Kaya Nelson

CST – 221

Micheal Landreth

06/18/2024

GitHub Link: [Kdeshun/CST-221-WEEK-7 (github.com)](https://github.com/Kdeshun/CST-221-WEEK-7)

Basic System Security –

a. How would User Administration and Access Controls help solve the problem of insider threats?

User administration and access controls are crucial in addressing the problem of insider threats. Some key ways they can help include:

1. Least Privilege: Ensuring users are only granted the minimum necessary permissions to perform their job functions. This limits the damage an inside actor can do if they become malicious.

2. Segregation of Duties: Separating critical functions across multiple users to prevent a single person from abusing their access.

3. Auditing and Logging: Maintaining detailed logs of user actions that can be reviewed for suspicious activity.

4. Multi-factor Authentication: Requiring additional verification steps beyond just a username and password, making it harder for attackers to gain access.

5. Automated Provisioning/Deprovisioning: Quickly revoking access when an employee leaves the organization to prevent them from misusing their credentials.

6. Regular Access Reviews: Periodically reviewing user accounts and permissions to identify and remove unnecessary or excessive access rights.

Implementing robust user administration practices and access controls is a key defense against the insider threat problem highlighted in the article.

b. What are some techniques and approaches to prevent buffer overflows?

Some key techniques and approaches to prevent buffer overflows include:

1. Bounds Checking: Validating that memory access is within the expected bounds of the buffer to avoid overflows.

2. Safe String Functions: Using safer string manipulation functions like strncpy() instead of strcpy() to limit copying to the buffer size.

3. Canaries: Inserting "canary" values on the stack that can detect if a buffer is overflowed and trigger a crash.

4. W^X: Enforcing that memory is either writable or executable, but never both, to prevent injected code from running.

5. Address Space Layout Randomization (ASLR): Randomizing the locations of code, stack, and heap in memory to make it harder to predict where to inject code.

6. Compiler Hardening: Leveraging compiler features like stack protectors and control flow integrity checks to detect and prevent buffer overflows.

7. Runtime Instrumentation: Using tools like AddressSanitizer to dynamically detect buffer overflows and other memory errors at runtime.

Applying a combination of these techniques at the coding, compiler, and runtime levels is essential for comprehensive protection against buffer overflow vulnerabilities.

c. How to add a new user, delete a user, change the password of a user, and control who has administrative privileges in Ubuntu Desktop

The Ubuntu Desktop Guide covers user administration tasks in the "User & System Settings" > "User Accounts" section:

Adding a new user:

- Click the "Unlock" button and enter your admin password

- Click the "+" button and fill in the new user's details

- Choose whether the new user should be an administrator or a standard user

Deleting a user:

- Click the user account you want to delete

- Click the "-" button to remove the user

Changing a user's password:

- Click the user account

- Click "Change Password" and enter the new password

Controlling administrative privileges:

- By default, the first user account created is an administrator

- To make a standard user an administrator, click their account and toggle "Allow this user to administer this computer"

- To revoke administrative privileges, toggle the setting back to "Standard"

The guide also covers other user management features like setting profile pictures, automatic login, and other account settings. Mastering these user administration capabilities is crucial for securely managing Ubuntu Desktop systems.

Applying Basic Security –

Here's the bash script to detect users who have logged in during non-office hours (8:00AM to 5:00PM):

A computer screen shot of a black screen

Description automatically generated

This script first sets the office hours start and end times. It then uses the last command to get the login history, and grep to find all the logins that are not between the office hours. Finally, it displays the results.

Output –



4. a. Areas and services that possibly need to be hardened:

* Network services (SSH, web server, database, etc.)
* User accounts and access controls
* System logging and monitoring
* Software updates and patches
* Firewall settings
* Encryption and data protection

b. Configuration files that possibly need to be hardened:

* /etc/ssh/sshd\_config (SSH server configuration)
* /etc/sudoers (sudo access control)
* /etc/hosts.allow and /etc/hosts.deny (firewall rules)
* /etc/security/limits.conf (user resource limits)
* /etc/passwd and /etc/shadow (user account information)
* /etc/fstab (file system mounts)
* /etc/sysctl.conf (kernel parameters)
* /etc/rsyslog.conf (system logging configuration)

c. Linux commands needed:

* ssh, scp, sftp (secure remote access)
* iptables, firewall-cmd (firewall management)
* useradd, userdel, usermod (user account management)
* chmod, chown, chgrp (file and directory permissions)
* crontab (scheduled tasks)
* lsof, netstat, ss (network activity monitoring)
* openssl (encryption and SSL/TLS management)
* yum, apt-get, dnf (software package management)
* journalctl (system log management)

d. Other possible tools:

* Intrusion detection and prevention systems (IDS/IPS)
* Security information and event management (SIEM) tools
* Vulnerability scanners (e.g., Nessus, OpenVAS)
* Configuration management tools (e.g., Ansible, Puppet, Chef)
* Log analysis and monitoring tools (e.g., Graylog, Splunk)
* File integrity monitoring tools (e.g., Tripwire, OSSEC)

e. Additional training or resources:

* Cybersecurity fundamentals and best practices
* Linux system administration and security
* Network security and firewall management
* Scripting and automation (e.g., Bash, Python)
* Incident response and forensics
* Regulatory compliance (e.g., HIPAA, PCI DSS, GDPR)
* Vendor-specific security training (e.g., Red Hat, Ubuntu, CentOS)
* Online resources (e.g., CIS Benchmarks, OWASP, NIST)
* Certifications (e.g., GIAC, CISSP, OSCP)

The key to effectively hardening a Linux server is to have a comprehensive understanding of the system, the services running, the potential vulnerabilities, and the best practices for securing each component. This requires a combination of technical skills, security knowledge, and ongoing monitoring and maintenance.

