Enumeration is the process of scanning a target system, network, or application and collecting information on it while in the process. Enumeration includes asking the system questions to get information such as usernames, machine names, shares, services, and other assets.

The information that can be collected during the enumeration phase can be utilized by an attacker to understand the structure and security of the targeted system so that the attacker would understand what comes next.

Enumeration can be used to gather any of the following information:

- Operating system details
- Network infrastructure details
- Usernames of valid users
- Machine names
- Share names
- Directory names
- Printer names
- Web server details

Why Is Enumeration Important?

enumeration can be used to find security vulnerabilities within systems and networks. By conducting an enumeration scan, you can see what ports are open on devices, which ones have access to specific services, and what type of information is being transmitted. This information can then be used to exploit weaknesses and gain unauthorized access.

Techniques for Enumeration

Several techniques can be used for enumeration, and your method will depend on the type of system you are targeting. The most common methods include email IDs and usernames, default passwords, and DNS zone transfer.

- Using email IDs and usernames is a great way to gather information about a system. You can use this information to brute force passwords or gain access to sensitive data. Default passwords are another common method of enumeration.
- By using default passwords, you can gain access to systems that have not been properly configured.
- DNS zone transfer is a technique that can be used to expose topological information. This information can be used to identify potential targets for attack.

The Types of Enumeration

There are many different types of enumeration. The most appropriate type will depend on the situation and the required information:

- **NetBIOS Enumeration:** NetBIOS is a protocol that allows devices on a network to share resources and communicate with each other. NetBIOS enumeration is querying a device to identify what NetBIOS resources are available. This can be done using tools like **nbtstat** and **net view**.
- **SNMP Enumeration:** SNMP is a protocol that allows devices to be managed and monitored remotely. SNMP enumeration is querying a device to identify what SNMP resources are available. This can be done using tools like **SNMP-check** and **snmpwalk**.
- **LDAP Enumeration:** LDAP is a protocol that allows devices on a network to share information about users and resources. LDAP enumeration is querying a device to identify what LDAP resources are available. This can be done using tools like **ldapsearch** and **ldapenum**.
- **NTP Enumeration:** NTP is a protocol that allows devices on a network to synchronize their clocks with each other. NTP enumeration is querying a device to identify what NTP resources are available. This can be done using tools like **Nmap** and **PRTG Network Monitor**

How to Protect Against Enumeration?

Understanding how enumeration works can help you implement strategies to defend against it. Some key protections include:

- **Firewalls**: Implement network-level protections to block unnecessary ports and

- **Access Control**: Use strong password policies, two-factor authentication, and user privilege
- **Intrusion Detection Systems (IDS)**: Deploy IDS to detect suspicious scanning or probing activity.
- **Network Segmentation**: Divide your network into segments to limit the exposure of sensitive
- **Encryption**: Use encryption to protect sensitive data, including passwords and network

Web Spidering Techniques

Spidering in web scraping involves using automated scripts to systematically browse the web and collect data from various websites. It helps in indexing, data mining, and content aggregation.

Web spidering, also known as web crawling, is a technique used to systematically explore and extract information from websites. In application and web security, it plays a crucial role in identifying vulnerabilities and weaknesses by mapping out the application's structure, discovering hidden directories, and finding potential attack vectors.

In web security

passive spidering involves gathering information about a website without directly interacting with it, relying on publicly available information and analysis of traffic.

Active spidering, on the other hand, involves directly interacting with the website by sending requests and analyzing responses, potentially triggering vulnerabilities.

The Impact of Spidering on Cybersecurity

Spidering, while essential for web indexing and data aggregation, poses significant cybersecurity risks. By systematically exploring and mapping out online presences, spidering can be exploited by malicious actors to gather sensitive information.

Data Harvesting: Collecting vast amounts of data, including personal and sensitive information.

Phishing Attacks: Crafting targeted phishing schemes using gathered data.

Server Overload: Overloading servers with aggressive crawling, leading to potential downtime.

Unauthorized Access: Gaining unauthorized access to restricted areas of websites.

Legal Issues: Ignoring robots.txt files, leading to potential legal consequences.

User-Directed Spidering/ manual spidering

User Directed Spidering with Burp is a great way to catch security vulnerabilities in a web application.

A spider is a software that goes through your website, following every link and looking for the next page it needs to visit. This means it can't get stuck when there are loops or missing links - which is what would happen with a regular browser like IE or Chrome.

