Information Security Engineering Team

Goal:

- Provide an automation configuration setup for a webserver including --> ApacheWebServer + MySql Database

Deliverable:

- Documentation of the audit, process, decisions made, arguments and reasons
  - Linux hardening
  - ApacheWebServer hardening
- DevOps deployment recommendations

Host Information:

- Machine Name : VagrantBox1
- Ip Address : 127.0.0.1
- Mac Address : xx:xx:xx:xx:xx:xx
- Audit by Individual : Srinivas Piskala Ganesh Babu
- Date : 05.31.2017
- Asset Number : 1
- Version : 1.0

Commands to work out:

- vagrant up
- Change the ssh port in vagrant to 8082 - uncomment the last line in vagrant to connect back
- kept the vagrant user enabled or not deleted to further debug and correct

Linux/Kernel Hardening

Operating System:

- Select a stable and latest version of linux with vulnerability fixes --> ubuntu/xenial64
- Review the vulnerability list if any, to audit the possible threat analysis using attack trees
- Make a checklist of used services in linux (apache+db) to keep track of zero day.

1.Linux Libraries:

```
* Task1: Linux repository update and default libraries update if any
  * Description: The library source should be update to pull in new
updates/patches/fixes for a library, also making sure new updated libraries are
further installed
  * Command:
    * apt-get -y update
    * apt-get -y upgrade
  * Threat/Vulnerability: possible bugs in old packages or libraries that could
expose the surface for any exploits
```

```
  * Mitigation/Fix: update/ upgrade all the packages

* Task2: Remove un-necessary applications or any legacy dependacies
  * Description: Remove any legacy dependancy that are not used anymore or unwanted existing in the machine - autoremove, to clean the local repository copy of any package use autoclean which clears out only the packages that are largely useless and no longer existing
  * Command:
    * apt-get -y autoremove
    * apt-get -y autoclean
  * Threat/Vulnerability: legacy libaries or additional packages that are not used or maintained actively might contain known bugs that could expose the server internals
  * Mitigation/Fix: remove all the unused or legacy libarires and packages to reduce the attack surface
```

2. Setting up the Web Server:

```
* Task1: Apache Web Server
  * Description: As the repository is already updated, pull the latest stable version of apache to start with the web server and configure the folder that would be exposed

* Task2: Sql Database
  * Description: Similarly download and install the mysql database from the repo.
```

3. Disable root - lock out the root account

```
* Task1: Disable root account or lock out root
* Description:
* Command:
  * passwd -l root or sudo usermod -p '!' root <set unusable password for root>
* Threat/Vulnerability: Privilege escalation possibilities occur during the presence of an admin or higher privilege account
* Mitigation: Possibilities of privilege escalation could be reduced by disabling high privilege accounts
```

4. Disable less secure protocol/services

```
* Task1: Disable less secure/plaintext protocols and services
* Description: Existence of less secure protocols provide a path for a threat - telnet, netcat, tftp, ftp, ..
* Command:
  * apt-get -y purge <service/protocol>
* Threat/Vulnerability: Weaker/easy paths in an Attack tree should be removed, they might make it easy for an attacker to exploit
* Mitigation: Remove the existence of such protocols so there is not a possibility of their existence or being enabled
```

5. SSH Hardening

```
* Task1: SSH Custom Port Run
* Description: Running SSH on a custom port would hide and protect it from a
scanning phase of hacking hence reduce the detection of ssh service
* Command:
  * Port <num> in the sshd_config
* Threat/Vulnerability: Running a service in known ports/ top ports will make it
easier to be detected and scanned, SSH being a control feature for the server it
should be properly hidden
* Mitigation: Less exposure to scans by setting ports > 1024 reserved category will
hide the ports from initial scans, reduce attack paths and increase
detection/response time

* Task2: SSH Root Login Password Disable
* Description: Disabling password login for root accounts and using public key
authentication would increase the security by enabling certificatte or key pinning
at the server
* Command:
  * PermitRootLogin prohibit-password
* Threat/Vulnerability: password logins are prone to brute force attacks/ privilege
escalation attacks
* Mitigation: disable password logins and replace them with a key based
authentication

* Task3: Account Lockout
* Description: Set max-retires and max-sessions to protect from brute force attack
logins and brute force attempts detection
* Command:
  * maxretires + maxsessions
* Threat/Vulnerability: Brute forcing attempts on any authentication is a easy
attempt that an attacker take to enumerate possible keys/passwords
* Mitigation: Setting max retries and sessions would keep track and help in earlier
detection and mitigation

* Task4: SSH Tuning
* Description: Removing unwanted ssh features like tunneling, forwarding would
prevent it from zero days and easy exploitation
* Command:
  * features like forwaring, tunneling should be disabled
* Threat/Vulnerability: unwanted features would make a protocol look complex,
complex construction is easier to break
* Mitigation: Reduce the attack surface of any protocol or feature by disabling
unwanted or unused feature

* Task5: Strengten the Crypto Used
* Description: Use strong ciphers and modes during encryption or secured operations
* Command:
  * Enable only ssh cipher that are strong + hmacs with strong hash
* Threat/Vulnerability: Enabling multiple modes of encryption would cause the usage
of a weak mode/crypto and exploit its weakness leading to data leakage
* Mitigation: Configure only strong crypto/modes of operation to work with which
would be free from zero days or known vulnerabilities

* Task6: Regenerate ssh key chain + disable vagrant default keys
* Description: Disable or alter the default keys used by vagrant + set the key
chain properly to eliminate default credentials
* Command:
  * sshd/authorized_keys
```

```
* Threat/Vulnerability: default keys presence in the directory would allow default
authentication possible with keys available from shodan or github or google
database
* Mitigation: Use custom generated strong keys with greater key lengths
```

