Deployment and Operations for Software Engineers 2nd Ed

Chapter 14—Secure Development

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

Items to protect

- Data
- Resources

Data to be protected

- There are several categories of data that must be protected.
  - User credentials for system access–user IDs and passwords.
  - Corporate sensitive data--business plans, sales data, product specifications
  - Personally identifiable information (PII)
  - Data subject to regulatory control. E.g. HIPAA
  - Data subject to contractual agreements. E..g. PCI

Sample data collection practices

- Avoid the collection of critical data (if possible) or minimize the amount the critical data acquired and retained.
- Encrypt critical data when at rest and in motion
- Use data models and schemas that separate critical information from other data.
- Do not expose critical data in logs.

Resources to be protected

- Critical resources are attacked for one of two reasons:
  - to inhibit the availability of a service or data(the A in CIA)
  - or as a means for gaining access to critical data (the C and I in CIA).
- Resources to be protected
  - CPU.
  - Memory.
  - Disk space.
  - Network access.
  - APIs.

Practices for securing web systems

- Sanitize inputs at the client side and server side.
- Encode request/response
- Use only current encryption and hashing algorithms.
- Do not allow HTTP/HTTPS requests to list a directory.
- Do not store sensitive data inside cookies.

Security reviews

- Services should be reviewed for security.
  - Requirements. E.g. Are potential adversaries identified explicitly?
  - Design,. E.g. What is the attack surface?
  - Development. E.g. Have secure coding techniques been used?

Discussion questions

1. What are the PCI requirements with respect to data?
2. Why should APIs be protected?

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

Authentication

- Authentication is the process of proving that you are who you say you are.
- Three categories of identification factors
  1. What you know. Items in this category are passwords, pin numbers, and keys.
  2. What you have. Items in this category are a smartphone with known number or a smart card
  3. What you are. Various forms of biometric identification (fingerprints, retinal scan) exist and the technology for recognizing biometric identifiers continues to improve.
- Two factor authentication (TFA) where the user must provide two different factors are becoming much more common.

LDAP

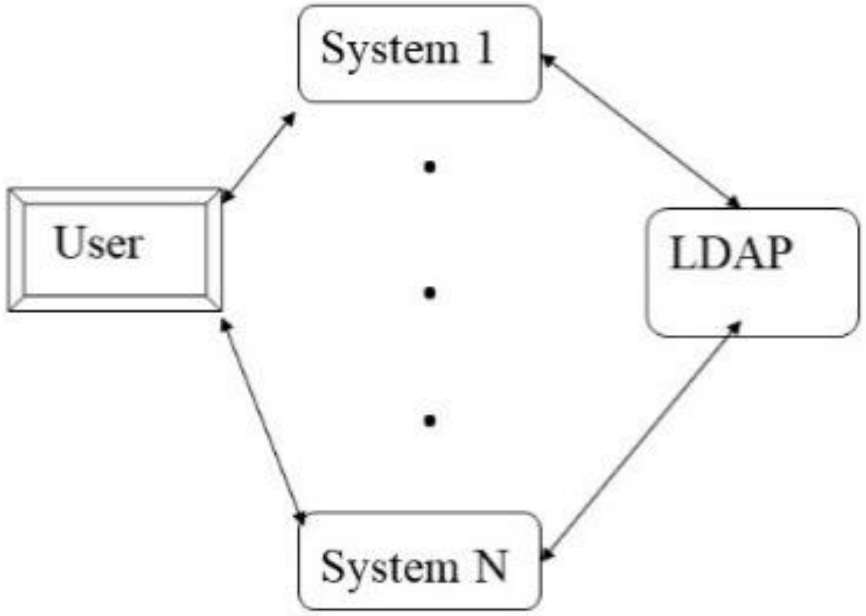
- The Lightweight Directory Access Protocol (LDAP) is the standard method for managing employee's credentials in an organization.
- Used when an organization has multiple systems that require authentication.
- LDAP has a central repository where all members of an organization are entered together with login ID and password.
- Systems are registered wth the LDAP service
- When a user signs on to a registered system, their credentials are sent to the LDAP service
- The LDAP Service authenticates the user.