Burp Spider:
Spidering is a browser security testing technique that requires using special tools in order to follow all possible paths through the application. These tools are designed in such a way that they can follow every visitor and every user action, as well as collect useful information while spidering.

manual spidering involves a human tester actively exploring the web application and its functionalities. The tester manually interacts with the application, following links, submitting forms, and analyzing the responses. This approach allows for a deeper understanding of the application's behavior and enables the tester to identify vulnerabilities that may be missed by automated tools.

manual spidering can be time-consuming and may not be suitable for large-scale applications with numerous pages. It heavily relies on the tester's skills, knowledge, and experience, which can introduce inconsistencies and subjectivity in the assessment process. Moreover, manual spidering may not be able to identify certain vulnerabilities that require extensive coverage or automated scanning techniques.

Advantages of User-Directed Spidering

User can follow unusual or complex navigation mechanisms

User can enter valid data where needed

User can log in as needed

User can avoid dangerous functionality, such as deleteUser.jsp

Automated spidering

Automated spidering, also known as automated crawling or web crawling, involves the use of specialized tools or scripts to automatically navigate through a web application. These tools simulate user interactions by sending HTTP requests to various URLs within the application and analyzing the responses received. The process is typically guided by predefined rules or algorithms that determine the traversal path of the spider.

One of the main advantages of automated spidering is its efficiency in covering a large number of pages within a short period. It can quickly identify common vulnerabilities, such as broken links, missing pages, or predictable URL patterns. Moreover, automated spidering can be performed repeatedly to ensure consistent results and facilitate regression testing.

Limitations of Automatic Spidering

May fail to handle unusual navigation mechanisms, such as dynamically created java script menus

So it may miss whole areas of an application

Links buried in compiled client-side objects like Flash or Java may be missed

Forms may have validation checks, such as user registration forms

Email address, telephone number, address, zip code

Too complex for most spiders, which use a single text string for all form fields

Spider cannot understand the "Invalid" error messages

Authentication: spider must be able to submit valid credentials

Perhaps using a valid cookie

However, spiders often break the authenticated session, by

Requesting a logout function

Submitting invalid input to a sensitive function

Requesting pages out-of-sequence

Automated spidering and manual spidering are two complementary approaches in web application penetration testing. While automated spidering offers efficiency and coverage, manual spidering provides flexibility and depth. A balanced combination of both methods can enhance the effectiveness of the testing process, ensuring a thorough assessment of the target scope.

Reference: -

https://eitca.org/cybersecurity/eitc-is-wapt-web-applications-penetration-testing/target-scope/target-scope-and-spidering/examination-review-target-scope-and-spidering/what-is-the-difference-between-automated-spidering-and-manual-spidering-in-web-application-penetration-testing/

Discovering Hidden Content

Content discovery in application and security refers to techniques used to find hidden or undocumented functionality, directories, files, and parameters within a web application or system. This is crucial for identifying potential security vulnerabilities, as attackers often exploit these hidden elements to gain unauthorized access or information.

Methods of Content Discovery

1. Manual Content Discovery

2. Automated Content Discovery

3.OSINT (Open-Source Intelligence)

Manual Discovery — robots.txt

The robots.txt file tells search engines which parts of a website they should not index. It is often used to hide admin portals, backups, or internal tools from being publicly listed.

Manual Discovery — Favicon

A **favicon** is the small icon displayed in a browser tab. Sometimes, default favicons from web development frameworks are left unchanged, revealing the technology stack used**.**

Manual Discovery — sitemap.xml

The sitemap.xml file provides search engines with a list of all pages the website owner wants to be indexed. It can sometimes reveal:

- Hidden or hard-to-navigate pages
- Legacy content still accessible but not linked on the sit

Manual Discovery — HTTP Headers

HTTP headers in server responses often reveal useful information, such as:

- Web server type and version
- Backend technologies (e.g., PHP, ASP.NET)
- Custom headers that may include flags or internal dat

Manual Discovery — Framework Stack

Identifying the framework a website uses can help uncover:

- Default paths (e.g., admin panels)
- Common misconfigurations
- Known vulnerabilities

Automated Discovery

Automated Discovery is the process of using tools to discover hidden content on a website through automated requests. These requests check for the existence of directories and files, helping discover resources that weren't previously known. This is typically achieved using **wordlists**, which contain commonly used directory and file names.

