Secure Web Server Workshop

Xavier Belanger
Wilmington IT Security Discussion Group
May 24 2025

Welcome!

- Thank you for attending today. By the end of this workshop, you should have a better understanding on how to secure a web server running on a Linux system in the cloud.
- All steps are documented, and instructions will be provided as needed.

Who am I?

- Xavier Belanger
- Formerly system administrator, network administrator, currently working as an IT security architect.
- Ran its first web server in 1999.

Ice breakers

- Hello, my name is _____
- I’m a _____
- And my hobby is _____

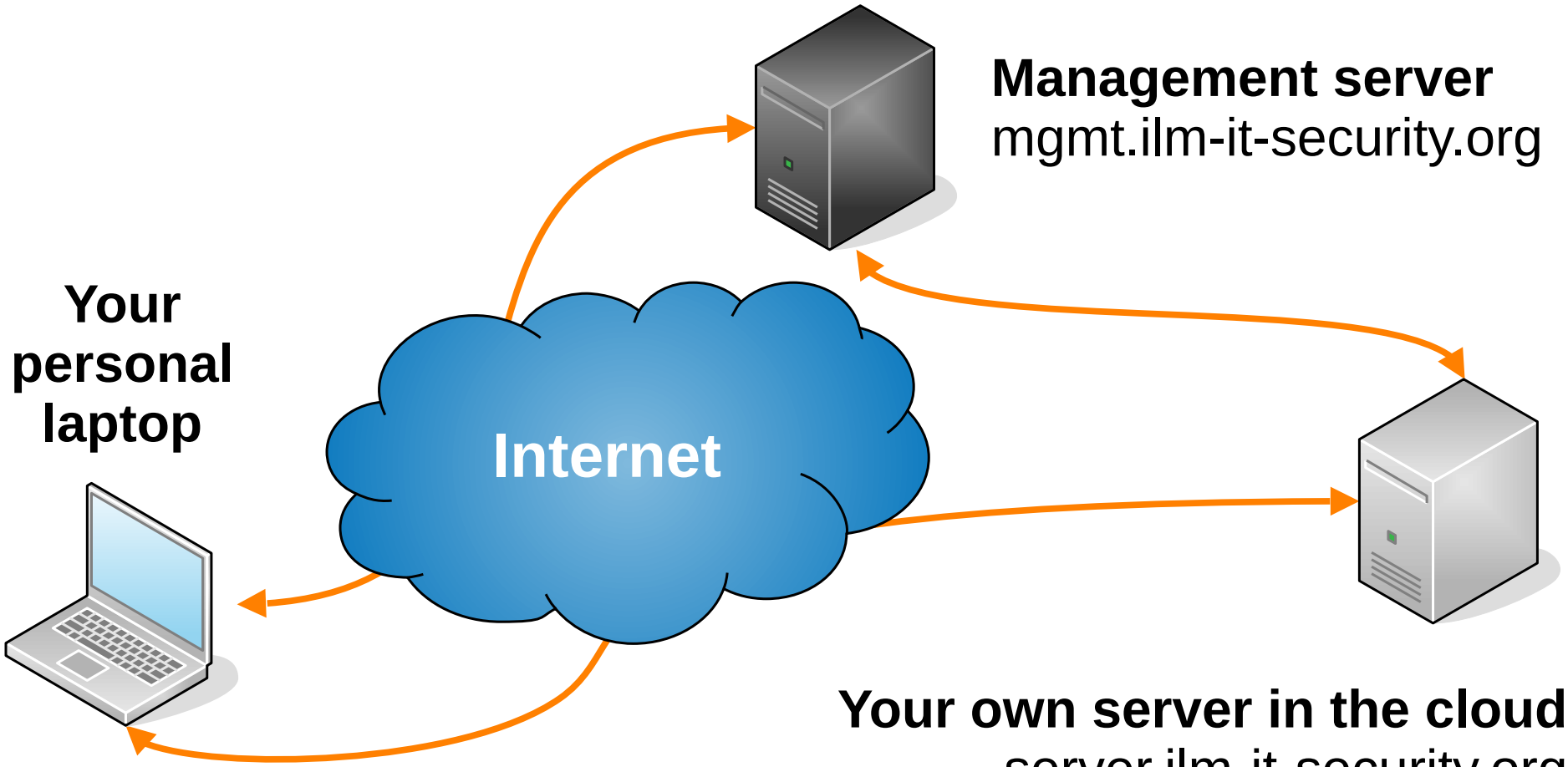
Scope

- This workshop is focused on securing a web server and the hosting operating system.
- Coding practices, use of databases and similar topics are not covered, due to time constraints.