Joining or leaving an organization

- When a new member joins the organization, they are given a login identifier and a method for authenticating themselves. These are entered into the LDAP service.
- When a member leaves the organization, their LDAP entry is deleted and they can no longer use the systems of the organization.

Single sign on

- Single Sign On (SSO) is a mechanism whereby providing credentials once is sufficient for all the systems in an organization
- Both the systems and the users must be registered with LDAP
- When a user signs on to one of the registered systems, their credentials are sent to LDAP and it returns a token.
- This token is sufficient for signing on to other registered systems.



Discussion questions

- The token returned during SSO is erphermal. What does that mean and why is it useful?
- Where is the token returned during SSO stored?

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

Authorization

- Authorization is the process of determining whether a user or system has the privileges to perform a specific function on a resource. The resource could be data, an API, a file, or a hardware resource.
- There are two ways of looking at authorization.
  - From th perspective of the resource, authorization is granted to a list of users, roles, or holders of credentials. Access Control Lists (ACLs) are typically used to enumerate the entities entitled to perform functions on the resources.
  - From the point view of the user or system., they have the *capability* to perform specific functions on a resource.

Authorization principles

- Authorization policy should be governed by a set of rules established by a system owner.
  - These rules should be independent of any implementation of the system. That is, the rules can be changed without requiring changes to the code
  - These rules should be version controlled.
- Least privilege. Users require different privileges to do their jobs. Any individual user should be granted the least privileges necessary for their job.

Role Based Access Control

- Role Based Access Control (RBAC) is a method for managing the credentials for groups of members of an organization.
- Every member is assigned to one or more roles. Credentials are assigned to roles.
- The roles an individual performs are recorded in the LDAP server.
- When an individual logs in, the credentials they are granted will be based on their role.
- When an individual changes their role, their new role is entered and their old role deleted.

Problems with RBAC

- Explosion of roles. Suppose in a bank with multiple branches, one branch gives managers the power to approve loans up to $10,000 and another branch gives managers power to approve loans up to $20,000. Are these two different roles? Managing the subtleties of the various roles becomes a problem.
- The same credentials are assigned to everyone is the same role. Suppose an individual leaves and they had database credentials. How are these credentials rescinded for that individual and not for the other individuals in that role?
  - This is one use case for rotation of credentials.
  - Rotation, in turn, is one use case for Vault.

Authorization solution

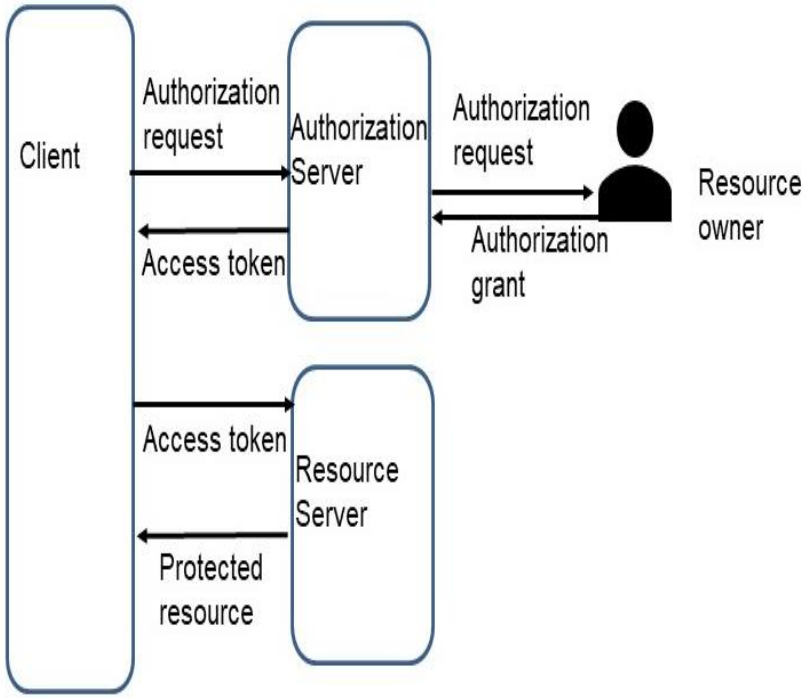
- A general solution for the authorization problem involves three different elements:
  - SSO
  - OAuth
  - A credential manager such as Vault