6. Programming Language/Scripting Compilers/Interpreters

```
* Task1: Disable all the programming interfaces that would allow an attacker to run
any malicious script
* Description: Except the prog interface the underlying webserver is using, disable
everything else
* Command:
  * chmod 000 or apt-get purge
* Threat/Vulnerability: presence of many programming interfaces (os by default has)
would increase the possibility of exploitation of known vulnerabilty like privilege
escalation as well as increasing attack tree paths
* Mitigation: Disable or remove unused compilers/interpresters
```

7. Firewall

```
* Task1: The default firewall configuration should block all the ports and deny
access except the services or ports we want to expose
* Description: Except the web server - 80, 443 and custom ssh port all the other
ports should be disabled
* Command:
  * ufw default deny, ufw allow ssh, http, https
* Threat/Vulnerability: A firewall would ensure blocking other ports, allow only
required ports, prevent ports from scanning, brute force attacks as well as some
intelligence to shield dos attacks
* Mitigation: Configure firewall rules to allow/deny ports
```

8. Kernel Tuning

```
* Task1: Address space randomization enable
* Description: Enable the address space randomization - aslr to make it harder for
buffer overflow attacks
* Command:
  * kernel.randomize_va_space=1
* Threat/Vulnerability: ASLR disable will allow easy buffer overflow exploits
leading to privilege escalation or reading extra buffer space.
* Mitigation: Randomizing address space would make it hard for buffer overflow to
occur

* Task2: Kernel Networking Shield
* Description: Certain redirects, routing and other error cases should be handled
at the socket present at the CPU
* Command:
  * icmp redirects, rp_filter, exec shield, spoofed packet logs
* Threat/Vulnerability: Sockets are created at the CPU, packets slip to the CPU
causing it error or crash or exploit based on the routing redirects or errors in
packets
* Mitigation: Such bogus packets should be discarded without cpu processing
```

10. User Accounts

```
* Task1: Remove Default Users
* Description: Removing default users and passwords from the machine
* Command:
  * deluser
* Threat/Vulnerability: Default user existence would increase the chances of brute
force logins
* Mitigation: Clearing out default username and password would ensure known
password attack protection

* Task2: Password Configurations - password reuse or expiry
* Description: Configure password reuse and expiry policy
* Command:
  * remember=15,
* Threat/Vulnerability: enabling password reuse and no expiry will cause the
password to never expire and attack prolong
* Mitigation: Prope password config helps promoting password compromise recovery

* Task3: New sudo user and delete all other accountse
* Description: Create new sudo user and delete all the other pre existing accoounts
* Command:
  * deluser, useradd,
* Threat/Vulnerability: Presence of known usernames == brute force attacks
* Mitigation: random unique username/password ensures thorough configuration
```

11. Enable SELinux or AppArmour

```
* Task1: Enable seLinux or appArmour and set necessary profiles to be enforcing
* Description: Permissions and Execution could be restricted by selinux or
appArmour applying roles and rules
* Command:
  * selinux enforcing - set enfroce
  * apparmour start
* Threat/Vulnerability: Security misconfig or loop holes might slip in some app
execution for an unprivileged user
* Mitigation: Enforcing rules from linux security core would protect and fill in
loop holes
```

15. Web Server Hardening

```
* Task1: Supress Apache Server details exposure
* Description: Apache server details like version subversion are exposed in the
http packets or in the web server pages
* Command:
  * server token, server signature, trace enable
* Threat/Vulnerability: Version and details exposure would be an cheap and easy
attack path for finding out the known vulnerability, exploits or pocs to use
against the server
* Mitigation: Disabling version, subversion would supress the information or
possibility of social engineering for any exploit/zero day search
```

