Simple but Effective Server Hardening

Kyle Rankin
VP of Engineering Operations
Final, Inc.

https://greenfly.org/talks/security/simple_hardening.html

Agenda

- Introduction
- Classic Hardening
- Security Best Practices
- What to Avoid
- SSH Server
- SSH Client
- SSH 2FA
- Root and Sudo
- Reuse Puppet Certs
- Simple Cloud Hardening
- General Tips
- Questions?

Introduction

- Security hardening more important than ever
- Was hardening infrastructure for a PCI audit
- Had to refer to an approved hardening guide
- Amazed at all the outdated and ineffective info
- A few simple steps can greatly increase security
- Certainly can harden further.

Classic Hardening

- Many hardening guides written for Red Hat circa 2005
- Not necessarily *bad* advice, just deprecated/already done
- Turn off telnet
- Inetd hardening
- Disable all unnecessary services
- Tcpwrappers
- shadow passwords
- Disable shells on common role accounts.

Security Best Practices

- Security best practices often == overall best practices
- Principle of Least Privilege
- Keep it Simple, Sysadmin
- Apply patches
- Layers of defense
- Good logging/audit trails
- Encrypt.

What to Avoid

- Obscurity (changing default ports)
- Attacker-generated firewall rules (fail2ban, etc.)
- Port knocking
- Reliance on any single security measure
- Network software that doesn't support encryption
- Complexity.

SSH Server

- A few basic changes to /etc/ssh/sshd_config
- Disable Root Login:

  ```
  PermitRootLogin no
  ```

- Only use Protocol 2:

  ```
  Protocol 2
  ```

- Disable Password Authentication:

  ```
  PasswordAuthentication no
  ```

- Limit Crypto Options:

  ```
  Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,
    aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr

  KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256

  MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,
    hmac-ripemd160-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,
    hmac-sha2-256,hmac-ripemd160,umac-128@openssh.com
  ```

SSH Client

- Generate strong keys:

  ```
  ssh-keygen -t rsa -b 4096
  ssh-keygen -t ed25519
  ```

- Use password-protected SSH keys
- Avoid copying private keys around
- Use ssh-add to cache password for limited time
- My lunch reminder:

  ```
  ssh-add -t 3h
  ```

- Pay attention to host key warnings.

SSH 2FA

- Requires an additional factor before login
- Some use TOTP, others SMS/Phone, or both
- A number of approaches, providers
- Many configured with PAM, others SSH client restrictions
- I like Duo's approach, but not free
- Google has wide support, free.

SSH 2FA Continued

- Install Google Authenticator from distro package (libpam-google-authenticator) or from source
- Enroll each user account:

  ```
  $ google-authenticator
  ```

- Scan QR code or add secret to Google Auth app
- Add to top of /etc/pam.d/sshd:

  ```
  auth required pam_google_authenticator.so
  ```

- On Debian-based systems comment out:

  ```
  @include common-auth
  ```

- Change /etc/ssh/sshd_config:

  ```
  ChallengeResponseAuthentication yes
  AuthenticationMethods publickey,keyboard-interactive
  ```

- Restart ssh service
- Login:

  ```
  $ ssh kyle@server1.example.com
  Authenticated with partial success.
  Verification code:
  ```

Root and Sudo

- Disable root/group accounts and use sudo:
  - Avoids shared passwords
  - Makes revoking access simpler
  - Provides audit trail
- Sudo best practices:
  - Restrict NOPASSWORD sudo to daemon role accounts
  - Try to avoid granting ALL access to users
  - Wrap risky commands inside custom scripts.

Reuse Puppet Certs

- If you use Puppet Masters, you have internal trusted CA
- Makes internal mutual TLS auth much simpler
- Each host has cert, key, CA cert locally:

```
CERT: /var/lib/puppet/ssl/certs/${cert_name}.pem
KEY:  /var/lib/puppet/ssl/private_keys/${cert_name}.pem
CA:   /var/lib/puppet/ssl/certs/ca.pem
CRL:  /var/lib/puppet/ssl/crl.pem
```

- To use in NGINX:

```
  ssl_certificate /var/lib/puppet/ssl/certs/${cert_name}.pem;
  ssl_certificate_key /var/lib/puppet/ssl/private_keys/${cert_name}.pem;
  ssl_client_certificate /var/lib/puppet/ssl/certs/ca.pem;
  ssl_crl /var/lib/puppet/ssl/crl.pem;
```

- Can add Subject Alt Names to Puppet certs with dns_alt_names option.

Simple Cloud Hardening

- Delete/disable default admin account
- Don't store secrets in userdata script
- Try to generate secrets on host when possible
- Limit access even within security groups
- Encrypt internal communication
- Store sensitive data on non-root, encrypted disks.

General Tips

- Use config management with configs checked into source control
- Encrypt any secrets checked into source control!
- Use orchestration software
- Use /dev/shm to store sensitive files
- Consider logging all new network connections
- Set up remote logging
- SSH into internal servers via bastion host
- Restrict access to networks via VPN
- Enable TLS between web services.

Questions?

Additional Resources

- This talk: https://greenfly.org/talks/security/simple_hardening.html
- Secure Secure Shell
- Google Authenticator
- @kylerankin
- kyle@getfinal.com