



INFORMATION GATHERING - WEB EDITION

CHEAT SHEET

Cheat Sheet

Web reconnaissance is the first step in any security assessment or penetration testing engagement. It's akin to a detective's initial investigation, meticulously gathering clues and evidence about a target before formulating a plan of action. In the digital realm, this translates to accumulating information about a website or web application to identify potential vulnerabilities, security misconfigurations, and valuable assets.

The primary goals of web reconnaissance revolve around gaining a comprehensive understanding of the target's digital footprint. This includes:

- **Identifying Assets:** Discovering all associated domains, subdomains, and IP addresses provides a map of the target's online presence.
- **Uncovering Hidden Information:** Web reconnaissance aims to uncover directories, files, and technologies that are not readily apparent and could serve as entry points for an attacker.
- **Analyzing the Attack Surface:** By identifying open ports, running services, and software versions, you can assess the potential vulnerabilities and weaknesses of the target.
- **Gathering Intelligence:** Collecting information about employees, email addresses, and technologies used can aid in social engineering attacks or identifying specific vulnerabilities associated with certain software.

Web reconnaissance can be conducted using either active or passive techniques, each with its own advantages and drawbacks:

Type	Description	Risk of Detection	Examples
Active Reconnaissance	Involves directly interacting with the target system, such as sending probes or requests.	Higher	Port scanning, vulnerability scanning, network mapping
Passive Reconnaissance	Gathers information without directly interacting with the target, relying on publicly available data.	Lower	Search engine queries, WHOIS lookups, DNS enumeration, web archive analysis, social media

WHOIS

WHOIS is a query and response protocol used to retrieve information about domain names, IP addresses, and other internet resources. It's essentially a directory service that details who owns a domain, when it was registered, contact

information, and more. In the context of web reconnaissance, WHOIS lookups can be a valuable source of information, potentially revealing the identity of the website owner, their contact information, and other details that could be used for further investigation or social engineering attacks.

For example, if you wanted to find out who owns the domain `example.com`, you could run the following command in your terminal:

```
whois example.com
```

This would return a wealth of information, including the registrar, registration, and expiration dates, nameservers, and contact information for the domain owner.

However, it's important to note that WHOIS data can be inaccurate or intentionally obscured, so it's always wise to verify the information from multiple sources. Privacy services can also mask the true owner of a domain, making it more difficult to obtain accurate information through WHOIS.

DNS

The Domain Name System (DNS) functions as the internet's GPS, translating user-friendly domain names into the numerical IP addresses computers use to communicate. Like GPS converting a destination's name into coordinates, DNS ensures your browser reaches the correct website by matching its name with its IP address. This eliminates memorizing complex numerical addresses, making web navigation seamless and efficient.

The `dig` command allows you to query DNS servers directly, retrieving specific information about domain names. For instance, if you want to find the IP address associated with `example.com`, you can execute the following command:

```
dig example.com A
```

This command instructs `dig` to query the DNS for the `A` record (which maps a hostname to an IPv4 address) of `example.com`. The output will typically include the requested IP address, along with additional details about the query and response. By mastering the `dig` command and understanding the various DNS record types, you gain the ability to extract valuable information about a target's infrastructure and online presence.

DNS servers store various types of records, each serving a specific purpose:

Record Type	Description
A	Maps a hostname to an IPv4 address.
AAAA	Maps a hostname to an IPv6 address.
CNAME	Creates an alias for a hostname, pointing it to another hostname.

Record Type	Description
MX	Specifies mail servers responsible for handling email for the domain.
NS	Delegates a DNS zone to a specific authoritative name server.
TXT	Stores arbitrary text information.
SOA	Contains administrative information about a DNS zone.

Subdomains

Subdomains are essentially extensions of a primary domain name, often used to organize different sections or services within a website. For example, a company might use `mail.example.com` for their email server or `blog.example.com` for their blog.

From a reconnaissance perspective, subdomains are incredibly valuable. They can expose additional attack surfaces, reveal hidden services, and provide clues about the internal structure of a target's network. Subdomains might host development servers, staging environments, or even forgotten applications that haven't been properly secured.

The process of discovering subdomains is known as subdomain enumeration. There are two main approaches to subdomain enumeration:

Approach	Description	Examples
Active Enumeration	Directly interacts with the target's DNS servers or utilizes tools to probe for subdomains.	Brute-forcing, DNS zone transfers
Passive Enumeration	Collects information about subdomains without directly interacting with the target, relying on public sources.	Certificate Transparency (CT) logs, search engine queries

Active enumeration can be more thorough but carries a higher risk of detection. Conversely, passive enumeration is stealthier but may not uncover all subdomains. Combining both techniques can significantly increase the likelihood of discovering a comprehensive list of subdomains associated with your target, expanding your understanding of their online presence and potential vulnerabilities.