Tools Used:

1. **ffuf**
2. **dirb**
3. **gobuster**

OSINT — Google Hacking / Dorking

Google Dorking leverages advanced search operators to discover hidden or sensitive information indexed by Google.

site:site:tryhackme.com Show results only from the specified domain

inurl:inurl:admin Search URLs containing a specific keyword

filetype:filetype:pdf Find files with a specific extension

intitle: intitle: admin Show pages with specific words in the title

OSINT — Wappalyzer

Wappalyzer is a powerful OSINT tool (available both as a browser extension and an online service) that helps identify:

- Frameworks
- CMS (Content Management Systems)
- Analytics tools
- E-commerce platforms
- Server technologies
- And even version numbers

OSINT — Wayback Machine

The **Wayback Machine** is a digital archive that stores snapshots of websites over time. It allows you to view old versions of websites, helping uncover outdated or hidden content that may still be accessible**.**

OSINT — S3 Buckets

Amazon **S3 Buckets** are storage solutions by AWS, allowing files to be stored and accessed over HTTP/HTTPS. Misconfigured access permissions may inadvertently expose sensitive content to the public.

S3 Buckets are typically accessible through the URL format:

`http(s)://{name}.s3.amazonaws.com`

Reference: -

https://medium.com/@PareXploit/tryhackme-writeup-content-discovery-7ac3667714dd

https://rahulk2903.medium.com/content-discovery-tryhackme-walkthrough-e595b88181db

Application Pages Versus Functional Paths

Application security, application pages refer to the individual user interface components (like web pages or mobile screens) of an application, while functional paths represent the sequences of operations or actions users can take within the application to achieve specific goals. Analyzing both is crucial for identifying vulnerabilities and ensuring the application's security.

Application Pages:

- Represent the visual structure of the application.
- Can be mapped and categorized (e.g., login page, user profile page, product listing page).
- Each page may have its own set of functionalities and potential vulnerabilities.

Functional Paths:

- Describe the logical flows or sequences of actions users can perform within the application.
- Examples include logging in, searching for a product, adding an item to a cart, or completing a purchase.
- These paths can be complex and involve multiple pages or components.

Discovering Hidden Parameters

In web application security, hidden parameters are used by developers for testing or internal features. These parameters are not visible in the user interface but can still affect how the application works. If found and used properly, hidden parameters can help a penetration tester discover important issues like unauthorized access, admin panels and many more.

How to Find Hidden Parameters

Use a Web Proxy Tool (like Burp Suite):

- Intercept and inspect requests
- Look for parameters in GET, POST, cookies, headers
- Change parameter values and see the response

5 methods to discover hidden parameters, to identify any hidden parameters that lack adequate validation and are accessible to the end-user.

1. HTML input fields
2. JavaScript file enumeration
3. Google/GitHub/Wayback Machine enumeration
4. Parameter fuzzing
5. Re-use of parameters

Automated tools

Automated tooling can severely help speed up your repetitive scanning tasks, including the discovery of hidden parameters. Below are several free, open-source tools listed that can help passively and actively enumerate any unreferenced input parameters.

Arjun

Arjun is a popular open-source, Python-based parameter discovery tool. It provides support for multiple content types and HTTP request methods, ideal for discovering parameters in several contexts. Arjun is available on GitHub:

x8

x8 is a blazing-fast, Rust-powered hidden parameter discovery suite. With its extensive configuration support and the ability to use custom wordlists, it makes it easy to perform any type of parameter discovery scan. x8 is available on Github:

ParamMiner

ParamMiner is a simple-to-use plugin to actively find unreferenced parameters within Burp Suite. It also comes with several extensive wordlists to help you quickly start new bruteforcing scans. ParamMiner is available on the BApp Store and on GitHub:

GetAllParams (GAP)

GetAllParams (GAP) is an alternative Burp Suite extension capable of actively and passively detecting new parameters. The extension can also be used to generate custom wordlists, making this an indispensable tool. GAP is available on GitHub:

Identifying Entry Points for User Input

Entry points

Web or application pages consume HTTP requests and process incoming values. All these variables become entry points or source to the system. Identifying entry points for user input is crucial in application and web security because these are potential attack vectors