Oauth description

- OAuth (Open Authorization) protocol is a standard that replaces authentication credentials with *access tokens* that are created, exchanged among, and used by four roles.
  - The *resource owner* is the person (or service) that has the credentials to access the protected resource.
  - The *resource server* hosts the protected resource and can accept and use tokens to grant access to the resource.
  - An *OAuth client* that accesses the protected resource with the authorization token granted by the resource owner.
  - An *authorization server* issues access tokens to the OAuth client, after authenticating the resource owner and obtaining authorization.

OAuth flow

- The resource owner authenticates to the OAuth authorization server. This returns a single sign-on identification token that is saved so that the resource owner does not have to log in for every access to the protected API.
- The client gets an initial access token that can be used to register specific instances of the client. The client identifies itself to the OAuth authorization server. It specifies the access privileges available to any access through this instance.
- The credentials of the user of the client and of the client are then sent to the protected API, which verifies them with the OAuth server.



Credential management

- A credential is anything that can be used for authentication or authorization.
- Vault is a common open-source tool used to manage credentials.
- Vault functions:
  - Encrypting the credentials and the individual systems delegate the responsibility for saving and retrieving the credentials to Vault.
  - Providing access control
  - Providing an audit trail of who has accessed which credentials.
  - Rotation of credentials.
- Once there is a centralized solution to the problem of managing credentials, the credentials can be made ephemeral. Ephemeral credentials have expiration periods.

Putting together SSO, Oauth; Vault

- The following steps will provide secure access for systems and individuals.
  - The resource server is registered with Vault.
  - A system that acts as a client registers itself with Vault.
  - An individual stores their SSO token in Vault.
  - A client uses OAuth to request access to the resource.
  - The returned access token is stored by the resource into Vault.
  - The client retrieves the token from Vault.
  - Vault verifies that the client's credentials allow them access to the resource and returns the current access credential for the resource.
  - The client uses the access credential to access the resource.

Discussion questions

1. Suppose entry to a secured area is controlled by a keypad on the door. How is entry affected when an individual who had access leaves the organization?
2. Why is a centralized credential management system better than allowing applications to manage their own credentials?

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

Weaknesses and vulnerabilities

- Software weaknesses are errors that can lead to software vulnerabilities.
- A software vulnerability allows an attacker to gain access to a system or network.
- The CWE is the Common Weakness Enumeration catalog
- Vulnerabilities are listed in the CVE (Common Vulnerabilities and Exposures) catalog.
- A vulnerability is a specific weakness (or collection of weaknesses) in a particular software package and version that an attacker can use to gain access to a system.

Vulnerability Discovery

- Vulnerability is discovered, either by the developing organization or by an external individual.
- Vulnerability is reported to CERT Coordination Center (CERT/CC).
  - This starts the disclosure window (currently 45 days) before the vulnerability is publicly reported.
  - The vendor or open-source maintainer is privately notified of the vulnerability, and the disclosure window gives them time to prepare a patch and incentive to react before the vulnerability information becomes public.
- Vulnerability is publicly disclosed and listed in the CVE. This involves contacting the CVE maintainer (currently the MITRE Corporation) and receiving an identifier. This identifier is used to reference the vulnerability thereafter. The vulnerability is also entered into the National Vulnerability Database (NVD).

Vulnerability Patching

- The vendor or open-source project issues a patch, referencing the CVE ID.
- The patch is publicized by the vendor, and a link to the patch is placed in the NVD.
- You are informed of the patch either through monitoring of the vendor's mailing list or of the NVD, or through an automated patch management service.
- You apply the patch.

Problems with patch process

- Too much information.
  - There are many vulnerabilities being disclosed, each accompanied by a patch or other remediation.
  - You use packages produced by multiple vendors. Each vendor has multiple products, and each product has multiple patches.
- Scanning software will inform you of vulnerabilities in your executing service.

