SERVER HARDENING

Presented by: Daniel Waymel and Corrin Thompson

at TexSAW 2014 at the University of Texas at Dallas

OUTLINE

- Intro
- Securing Your Access
- Restricting Unwanted Access
- Monitoring and Alerts
- Recap

- Note for viewing these slides before the workshop: These slides contain an outline of topics that will be covered, but they do not encompass the entire workshop material. Much of this will be covered in live demonstrations and hands-on exercises for participants to follow along with.

THE PRESENTERS

- Daniel Waymel – slides & demo
  - Master’s degree in Computer Science expected May 2015
  - President of the CSG (Computer Security Group) at UT Dallas
- Corrin Thompson – individual questions & participation assistance
  - Bachelor’s degree in Software Engineering expected May 2017
  - Secretary of the CSG at UT Dallas

SCOPE

- Target System:
  - An always-on, always-connected Linux system hosted off-site. (ex. Linux VPS (Virtual Private Server))
  - System is accessed by a single user for miscellaneous tasks.

OBJECTIVES

- Understand how to:
  - Configure secure remote access
  - Defend against basic network-based attacks
  - Configure remote alerts and monitoring of a system
  - Apply the concepts applied here to different systems and scenarios

SECURING YOUR ACCESS

OBJECTIVES

- Be able to securely access the remote system

- We will look at:
  - SSH (single user)
- Other related topics:
  - SSH (multiple users)
  - SFTP (not FTP, why?)
  - HTTP

USER ACCOUNTS PT 1

CREATE NEW USER

- Create a non-root user (with sudo rights)
  - `# adduser <username>`
    - Create password, confirm user info
  - `# adduser <username> sudo`

USER ACCOUNTS PT 2

DISABLE REMOTE ROOT LOGIN

- Restrict access via ssh
  - `# nano /etc/ssh/sshd_config`
    - Note: You can use other text editors, but it must be done as root or with sudo.
  - Additional lines:
    - `PermitRootLogin no`
      - Disallows ssh access for root account

BACKGROUND – RSA KEY PAIRS
FOR SECURING SSH

- A public/private key pair. Each private key has a corresponding public key generate with it.
- Given one, it is (presumably) infeasible to determine the other one.
- Either can be used for encryption. If data is encrypted with the public key, only the private key can decrypt it; likewise the reverse is true.

SSH ACCESS – RSA KEY FILES
FOR SECURING SSH

- The public RSA key is stored on the server to be accessed, the private is used to authenticate.
- The same pair can be used for multiple machines – store the same public key on different machines, and the private key can be used to access all of them.

SSH ACCESS – RSA KEY FILES

GENERATE SSH KEYS AND ADD TO SYSTEM

- On remote system
  - `$ mkdir ~/.ssh`
  - `$ chmod 700 ~/.ssh`
- On local system
  - `$ ssh-keygen  -t rsa`
  - Or
  - `$ ssh-keygen –t rsa –b 4096`
  - `$ ssh-copy-id <user>@<remote_system> [-i public_key_file]`

SSH ACCESS – RSA KEY FILES

REQUIRE KEY FILES FOR REMOTE ACCESS

- Enforce key files for access via ssh instead of passwords.
- On remote system
  - In the file /etc/ssh/sshd_config
    - `PasswordAuthentication yes`
      - Change to no
    - `UsePAM yes`
      - Change to no
    - `#AuthorizedKeysFile %h/.ssh/authorized_keys`
      - Uncomment (remove ‘#’)

QUICK NOTE: SSH KEYS

- Examples given are for OpenSSH style key pairs on a Linux/Unix system. If you are accessing a Linux system from a Windows system, other options are available;
  - Putty: Comes with PuttyGen, which can generate RSA key pairs in the format the Putty uses. It can also be used to convert key files generated by ssh-keygen (OpenSSH) to those usable by Putty. (Putty is freely available.)
  - Xshell (by NETSARANG): Xshell can generate keys usable by OpenSSH and deploy them to servers. It can also import existing key files in the OpenSSH format. (Xshell is a commercial paid product, but can be downloaded free for home and student use.)

RESTRICTING UNWANTED ACCESS

OBJECTIVES

- Be reasonably confident that unauthorized access attempts will not work (at least, not without effort)

- We will look at:
  - Lockout of IP addresses following failed access attempts
  - Basic firewall configuration (iptables)
  - Security by obscurity
- Other related topics:
  - Network-based IDS/IPS (Intrusion Detection/Prevention Systems)

BLOCK BAD SSH ATTEMPTS

SETUP FAIL2BAN

- fail2ban allows easy lockouts following failed connection attempts. It can be used to ban an IP address temporarily for a chosen amount of time after a configurable amount of failed attempts.
- `$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`
- `$ sudo service fail2ban restart`
- Can edit jail.local to make changes:
  - Enable for more services besides ssh
  - Change ban time
  - Change allowed attempts
  - Change whitelisted IPs (ignore bad attempts from loclalhost for example [default])
  - Send alerts by email (or SMS via email-to-SMS gateway)
    - More on this later.
- [Demo]

CLOSE UNNECESSARY PORTS

- iptables - a configurable firewall.
- fail2ban, and many other programs, interact with iptables to achieve some of their functionality.
- NOTE: iptables operates on IPv4 only, ip6tables will operate on IPv6 only. The scope here is iptables and IPv4 only. The makers of iptables are developing nftables as an upgraded replacement which will handle all of what the various existing programs in the *tables suite do.