Query string as an entry point

Each incoming HTTP request can pass on values through query string to web/application pages. Here is an example

http://example.com/login.aspx?username=shah

In above case we have name value pair username & shah is passed as HTTP parameter to application page called login.aspx. If we look at actual HTTP request, then it would look like following.

```
GET /login.aspx?username=shah HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.0.1)
Gecko/2008070208 Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

it is GET request hitting to application pages along with some other HTTP headers. All these headers and name-value pair get processed by web pages.

POST-Name/Value pair as query string

We have passed value using GET request by querystring in above case, it is also possible to pass similar value by POST request where size of value can be unlimited. In querystring one can pass maximum buffer of 256 characters only

Here are some important different entry points with respect to Web 2.0 or non name-value pair based entry points.

XML-RPC – RPC (Remote Procedure Call) is very old concept to invoke procedures remotely. These remote calls were for underlying operating system which can be invoked from same machine or over the network. Over period to defend operating system some of the ports were

closed and only available open ports were 80 and 443. These ports are supporting HTTP protocol only

SOAP – SOAP (Simple Object Access Protocol) message is known as SOAP envelope and it gets exchanged between server and client. SOAP has specific skeleton for information exchange.

REST – REST(Representational State Transfer) REST is a modern Web application architecture style. It can fall in SOA as well. REST uses XML structures to communicate between client and server and vice a versa.

JSON – JSON (JavaScript Object Notation) is very light weight format for data exchange. JSON is getting popularity with developers for Web 2.0 applications. JSON is supported by various languages and scripts so it is easy to integrate and becoming very popular as well.

Tools for Identifying Entry Points:

1.Web Proxies:

Tools like OWASP ZAP, Burp Suite, and Fiddler can intercept and analyze HTTP requests to reveal entry points and data flow.

2.OWASP Dependency-Track:

This tool helps identify and manage vulnerabilities in third-party components used by the application.

3.OWASP Attack Surface Detector:

This tool helps in identifying the attack surface of an application by enumerating its entry points.

Identifying Server-Side Technologies

Server-side technology encompasses the tools and frameworks that operate on the backend of web applications, safeguarding sensitive data and maintaining the integrity of critical systems. By handling operations like authentication, encryption, and secure data management, it forms the first line of defense against cyber threats.

Unlike client-side technology, which operates in a user's browser and is often more vulnerable to tampering, server-side systems provide controlled environments where security measures can be tightly implemented and monitored.

Cybersecurity* Features *Enabled by Server-Side Technology

- **Data Encryption:**

Encrypts sensitive information like login credentials and financial transactions, ensuring that intercepted data remains unreadable.

- **User Authentication:**

Implements robust login mechanisms, including multi-factor authentication (MFA) and single sign-on (SSO), to verify user identities securely.

- **Secure Session Management:**

Protects active user sessions against hijacking or unauthorized access.

- **Input Validation:**

Prevents malicious inputs such as SQL injection and cross-site scripting (XSS) by sanitizing and validating user data.

- **Access Control:**

Ensures that users can only access the data and functionalities they are authorized to, minimizing the risk of privilege escalation.

How to Enhance Server-Side Security?

1. **Implement Strong Encryption Standards:**

Use protocols like TLS 1.3 for data transmission and AES encryption for data storage.

2. **Adopt Secure Coding Practices:**

Follow secure development guidelines to minimize vulnerabilities in server-side code.

3. **Regularly Patch and Update:**

Ensure server software, frameworks, and dependencies are up to date to protect against known exploits.

4. **Conduct Penetration Testing:**

Simulate cyberattacks to identify and address weaknesses in the server-side infrastructure.

5. **Enable Intrusion Detection Systems (IDS):**

Monitor server activities for signs of unauthorized access or malicious behavior.

Importance of Understanding Client-Side vs Server-Side Security

Client-side security focuses on the code and functionality in the user's browser, making it inherently more vulnerable to attacks since threat actors can easily access, analyze, and modify the code.

server-side security deals with the back-end infrastructure where sensitive data processing and storage occur. While server-side components are typically more protected within controlled environments, they can still be vulnerable to attacks such as SQL injection, cross-site scripting (XSS), and distributed denial of service (DDoS).

Best Practices for Securing Server-Side Applications secure

- **Input Validation:** Always validate and sanitize user inputs. Remember, not all input is good input!
- **Parameterized Queries:** Use parameterized queries to prevent SQL injection. It's like using a lock instead of a flimsy door.
- **Content Security Policy (CSP):** Implement CSP to mitigate XSS attacks. Think of it as a security guard for your web pages.
- **Access Control Lists (ACLs):** Define who can access what. No more "everyone gets in" parties!
- **Secure APIs:** Use authentication and authorization for APIs. Don't let just anyone waltz in!