Common rules

- Do not hesitate to ask questions, make comments or request to review a specific point in detail.
- Feel free to share your experiences, discuss alternatives and bring your point of view.

Our lab environment



Additional tools

- Few additional scripts are provided for this workshop (not part of the regular Ubuntu distribution).
- `mkbackup.sh`
- `mkwebsite-http.sh`, `mkwebsite-https.sh`
- `LE-certs-cleanup.sh`, `LE-certs-copy.sh`, `LE-certs-renew.sh`
- `start-ssh-agent.sh`
- `nftables.conf`, `webtraffic-add.sh`, `webtraffic-delete.sh`, `check-firewall.sh`, `check-logs.sh`, `reset-firewall.sh`

Accessing your new server

- A Linux server has been created for each participant in the Linode cloud environment.
- Your first step is to confirm access to this server and make sure that you can fully manage it.

First SSH connections

- `ssh username@mgmt.ilm-it-security.org`
- `ssh username@server.ilm-it-security.org`
- You will get a warning at your very first connection to confirm that you're trying to reach each server.
- At this point you can choose to connect via the management server to access your personal server or not. The management server will be used to run tools that you may not have on your personal computer.

Validating your account

- Check the account that you are using:
- `whoami`
- Check that you have administrator level access:
- `sudo whoami`

Confirming the server name

- In order to host a web site, your server needs a name registered in DNS (Domain Name System).
- For this workshop the DNS configuration has already been set for you.
- Each server is set with two names:
  - `server.ilm-it-security.org`
  - `www.server.ilm-it-security.org`

IP and DNS commands

- `ip addr show`
- `dig A server.ilm-it-security.org`
- `dig A www.server.ilm-it-security.org`
- `dig AAAA server.ilm-it-security.org`
- `dig AAAA www.server.ilm-it-security.org`
- `dig -x <IP address>` (public IPv4)

Updating the system

- By default any new server will requires updates; you will need to complete this before making any other change.
- `sudo apt update`
- `sudo apt list --upgradable`
- `sudo apt upgrade -y`
- `sudo reboot`
- Note: sometimes a second round of updates may be needed.

Keeping the system up-to-date

- When using your server for production on a permanent basis, make sure to schedule maintenance windows to perform system updates and changes on regular intervals.
- A common practice is to have maintenance operations scheduled once a month.

Installing your web server

- For this workshop we will be using Apache HTTPD (commonly known as ‘Apache’). It is one of the most popular open-source web server available.
- Apache HTTPD provides a lot of features, with a modular configuration. Plenty of documentation and resources are available to use it.
- `sudo apt install apache2 -y`
- `sudo systemctl status apache2`

Apache HTTPD on Ubuntu

- Ubuntu Linux being based on Debian it inherit of the same approach to configure Apache HTTPD.
- The configuration is split in various files, under the `/etc/apache2` directory. Specific commands are also available (not part of Apache HTTPD proper).
- You will need to adjust the instructions provided here if you are working on a Linux distribution that is not Debian-based.

Creating a website

- You need to create a basic configuration to host your website; we will also need some placeholder content at the beginning.
- To streamline this process we will be using a script to generate most of this automatically.
- `sudo mkwebsite-http.sh server.ilm-it-security.org`

Activating a website

- Your web site configuration must be enabled, and the default one disabled.
- `sudo a2ensite server-http`
- `sudo a2dissite 000-default`
- `sudo systemctl reload apache2`

Validating your website

- Now that your website is online, you can run various tools to confirm how it is working.
- We will be using a combination of local tools and online services.
- You can also check the logs for the web server and the website.