* Task2: Directory Traversal Attacks + Content Attacks
* Description: Existence of directory traversals or content exposure attacks are made possible by default settings of apache
* Command:
  * -Indexes, -Includes, -ExecCGI, -FollowSymLinks
* Threat/Vulnerability: Configure apache such that the direcory traversl + content attack surface which can be used to further explore and exploit the web application
* Mitigation: Eliminate such provisions to prevent and reduce attack surface

* Task3: File Details or File Social Engineering Shield
* Description: File internals and details about the inode are exposed throught the etag of the file
* Command:
  * FileEtag None
* Threat/Vulnerability: File internal information or any possible internal data exposure would be a valuable data for attack tree traversal
* Mitigation: Enforce rules to hide such data

* Task4: Dos Attack Protection
* Description: DosAttacks compromising the availability take down the server with flooding, tuning such params could make the server robust
* Command:
  * timeout 60, LimitRequestBody, keepAliveTimeout, MaxClients, LimitRequestFields, LimitRequestFieldSize
* Threat/Vulnerability: Requests, RequestSizes, Clients and Timeouts are parameters that could be exploited with flooding taking down the server
* Mitigation: Tuning of such params would ensure more control and less flooding issues

* Task5: Application Security Protection
* Description: Xss, ClickJacking, csrf protection with httponly cookie or readonly, xss protection enforcing could be set to prevent application security attacks
* Command:
  * HttpOnly Cookie, Secure Cookie, XSS Protection, Xframe options
* Threat/Vulnerability: Without proper app sec protections any web application running on the web server could be exploited and its customer being compromised
* Mitigation: Setting rules like cookie readonly and measures from OWASP page would ensure good appsec practices

* Task6: System specific directory Protection
* Description: Directory with system files could be protected with proper rules
* Command:
  * <dir> -Indexed -AllowOverride None</dirr>
* Threat/Vulnerability: Access to system files directory would expose huge data and possibilities
* Mitigation: Separate directories with system files could be provided additional layer of security

* Task7: HTTP Version Degradation Protection
* Description: degrading http version to perform attacks could be protected using rewrite function in headers
* Command:
  * Rewrite Rules
* Threat/Vulnerability: Versin degradation attacks are made possible by existing exploit PoCs about legacy version
* Mitigation: Proper plugin and rules should be setup to not allow such scenario

```
* Task8: SSL Settings
* Description: Enable SSL and add keys for secure web communication that would be
required for a logged in user
* Command:
  * a2enmod ssl + ssl key and cert addition
  * Key for now is without a passphrase to overcome vagrant configuration of
password query
* Threat/Vulnerability: Improper ssl configuration would lead to possible crash or
breaking secure communication
* Mitigation: SSL keys with strong key pair and properly configured web server
would ensure secure communication

* Task9: SSL Hardening
* Description: Harden SSL by limiting certain ciphers, enabling perfect forward
secrecy, high security mode and disabling sslv2 sslv3 would make ssl robust
* Command:
  * SSL Protocol, SSLCipherSuite
* Threat/Vulnerability: Weak Ciphers and Modes are prone to known attack and
takedowns
* Mitigation: Enforcing high security mode would ensure only the strong ciphers are
used

* Task10: Firewall or Reverse Proxy for Web Server
* Description: Nginx or ModSecurity could be configured to add intelligence server
or layer between client and server
* Command:
  * modsecurity, secengine on
* Threat/Vulnerability: Exposing server directly without any layer would increase
the risk of compromising the server
* Mitigation: An intelligence layer to limit add rules to the packets would perform
necessary protection

* Task11: Web Server Files Protection - Setting separate user for web server
* Description: The web server should be set with a separate user to work with
* Command:
  * User:, Group:
* Threat/Vulnerability: Setting a more privilege user of the web server files would
allow more possibilities and increase attack surface
* Mitigation:  A less privilege user to be the owner of the web server files would
ensure no execution and proper priv access
```

Future Upgrades or Pending Security Measures to work on:

12. Drive:

```
* Task1: Makeing Temperory storage directories nonexecutable
  * Description: temperory storage directory is a place where a normal non sudo
user might have permission to execute any file or binary
  * Command:
  * Threat/Vulnerability:
  * Mitigation/Fix:
```

13. BIOS:

- Task1: Protect BIOS with a password
  - Description: BIOS security should be protected from override
  - Command:
  - Threat/Vulnerability:
  - Mitigation/Fix:
- Disable booting from external devices (CD/DVD/USB)
  - Description: BIOS security should be protected from override
  - Command:
  - Threat/Vulnerability:
  - Mitigation/Fix:

14. Directory Setting:

- Task1: Lock down the boot directory to read-only

15. Media:

- Task1: Disable USB usage

#

DevOps Team Recommendations

Deployment Fixes:

- Replace the keys for SSH in the Folder with a stronger key generated
- Replace the keys for SSL in the Folder
- Proper unlinking of the shared folder of vagrant
- Proper Hiding of Private Keys

#