CLOSE UNNECESSARY PORTS

- Two main approaches:
  - Specify what to allow (whitelist)
  - Specify what to not allow (blacklist)
- In this scenario, a whitelist is easy to implement and the most effective.

CLOSE UNNECESSARY PORTS

OVERVIEW OF IPTABLES

- [Sample iptables rules]
- Rules are evaluated top-down. The first rule that fits gets applied. Once a packet is ACCEPTED, REJECTED, or DROPPED, no further rules are evaluated for that packet
    - What this means: The ordering of your rules is very important! Sometimes multiple rules could apply, but only the first one will.

CLOSE UNNECESSARY PORTS

CONFIGURE BASIC IPTABLES RULES

- Some sample rule commands [demo]
- Allow traffic on localhost
  - `$ sudo iptables -I INPUT 1 -i lo -j ACCEPT`
- Allow established connections to persist:
  - `$ sudo iptables -I INPUT 2 -m -ctstate ESTABLISHED,RELATED -j ACCEPT`
- Default INPUT chain to DROP:
  - `$ sudo iptables -P INPUT DROP`

ADDING ANOTHER LAYER
BACKGROUND FOR PORT KNOCKING

- Three possible ways to authenticate:
  - Something you know
  - Something you have
  - Something you are
- Single-Factor Authentication: choose one type of the above
- Two-Factor Authentication: choose two types

ADDING ANOTHER LAYER
SAMPLE PORT KNOCKING USE

- Port Knocking is similar to two-factor authentication in its rationale.
- An example case:
  - SSH is the only service running on a system
  - All ports are closed
  - Requesting a connection (which will be refused as the ports are closed) on a pre-determined sequence of ports in a specific order within a specific time period will open the port for SSH.
  - The port closes automatically after the allowed window has passed.
  - NOTE: Authentication for SSH connections is not specified, but should still be at the appropriate level of security.

ADDING ANOTHER LAYER
SETTING UP PORT KNOCKING

- For this scenario:
  - Must “knock” on three ports in sequence within a 10 second period.
  - Standard SSH port 22 will open for 10 seconds receiving connection attempts before closing automatically.

ADDING ANOTHER LAYER
SETTING UP PORT KNOCKING

- `$ sudo apt-get install knockd`
- This will install both the knockd daemon, as well as the knock utility.
- Default configuration is to have one knock sequence to open a port, and another sequence to close the port.
  - Problem: What if you forget to close it?
- Default for adding iptables rule to open a port is flawed in some cases.
- [Demo]

MONITORING AND ALERTS

OBJECTIVES

- Increase awareness of important events on the remote system
- We will look at:
  - Automated email/SMS alerts
  - System Logs
- Other related topics:
  - Anti-virus
  - Host-based IDS

SENDING EMAIL NOTIFICATIONS
SSMTP CONFIGURATION

- Monitoring for events and logging is good, but only if those logs and events are known.
  - Failed access attempts (SSH in our case)
  - Unexpected system changes (flagged by IDS, such as tripwire)
  - Benign events:
    - Task has completed
    - Message received (ex. IRC)
    - …

SENDING EMAIL NOTIFICATIONS
SSMTP CONFIGURATION

- ssmtp can be used to easily send email notifications.
- For this scenario:
    - Create a gmail account to use for sending
    - Configure ssmtp on the system to use that account
    - Create a script to streamline sending notifications
- [Demo]

SYSTEM LOGS
TRACKING EVENTS

- Some logs related to topics covered:
  - /var/log/auth.log
  - /var/log/fail2ban.log
  - /var/log/mail.log
  - ~/portknock.log
- Logs such as those listed above can be a useful tool to determine what has happened on a given system. They act as a timeline of recent events, and can provide information useful in determining if something bad has happened (such as someone gaining unauthorized access, or an attempt at access).
- [Demo]

RECAP

RECAP

- Covered the following for a Linux system:
  - Securing (individual) remote access
    - Login using on-root account using keyfiles only; no remote root access permited
  - Restricting unwanted access
    - Configure lockouts after bad access attempts, basic firewall rules, and a means of adding more layers of defense
  - Setting up notification of system events
    - Setup email/SMS alerts; discussed means of system changes triggering alerts
  - Brief look at system logs
- What about a Windows system? Portable systems (tablets, notebooks, etc)?
  - (Discussion)

CONTACT INFO

- Daniel Waymel
  - daniel.waymel@utdallas.edu
- Corrin Thompson
  - cnt130030@utdallas.edu