Validating with local tools

- `nmap <IPv4 address>`
- `nmap -6 <IPv6 address>`
- `nmap server.ilm-it-security.org`
- `wget server.ilm-it-security.org`
- `curl -I server.ilm-it-security.org`
- Your web browser of choice

Validating with online tools

- https://www.hardenize.com/
- https://internet.nl/
- https://securityheaders.com/
- https://radar.cloudflare.com/scan
- https://web-check.xyz/

Checking the logs

- Various log files are available under `/var/log/apache2`:
- `access.log`
- `error.log`
- `server.ilm-it-security.org-http-access.log`
- `server.ilm-it-security.org-http-error.log`
- Use the ‘cat’ or ‘tail’ commands to review the files. Do not use a regular text editor.

Securing the web server

- The very first step to improve the security of your web site is to enable HTTPS, to encrypt network traffic.
- This require to obtain a digital security certificate and to reconfigure Apache HTTPD.

Obtaining a security certificate

- We will be using the Let's Encrypt free service to obtain a security certificate.
- A dedicated tool, named Dehydrated, will allow us to automate the request and renewal for that certificate.
- `sudo apt install dehydrated -y`
- `sudo apt install dehydrated-apache2 -y`

Configuring Dehydrated

- Edit `/etc/dehydrated/domains.txt`
- Add one line with the following content:
- `www.server.ilm-it-security.org server.ilm-it-security.org`
- `sudo /usr/bin/dehydrated --register --accept-terms`
- `sudo LE-certs-renew.sh`
- `sudo mkdir /etc/apache2/sites-certificates`
- `sudo chmod 750 /etc/apache2/sites-certificates`
- `sudo LE-certs-copy.sh`
- Do not restart Apache HTTPD yet!

Enabling HTTPS

- Apache HTTPD will require to have a module enabled to work with HTTPS.
- A new configuration is required for our website. Once again, we will be using a script to generate the required file.

Configuring Apache HTTPD

- `sudo a2enmod ssl`
- `sudo a2enmod rewrite`
- `sudo mkwebsite-https.sh server.ilm-it-security.org`
- `sudo a2ensite server-https`
- `sudo systemctl restart apache2`

Validating your website (again)

- Since we have now adjusted our configuration, we will be running the same tools than before to confirm that all changes have been deployed successfully.
- `nmap -p T:443 server.ilm-it-security.org --script=ssl-cert`
- `nmap -p T:443 server.ilm-it-security.org --script=ssl-enum-ciphers`

More TLS security

- Note: this section is optional for the workshop
- In itself, using TLS and a security certificate is not sufficient. Some fine tuning is needed.
- You can use the Mozilla SSL Configuration Generator to get more insight.
- `https://ssl-config.mozilla.org/`
- Server version: `apache2 -v`
- OpenSSL version: `openssl version`

More security configuration

Additional configuration can be added to Apache HTTPD to make it more secure and provide additional features:

- Automatic redirection from HTTP to HTTPS
- Enabling HTTP/2
- Define security headers

HTTP to HTTPS Redirection

- mod_alias must be enabled
- `sudo a2enmod alias`
- The “http” configuration file for your website already contains the necessary item; it just need to be activated.
- `Redirect permanent / https://www.server.ilm-it-security.org/`
- `sudo systemctl reload apache2`

Enabling HTTP/2

- Once again, we need a module for this.
- `sudo a2enmod http2`
- `sudo systemctl restart apache2`

Updating existing HTTP headers

- In the `/etc/apache2/conf-available/security.conf` file:
- Update *ServerTokens* from *OS* to *Prod*
- Update *ServerSignature* from *On* to *Off*
- `sudo systemctl reload apache2`

Adding new HTTP headers

- Yes, we will need another module.
- `sudo a2enmod headers`
- Set the following headers in the `/etc/apache2/conf-available/security.conf` file:
- `Header set X-Frame-Options: "DENY"`
- `Header set Referrer-Policy: "same-origin"`
- `Header set X-Content-Type-Options: "nosniff"`
- `Header set Content-Security-Policy "default-src 'self' 'unsafe-inline'"`
- `sudo systemctl restart apache2`

