (/usr/local/lynis/plugins /usr/local/share/lynis/plugins /usr/share/lynis/plugins and /etc/lynis/plugins). If these directories are not found, then the local work directory is being used. To use a different directory, use the **--plugin-dir** parameter, followed by the directory name.

Custom plugins
When creating personal plugins, you are adviced to add a personal prefix and making the file name unique (ie. custom_myplugin). This prevents the file being overwritten at a new release. If you just want an invidual test, you are advised to use the custom_test file instead, as plugins have a different goal.

If you consider writing a plugin, we ask you to contact us to determine the possibilities.

5. Reporting and Logging

Lynis supports one report format, which can be used to gather results and display them in a custom or (more) friendly presentation. The report file can also be used to compare scan results from the past with a current scan. Lynis Enterprise on the other hand has much more possibilities to display data, including extended reports in several formats.

Contents of report file:
- Remarks: #<remark>
- Section: [<section name>]
- Option/value: <option name>=<value of option>

When an option may have multiple values (like installed packages for example), brackets ([]) are added. Example: installed_package[]=Package-1.0.0

Logging
When a system is scanned and results are displayed, additional debugging information will be added to the log file (default: /var/log/lynis.log). For advanced testers this information will be useful to see what the program did in the background or where anomalies showed up (and often why).

Information in the log file:

- Time of an action/event
- Reason(s) why a test failed or will be skipped
- Output of (internal) tests and sub tests
- Suggestions about configuration options or how to fix/improve things
- Threat/impact score

Remark: the log file will be purged every scan. If you need debugging or logging information for previous scans, schedule log rotation or make a backup before running Lynis again.

6. Integration with Lynis Enterprise

Lynis data can be uploaded via the **--upload** option. For companies using multiple systems, the Lynis Collector is usually the preferred option. This specific tool has more capabilities and features to deal with batches of data. Both the --upload parameter and Lynis Collector, are meant to upload data to the central node.

Upload via Lynis

Add the license key in in the scan profile (default.prf or your customized profile). After adjusting, use the --upload parameter to upload the data after scanning.

For data uploads, the cURL utility is being used. For environments which require a proxy, the profile allows you to define any options given to cURL, like --*proxy*. In case a self-signed certificate is being used on the central node, you can disable full certificate checking with the --*insecure* option.

Upload via Lynis Collector

An alternative option is to use Lynis Collector. This tool allows uploads from a central machine with an internet connection available. This way other machines within the network don't need direct access. Lynis Collector is available for customers in the downloads section. See part II for more details about this component.

Part II: Lynis Collector

Introduction

The Collector component within the Lynis suite, is a supporting tool. It collects reports from many systems and uploads it in one batch. It is available to users of the Enterprise solution. Only one Collector is usually needed within the network.

See the Lynis Collector documentation for full details.

Part III: Lynis Enterprise

Introduction

Requirements

To use Lynis Enterprise, you need a license key and an active user account. An account can be created during the ordering process or before. Management happens with a web based solution. No external plugins have to be installed to work with our solution.

For auditing purposes, Lynis needs to be configured on the systems that are to be audited. Optionally the Lynis Collector can be used to collect data, then upload it to the central system.

Please refer to Section I and Section II to install and configure these components.

14. Account Management

An account on the Lynis Enterprise service provides access to the systems and configuration of a company. Each account can be linked only to one company.

15. Management of systems

Adding new system

During the life cycle of a system, it can be easily added by uploading the data to Lynis Enterprise. If the unique ID of the system is not known yet, the system will be added to the pool of machines.

Removing

When a system is being decommissioned, the absence of new scan data will be noticed and a related event will be raised. If the system is to be removed altogether, select the system and use the delete icon (🖥).

16. Security hardening

One of the main components of Lynis and Lynis Enterprise is to provide guidance in security hardening of systems. Per system the related security controls are listed. The Enterprise edition includes an **implementation plan**, customized to your environment. This way help is provided to select which controls.

17. Compliance and Baselines

Baselines provide a means to check for a defined policy and report about the compliance with the policy. Each system can be linked to a baseline of choice.

18. Change management

More details will follow later

19. Event handling

Events within the Lynis Enterprise solution are used to inform the administrator(s) about a specific activity. This includes exceeding a warning threshold or for example missing data. Usually an event is a call to action and check the related component.