- **Regular Security Audits:** Conduct regular audits to identify vulnerabilities. It's like a health check-up for your application.
- **Use HTTPS:** Always use HTTPS to encrypt data in transit. It's like sending your data in a secure envelope.
- **Security Headers:** Implement security headers like X-Content-Type-Options and X-Frame-Options. They're like extra locks on your doors.
- **Backup Data:** Regularly back up your data. If things go south, you'll want a safety net!

Threat	Description	Example
SQL Injection	Attackers inject malicious SQL queries to manipulate databases.	Using ' OR '1'='1′ in a login form to bypass authentication.
Cross-Site Scripting (XSS)	Injecting malicious scripts into web pages viewed by other users.	Displaying user comments without sanitization, allowing script execution.

Insecure Direct Object References (IDOR)	Accessing unauthorized data by manipulating URLs or parameters.	Changing a user ID in the URL to access another user's data.
Cross-Site Request Forgery (CSRF)	Tricking users into executing unwanted actions on a web application.	Sending a malicious link that performs actions on behalf of the user.
Denial of Service (DoS)	Overloading a server to make it unavailable to legitimate users.	Flooding a server with requests until it crashes.
Session Hijacking	Stealing session tokens to impersonate users.	Using a packet sniffer to capture session cookies.
Malware Injection	Injecting malicious code into applications to compromise them.	Uploading a malicious file that executes on the server.
Unvalidated Redirects and Forwards	Redirecting users to malicious sites without validation.	Using a URL parameter to redirect users to a phishing site.
Insufficient Logging and Monitoring	Failing to log and monitor activities can lead to undetected breaches.	Not logging failed login attempts, allowing brute-force attacks.
Configuration Errors	Misconfigured servers can expose sensitive data or services.	Leaving default passwords or open ports on a server.

Tools for Server-Side Application Security

Tool	Description	Use Case
OWASP ZAP	An open-source web application security scanner.	Finding vulnerabilities in your web applications.
Burp Suite	A popular tool for web application security testing.	Performing penetration testing on web applications.
SonarQube	A tool for continuous inspection of code quality.	Identifying bugs and vulnerabilities in your code.
Fortify	A static application security testing tool.	Analyzing source code for security vulnerabilities.
Veracode	A cloud-based application security testing platform.	Scanning applications for vulnerabilities before deployment.
Splunk	A tool for monitoring and analyzing machine-generated data.	Detecting and responding to security incidents.
Wireshark	A network protocol analyzer.	Monitoring network traffic for suspicious activity.
Nessus	A vulnerability scanner for identifying security flaws.	Scanning your network for vulnerabilities.

AppScan	A tool for dynamic application security testing.	Testing web applications for security vulnerabilities.
SSL Labs	A tool for testing SSL/TLS configurations.	Ensuring your HTTPS setup is secure.

Refference:

https://blog.heycoach.in/server-side-application-security/

Identifying Server-Side Functionality

Server-Side Request Forgery (SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing. Typically, SSRF vulnerabilities occur when an application receives a URL or part of it from an untrusted source and fails to validate or sanitize it.

Step 1**: Identify the Functionality** First, as an attacker, you need to find a functionality that could potentially be exploited for SSRF. In this case, you find that the web application fetches images from a URL provided by the user for their profile picture.

Step 2**: Manipulate the Input** The next step is to manipulate the input. Instead of a URL pointing to an image, you input an internal, non-routable IP address like http://192.168.0.1.

Step 3: **Analyze the Response** If the application throws an error or behaves abnormally (e.g., the server takes an unusually long time to respond or times out), this could be a sign that the server tried to connect to that internal IP address. This might indicate the presence of an SSRF vulnerability.

Step 4: **Verify the Vulnerability** To confirm**,** use an external server you control (for example, https://webhook.site) and put its URL in the picture URL field. If your server logs an incoming connection from the web application, this confirms the SSRF vulnerability.

Step 5: **Exploit the Vulnerability** Now that you've confirmed the SSRF vulnerability, it can be exploited to carry out malicious actions. For example, you could use this vulnerability to make requests to internal network services that are normally not accessible from the outside. As an example, if there was a database management system running on http://192.168.0.2:8080, you could potentially use the SSRF vulnerability to interact with it.

Mapping the Attack Surface

Attack surface mapping and analysis is the process of identifying and examining all the possible points where an unauthorized user can try to enter or extract data from an environment. Think of it as creating a detailed map of all the doors, windows, and vents of a building. This map then helps in understanding where the defenses are strong and where they need reinforcement.

The main goal is to find every spot an attacker might use to gain access or cause harm, whether it's through a website, an email account, or even physical access to a computer or device.

Attack Surface Analysis is usually done by security architects and pen testers, developers should understand and monitor the Attack Surface as they design and build and change a system.

Attack Surface Analysis helps you to:

1. identify what functions and what parts of the system you need to review/test for security vulnerabilities
2. identify high risk areas of code that require defense-in-depth protection - what parts of the system that you need to defend
3. identify when you have changed the attack surface and need to do some kind of threat assessment

Why Is Attack Surface Mapping Important?

Attack surface mapping is crucial because it provides a detailed understanding of where a system is most vulnerable to attacks. This knowledge allows organizations to:

- **Identify weaknesses**: Pinpoint specific areas that need fortification.

- **Prioritize security efforts**: Allocate resources and efforts to the most critical vulnerabilities.
- **Improve incident response**: Enhance the ability to detect, respond to, and recover from security incidents.
- **Support compliance**: Ensure adherence to regulatory requirements and industry standards.
- **Reduce risk**: Minimize the likelihood of successful attacks and data breaches.

Types of attack surfaces

1.Network

This includes all the devices and connections that make up your organization's network. It's not only the hardware, like routers and switches, but also the protocols and services running on them. Monitoring and securing this surface involves managing who can access the network and what they can do once they're on it.

2.Application

Applications, whether developed in-house or acquired from third parties, can have vulnerabilities that attackers exploit. This surface covers all the software your organization uses, from email clients to enterprise resource planning systems. Securing it means regularly updating applications and checking for vulnerabilities.

3.Endpoint

Endpoints are the devices that connect to your network, like laptops, smartphones, and tablets. Each device is a potential entry point for attackers. Protecting endpoints involves installing security software, enforcing strong authentication, and educating users about safe practices.

4.Human

Unfortunately, people are often the weakest link in security. This surface encompasses the actions and behaviors of people within the organization, such as how they handle sensitive information or respond to suspicious emails. Strengthening this surface involves regular training and awareness programs.

5.Physical

Physical security focuses on protecting the tangible aspects of your organization, such as buildings, servers, and workstations. It includes measures like access control systems, surveillance cameras, and secure equipment disposal.

6.Cloud

As organizations move more of their operations to the cloud, this surface becomes increasingly important. It includes the data stored in cloud services, the cloud-based applications, and the infrastructure itself. Security measures include encryption, access controls, and collaboration with cloud providers to ensure security standards are met.

7.Supply chain

Your organization's security is only as strong as the weakest link in its supply chain. This surface includes all the third-party services and products that you rely on. To manage this risk, organizations must assess the security practices of suppliers and establish clear requirements.

8.Wireless

Wireless networks and devices add convenience, but also vulnerabilities. This surface covers all wireless communication within an organization, including Wi-Fi, Bluetooth, and NFC (near-field communications). Protecting it involves securing wireless networks and monitoring them for unauthorized access.

9.IoT devices

The Internet of Things (IoT) has expanded the attack surface dramatically, with connected devices ranging from cameras to industrial control systems. These devices often lack robust security features, making them easy targets. Security measures include segmenting IoT devices onto separate networks and regularly updating their firmware.

How to Reduce Your Internal Attack Surface

1.Implement a Zero Trust Policy

Zero trust policy requires all users, inside or outside an organization's network, to be authorized, authenticated, and continuously validated for security purposes. In other words, no user should have access to your assets until they have proven their identity. This model revolves around a mindset that puts security over convenience to minimize attack surfaces.

2.Safeguard Your Backups

Backups of data and code are widespread attack surfaces that hackers exploit. Applying strict protection protocols like immutability is a good rule of thumb to protect your backups. These protocols may include access restrictions and evaluating the vendor's security measures.

3.Maintain the Principle of Least Privilege

Organizations should restrict access to their resources and sensitive data, both internally and externally. In an average company, people continuously move in and out of work. Access permissions should be revoked as soon as a person leaves your organization.

4.Regularly Scan Your Digital Assets

Digital assets, like repositories, credentials, API keys, and users, present vulnerability risks. As your company's resources increase, so does your attack surface. You must automate your asset scanning and maintain it regularly to keep things working.

5.Secure SaaS Apps in Your Organization

Securing the authorized and unauthorized SaaS applications used in your organization is also critical in minimizing the attack points hackers can exploit. A SaaS security tool can help you discover all the apps employee's login, identify, and help you remediate the security risks in those applications

REFERENCE

https://jumpcloud.com/blog/attack-surface-mapping

https://www.sentinelone.com/cybersecurity-101/cybersecurity/attack-surface-mapping/

https://www.cycognito.com/learn/attack-surface/attack-surface-mapping.php

https://jetpack.com/resources/attack-surface-mapping-and-analysis/