Validating your website (still)

- Because changes have been made, we need to validate that our website is working properly with the new and improved configuration.
- `curl -I --http2 https://server.ilm-it-security.org`

Apache HTTPD log files

- Note: this section is optional for the workshop
- By default, log files will be kept for 14 days, rotated every day, and compressed after the first one.
- If you like to change that behavior
  you will need to adjust the `/etc/logrotate.d/apache2` file. And restart the logrotate service.
- `sudo systemctl restart logrotate`

Enabling delegated access

- You may be in a situation where you would need to provide access to additional people to manage the website, without granting full access to the server.
- This will require to have specific user accounts, group and configure policies and permissions.

Adding a group

- Create a group, then include all users managing the web site to that group, plus the Apache user itself.
- `sudo groupadd webmasters`
- `sudo usermod -a -G webmasters user`
- If connected, the user will need to reconnect to gain the new group membership.

Applying new permissions

- Now that we have a dedicated group, we can use that group for setting permissions on all the web content.
- `sudo chgrp -R webmasters /var/www/server`
- `sudo chmod g+s /var/www/server`
- This could also potentially be used for granting access to configuration files and logs.

Securing SSH access

- Since SSH is used to manage your server this would also require some attention to improve the default configuration.
- This will additionally allow more automation and ease of use.

SSH hardening

- Edit the `/etc/ssh/sshd_config` file
- `PermitRootLogin no`
- `PermitEmptyPasswords no`
- `PubKeyAuthentication yes`
- `AllowGroups <group>`
- `sudo systemctl restart ssh`

Warning

- The next steps depends on the application that you are using on your personal laptop, especially your SSH client. The required configuration and tools may differ greatly.
- You may lose access to your server if using an incorrect configuration.

Using SSH keys

- Initially using SSH keys requires a little bit of setup and practice and will make things more easier and secure on the long run.
- In a nutshell:
  - Create a SSH key and push it to the server
  - Confirm that your access is working
  - Disable password-based access

Managing your SSH key

- Generate the key: `ssh-keygen`
- Copy the key to the server: `ssh-copy-id` or `scp`
- Load the key in a SSH agent: `ssh-agent` and `ssh-add` (or use the custom `start-ssh-agent.sh` script)
- On MS Windows, you may have a better user experience using PuTTY.

Using a firewall

- On top of the network filtering options that may be available from the cloud hosting service you can use some firewall rules on the system itself.
- This provides an additional layer of security, with more granular rules and better control.

Our firewall policy

- Allow incoming HTTP (TCP/80) and HTTPS (TCP/443) from anywhere on the Internet.
- Allow incoming SSH (TCP/22) only from the management server.
- Allow outgoing basic services (NTP – UDP/123, DNS UDP53 & TCP/53). HTTP and HTTPS will not be enabled by default, only on demand.

Setting your own firewall

- Ubuntu provide ufw (Uncomplicated Firewall) by default for network filtering. It is based on nftables.
- To benefit from some advanced functions we will be disabling ufw and using directly nftables.
- `sudo systemctl stop ufw`
- `sudo systemctl disable ufw`
- `sudo systemctl enable nftables`

Warning

- You may lose access to your server if using an incorrect configuration.
- You can schedule a task to automatically reset your firewall when testing and fine-tuning your access rules.

Automatic firewall reset

- Use a cronjob to reset your firewall rules at a set time before applying the new ones
- `sudo crontab -e`
- `<mm> <hh> * * *`
  `/usr/local/sbin/reset-firewall.sh`
- Don’t forget to disable that rule once things are working properly!

Configuring nftables

- We will be using an already prepared configuration file to build our network filtering rules.
- Copy the configuration file to `/etc/nftables.conf`
- `sudo systemctl restart nftables`

That’s the end!

- Thanks again for attending this workshop today!
- Please share your questions, comments, feedback.
- What could have been done better? What did you like the most? Any other suggestions?