Deciding whether to apply a patch

- You must decide whether to apply the patch. This depends
  - on the severity of the problem that the patch fixes
  - the disruption to your organization from applying the patch to all the instances using that software package. Applying the patch is equivalent to releasing a new version of all your systems. Each system must be tested, the new version deployed, and old versions taken out of service.
- One option is not to apply the patch and rely on the next normal build of your system to get to the latest patch level. Some organizations rebuild all their systems daily or on a frequent periodic basis so that all their systems incorporate patches without special action by developers.

Discussion questions

1. What are the pluses and minuses of making vulnerabilities public?
2. Find a vulnerability for a tool in your deployment pipeline. Has a patch for this vulnerability been released? How serious is the vulnerability and what priority would you give applying the patch?

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

What is a supply chain?

- The software supply chain consists of libraries, code, hardware, and tools that transform code into a final service.
- Much of the software that is executed by your service, from the operating system to middleware and API frameworks, was developed by someone outside your organization as open source or commercial off-the-shelf software. This software is a portion of your supply chain.
- A portion of your supply chain consists of the tools and hardware that contribute to the production of your service.

Choosing open source software

- Some criteria for evaluating packages delivered by open- source projects are
  - Project maturity and development activity. Open source projects can be abandoned. The use of the project in other systems, in production, demonstrates a reasonable level of maturity.
  - Identified and engaged maintainer(s). Without oversight by an engaged maintainer, contributions can introduce vulnerabilities and malicious code.
  - Repository used for the project. The project should be housed in a modern repository with a defect-tracking system
  - Download-confirmation hash. This prevents a mirror site from changing the contents of the download.
  - Pedigree. Projects with commercial sponsorship may have more resources.

Principles for securing the supply chain

- Every step in a supply chain should be “trustworthy” as a result of a combination of cryptographic attestation and verification. No step in the supply chain should rely on assumptions about the trustworthiness of any previous steps or outputs — trust relationships must be explicitly defined.
- Second, automation is critical to supply chain security. Automating as much of the software supply chain as possible can significantly reduce the possibility of human error and configuration drift.

Principles for securing the supply chain

- Third, the build environments used in a supply chain should be clearly defined, with limited scope. The human and machine identities operating in those environments should be granted only the minimum permissions required to complete their assigned tasks.
- Fourth, all entities operating in the supply chain environment must be required to mutually authenticate using hardened authentication mechanisms with regular key rotation.

Securing Kubernetes

- Scan containers and Pods for vulnerabilities or misconfigurations.
- Run containers and Pods with the least privileges possible.
- Use network separation to control the amount of damage a compromise can cause.
- Use firewalls to limit unneeded network connectivity and encryption to protect confidentiality.

Securing Kubernetes

- Use strong authentication and authorization to limit user and administrator access, as well as to limit the attack surface.
- Use log auditing so that administrators can monitor activity and be alerted to potential malicious activity.
- Periodically review all Kubernetes settings and use vulnerability scans to help ensure risks are appropriately accounted for and security patches are applied.
- A variety of scanners exist in the marketplace that will examine the bill of materials of the containers in your system and report known vulnerabilities.

Discussion questions

1. Is a deployment pipeline built with tools from multiple vendors going to be more or less secure than one built with tools from a single vendor?
2. Jenkins is a widely used CI server. What company or companies fund maintenance of Jenkins?

Outline

What to protect

Authentication

Authorization

Weaknesses and vulnerabilities

Software supply chain

DevSecOps

What is DevSecOps

- DevSecOps=DevOps+Security
- Security has always been a portion of the description of the duties of the IT organization.
- DevSecOps serves to emphasize the security aspects of operations.

Impact of DevSecOps

- The following have all matured significantly in the past several years,
  - Authentication and authorization services including credential management
  - Monitoring of individual microservices and consolidated logging.
  - Secure communication between pods in Kubernetes (e.g. with TLS)
  - Tools that examine the bill of materials for containers and compare them to known vulnerabilities.
  - Securing the deployment pipeline
  - Securing the orchestration platform
  - Protecting Infrastructure as Code
  - Building secure container registries