20. Software upgrades

Since Lynis will be often updated to support new tests and software, performing regular updates is advised. Depending on the installation method there are two options to stay up-to-date:

Option 1. Using software packages
If the platform(s) used provides an up-to-date Lynis package, it could be used as part of your software upgrade strategy.

In case the vendor maintains only "stable" releases, they will usually not release newer versions of Lynis, until the next OS release. Then you might consider creating a Lynis package yourself. See the Tips section on how to create a package.

Option 2. No install
Instead of installing Lynis at all, a cronjob could be used to fetch the latest Lynis package from an internal server (e.g. HTTP/FTP/SCP/NFS). The cronjob first extracts the tarball into a temporary directory. Then it runs Lynis from there, and before cleaning up, it sends the data to the Lynis Collector.

Updating to a new release would mean the administrator will test the new version first on a few systems and then upload it to the central location. From that very moment all systems will be using the latest version.

Tips and Suggestions

Staying up-to-date

Staying up-to-date with software is important to have access to the latest functionality and make sure that known bugs have been solved. Since Lynis is an auditing tool, it's possibly even more important to keep up with the latest version. Right now we don't provie auto-updating yet. We believe strongly that people should test software releases before applying them into production.

To get notified when new releases are available, the following options are available:

Notification list
At our Downloads page you can subscribe to the notification list. When a new release is available, you get an email with the changes.

Twitter
Follow founder Michael Boelen (@mboelen) and our company @cisofy_is

Lynis --check-update
For monitoring purposes or to notify yourself, it's possible to parse the output of Lynis while it checks for the latest version. If the output is "Outdated", a new version is available. Background information: Lynis uses DNS to check for the latest version by using a TXT record query.

Building your own packages

For companies or individuals who prefer their own packages, there is a lynis.spec file available. Building a RPM is very easy due to the low number of dependencies of Lynis and consists of the following steps:

Package creation:
To create a custom package for installation on your machine(s):
- Download lynis.spec file (see project page)
- Adjust version number and if needed, paths
- Run 'rpmbuild -ta lynis-version.tar.gz' to build the RPM package
- Install package by running: rpm -ivh <filename>

Error: You have to be root (or equivalent) to perform an audit. Please su(do) and try again.

Lynis needs to be executed as root for a full audit. Change to the root and execute Lynis again. Sudo might work as well. If you don't have root permissions, use the --pentest option.

Error: Change ownership of ./include/const and ./include/functions to 'root'

To protect alteration of the files, Lynis perform a few security checks. If the related files are not owned by root, or their permissions are not strict enough, Lynis will show this on screen, including the commands to fix it. Usually it is caused because files were untarred by a user other than root.

Part V: Support

Lynis is tested on the most common operating systems. The documentation (README, FAQ) and the debugging information in the log file usually provides the answer to most questions (or problems). Bugs can be reported by contacting us. **Commercial support is available.**

Part VI: Frequently Asked Questions

Is Lynis really free?

Yes, Lynis is open source and free to use. By default it comes without warranties or support, as described in the Lynis package. If you prefer support, then integration with Lynis Enterprise is better suitable for your needs.

Is Lynis restricted in functionality, compared to Enterprise version?

There are no limitations regarding functionality. Lynis is also part of the Enterprise version, therefore it has full functionality. Companies benefit from using the Enterprise version, as it includes additional plugins.

What systems are supported?

All common systems based on Unix/Linux are supported. Examples include Linux, AIX, *BSD, HP-UX, macOS and Solaris.
For package management are the following tools supported:
- dpkg/apt, pacman, pkg_info, RPM, YUM, zypper

What is the difference between Lynis and Lynis Enterprise?

Lynis is an open source auditing tool, focused on auditing single Linux or Unix based systems.

Lynis Enterprise is a centralized auditing system, with additional reporting, ready-to-use hardening scripts, monitoring and dashboards. Primary benefits is saving time by automation and always having up-to-date reports at hand.

Can I create my own tests?

Sure you can! Lynis has a test category named "custom" (filename tests_custom). If this file exists, it will execute your own defined tests.
While creating your own tests is totally fine, please consider if others could benefit from them as well by sharing them with us. We accept most tests and give you the appropriate credit (or keep it anonymous if preferred).

The colors used are hard to read with my white background, how can I solve this?

Disable color usage or use the --reverse-colors option

What is the difference between a normal test and a plugin?