Ethical Hacking

i. Some tutorials related to IT and software development:

* Network Scanning and Enumeration
* Vulnerability Identification and Analysis
* Exploitation Techniques
* Web Application Hacking
* Wireless Network Hacking
* Cryptography and Steganography

ii. Types of hacking:

* Black Hat Hacking: Malicious hacking for personal gain or to cause harm.
* White Hat Hacking: Ethical hacking to identify and address security vulnerabilities.
* Grey Hat Hacking: A mix of both black hat and white hat hacking, sometimes without explicit permission.

iii. Hacking tools:

* Network Scanning Tools (e.g., Nmap, Angry IP Scanner)
* Vulnerability Scanners (e.g., Nessus, OpenVAS)
* Exploitation Frameworks (e.g., Metasploit, Kali Linux)
* Web Application Hacking Tools (e.g., OWASP ZAP, Burp Suite)
* Wireless Hacking Tools (e.g., Aircrack-ng, Kismet)
* Password Cracking Tools (e.g., John the Ripper, Hashcat)

iv. Types of attacks:

* Network-based Attacks (e.g., DoS, DDoS, Man-in-the-Middle)
* Web Application Attacks (e.g., SQL Injection, Cross-Site Scripting)
* Wireless Attacks (e.g., WEP/WPA cracking, rogue access points)
* Social Engineering Attacks (e.g., phishing, pretexting)
* Physical Attacks (e.g., dumpster diving, shoulder surfing)

v. Other possible tools:

* Incident Response and Forensics Tools (e.g., Autopsy, Sleuth Kit)
* Secure Communication Tools (e.g., VeraCrypt, GPG)
* Penetration Testing Frameworks (e.g., Cobalt Strike, Maltego)
* Network Traffic Analysis Tools (e.g., Wireshark, tcpdump)
* File and Memory Analysis Tools (e.g., Volatility, Rekall)

vi. Additional training and resources:

* Cybersecurity and Ethical Hacking Certifications (e.g., CEH, OSCP, GIAC)
* Online Courses and Training Platforms (e.g., Udemy, Coursera, edX)
* Security Conferences and Workshops (e.g., Black Hat, DEF CON, DEFCON)
* Security Research and Publications (e.g., security blogs, research papers)
* Hands-on Practice and Bug Bounty Programs
* Networking with the security community (e.g., online forums, local meetups)

The field of ethical hacking and penetration testing requires a comprehensive understanding of security principles, network technologies, programming, and a wide range of tools and techniques. Continuous learning, hands-on practice, and staying up-to-date with the latest trends and methodologies are crucial for success in this domain.

i. What is the Trusted Computing Base (TCB)?  
The Trusted Computing Base is the set of all hardware, firmware, and software components in a computer system that are critical to its security. It is the foundation upon which trust in the system is based. The TCB includes the operating system, system firmware, certain applications, and other core components that must be secure and functioning correctly for the system to be trusted.

ii. Why is the Trusted Computing Base important?  
The Trusted Computing Base is important for several reasons:

1. Security Assurance: The TCB represents the minimal set of components that must be trusted to provide security guarantees. If any part of the TCB is compromised, the entire system's security can be undermined.
2. Isolation and Containment: The TCB helps to isolate and contain the security-critical components of the system, ensuring that vulnerabilities or attacks in other parts of the system do not affect the trusted core.
3. Reduced Attack Surface: By minimizing the size and complexity of the TCB, the attack surface of the system is reduced, making it harder for attackers to find and exploit vulnerabilities.
4. Verification and Validation: The smaller the TCB, the more feasible it is to thoroughly verify and validate its security properties, ensuring that it is functioning as intended.
5. Trusted Computing Initiatives: The concept of the TCB is foundational to various trusted computing initiatives, such as Trusted Platform Modules (TPM) and Secure Enclaves, which aim to provide hardware-based security assurances.

In summary, the Trusted Computing Base is the critical foundation upon which a system's security relies. By identifying and securing the TCB, system designers can establish a high level of trust in the overall system's security and integrity.

Research Questions

a) Constraints:

* The matrix must adhere to the constraint that each combination of two characters (row and column) can only appear once in the matrix.
* This ensures that during decryption, there is no ambiguity about which ciphertext characters correspond to the original plaintext characters.

b) Number of possible keys:

* Since the plaintext is encrypted two characters at a time, the number of possible keys is determined by the number of unique combinations of the 26 letters (A-Z).
* The total number of unique combinations is 26 × 26 = 676.
* This is because the first character can be any of the 26 letters, and the second character can also be any of the 26 letters, resulting in 26 × 26 = 676 possible combinations.

You're absolutely right that the constraint of not repeating any of the ciphertext character combinations is crucial for ensuring unambiguous decryption. Additionally, the restriction to only the 26 letters of the alphabet (without any punctuation or special characters) is another important constraint to consider.