While both are similiar in what they can do, a test has the main goal of performing a check and directly form a conclusion (something is present or not, the outcome is good or bad etc). The purpose of plugins is to collect data for later use. In particular the Lynis Enterprise solution will use plugins to collect extra data which will be later analyzed. One example would be to determine exceptions or outliers. It would not make sense to have everyone build up databases of data, while all information is already centrally stored.

Is it possible to become reseller of Lynis?

Yes. Please contact us for the possibilites.

How can I verify the downloads

After downloading, test the file to confirm the integrity of the download. The related SHA1 and SHA256 hash are provided on the website as well. Depending on your OS, this can be performed with the command sha1, sha1sum or with openssl.

```
 $ sha1sum lynis-version.tar.gz
 $ sha256sum lynis-version.tar.gz
 $ sha1 lynis-version.tar.gz
 $ sha256 lynis-version.tar.gz
 $ openssl sha1 lynis-version.tar.gz
 $ openssl sha256 lynis-version.tar.gz
```

The resulting hash displayed should be the same as on the website. If not, try downloading it on another machine or via a browser, to confirm the download was not corrupted. If the hash still shows a different value, please contact us.

4. Verification - Signature

If you have GnuPG installed on your system, you can download our public key (https://cisofy.com/files/cisofy-software.pub) and the related signature of the download itself. The signature itself is a file with the extension .asc.

```
$ wget https://cisofy.com/files/cisofy-software.pub
$ gpg --import cisofy-software.pub
$ gpg --list-keys --fingerprint
/home/user/.gnupg/pubring.gpg
------------------------
pub   4096R/D5B79251 2014-11-04 [expires: 2030-10-31]
      Key fingerprint = 73AC 9FC5 5848 E977 024D  1A61 429A 566F D5B7 9251
uid                  CISOfy (Software Signing Key) <security@cisofy.com>
sub   4096R/6E8847E1 2014-11-04 [expires: 2030-10-31]
```

Verify that the results are the same as listed above. Next step is verifying the download:

```
$ gpg --verify lynis-version.tar.gz.asc lynis-version.tar.gz
gpg: Signature made Wed 05 Nov 2014 12:39:26 AM CET using RSA key ID D5B79251
gpg: Good signature from "CISOfy (Software Signing Key) <security@cisofy.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 73AC 9FC5 5848 E977 024D  1A61 429A 566F D5B7 9251
```

Note: The warning is displayed as the system simply does not not know if it is trusted. As there is no central authority to validate, you can mark the key as trusted yourself. Before doing so, we have some tips to make sure you are using the right key:

- Compare the key output you get, with the results on this page
- Check DNS entry cisofy-software-key.cisofy.com, it should give the same fingerprint.

```
$ host -t txt cisofy-software-key.cisofy.com
cisofy-software-key.cisofy.com descriptive text "Key fingerprint =
73AC 9FC5 5848 E977 024D  1A61 429A 566F D5B7 9251"

$ gpg --edit-key security@cisofy.com
gpg (GnuPG) 1.4.16; Copyright (C) 2013 Free Software Foundation, Inc.
```

```
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  4096R/D5B79251  created: 2014-11-04  expires: 2030-10-31  usage: SC
                     trust: unknown       validity: unknown
sub  4096R/6E8847E1  created: 2014-11-04  expires: 2030-10-31  usage: E
[ unknown] (1). CISOfy (Software Signing Key) <security@cisofy.com>

gpg> trust
pub  4096R/D5B79251  created: 2014-11-04  expires: 2030-10-31  usage: SC
                     trust: unknown       validity: unknown
sub  4096R/6E8847E1  created: 2014-11-04  expires: 2030-10-31  usage: E
[ unknown] (1). CISOfy (Software Signing Key) <security@cisofy.com>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

Your decision? 5
Do you really want to set this key to ultimate trust? (y/N) y

pub  4096R/D5B79251  created: 2014-11-04  expires: 2030-10-31  usage: SC
                     trust: ultimate      validity: unknown
sub  4096R/6E8847E1  created: 2014-11-04  expires: 2030-10-31  usage: E
[ unknown] (1). CISOfy (Software Signing Key) <security@cisofy.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg> quit
```

Now the key has be marked as being trusted and the related warning will be gone when verifying our downloads.

Note: ultimate trust is the most extensive type of trust in a key. You may also use option 4 (I trust fully).