Subdomain Brute-Forcing

Subdomain brute-forcing is a proactive technique used in web reconnaissance to uncover subdomains that may not be readily apparent through passive methods. It involves systematically generating many potential subdomain names and testing them against the target's DNS server to see if they exist. This approach can unveil hidden subdomains that may host valuable information, development servers, or vulnerable applications.

One of the most versatile tools for subdomain brute-forcing is `dnsenum`. This powerful command-line tool combines various DNS enumeration techniques, including dictionary-based brute-forcing, to uncover subdomains associated with your target.

To use **dnsenum** for subdomain brute-forcing, you'll typically provide it with the target domain and a wordlist containing potential subdomain names. The tool will then systematically query the DNS server for each potential subdomain and report any that exist.

For example, the following command would attempt to brute-force subdomains of **example.com** using a wordlist named **subdomains.txt**:

```
dnsenum example.com -f subdomains.txt
```

Zone Transfers

DNS zone transfers, also known as AXFR (Asynchronous Full Transfer) requests, offer a potential goldmine of information for web reconnaissance. A zone transfer is a mechanism for replicating DNS data across servers. When a zone transfer is successful, it provides a complete copy of the DNS zone file, which contains a wealth of details about the target domain.

This zone file lists all the domain's subdomains, their associated IP addresses, mail server configurations, and other DNS records. This is akin to obtaining a blueprint of the target's DNS infrastructure for a reconnaissance expert.

To attempt a zone transfer, you can use the **dig** command with the **axfr** (full zone transfer) option. For example, to request a zone transfer from the DNS server **ns1.example.com** for the domain **example.com**, you would execute:

```
dig @ns1.example.com example.com axfr
```

However, zone transfers are not always permitted. Many DNS servers are configured to restrict zone transfers to authorized secondary servers only. Misconfigured servers, though, may allow zone transfers from any source, inadvertently exposing sensitive information.

Virtual Hosts

Virtual hosting is a technique that allows multiple websites to share a single IP address. Each website is associated with a unique hostname, which is used to direct incoming requests to the correct site. This can be a cost-effective way for organizations to host multiple websites on a single server, but it can also create a challenge for web reconnaissance.

Since multiple websites share the same IP address, simply scanning the IP won't reveal all the hosted sites. You need a tool that can test different hostnames against the IP address to see which ones respond.

Gobuster is a versatile tool that can be used for various types of brute-forcing, including virtual host discovery. Its **vhost** mode is designed to enumerate virtual hosts by sending requests to the target IP address with different hostnames. If a virtual host is configured for a specific hostname, Gobuster will receive a response from the web server.

To use Gobuster to brute-force virtual hosts, you'll need a wordlist containing potential hostnames. Here's an example command:

```
gobuster vhost -u http://192.0.2.1 -w hostnames.txt
```

In this example, **-u** specifies the target IP address, and **-w** specifies the wordlist file. Gobuster will then systematically try each hostname in the wordlist and report any that results in a valid response from the web server.

Certificate Transparency (CT) Logs

Certificate Transparency (CT) logs offer a treasure trove of subdomain information for passive reconnaissance. These publicly accessible logs record SSL/TLS certificates issued for domains and their subdomains, serving as a security

measure to prevent fraudulent certificates. For reconnaissance, they offer a window into potentially overlooked subdomains.

The **crt.sh** website provides a searchable interface for CT logs. To efficiently extract subdomains using **crt.sh** within your terminal, you can use a command like this:

```
curl -s "https://crt.sh/?q=%25.example.com&output=json" | jq -r '[][.name_value]' | sed 's/\*\.//g' | sort -u
```

This command fetches JSON-formatted data from **crt.sh** for **example.com** (the % is a wildcard), extracts domain names using **jq**, removes any wildcard prefixes (*.) with **sed**, and finally sorts and deduplicates the results.

Web Crawling

Web crawling is the automated exploration of a website's structure. A web crawler, or spider, systematically navigates through web pages by following links, mimicking a user's browsing behavior. This process maps out the site's architecture and gathers valuable information embedded within the pages.

A crucial file that guides web crawlers is **robots.txt**. This file resides in a website's root directory and dictates which areas are off-limits for crawlers. Analyzing **robots.txt** can reveal hidden directories or sensitive areas that the website owner doesn't want to be indexed by search engines.

Scrapy is a powerful and efficient Python framework for large-scale web crawling and scraping projects. It provides a structured approach to defining crawling rules, extracting data, and handling various output formats.

Here's a basic Scrapy spider example to extract links from **example.com**:

```
import scrapy

class ExampleSpider(scrapy.Spider):
    name = "example"
    start_urls = ['http://example.com/']

    def parse(self, response):
        for link in response.css('a::attr(href)').getall():
            if any(link.endswith(ext) for ext in self.interesting_extensions):
                yield {"file": link}
            elif not link.startswith("#") and not link.startswith("mailto:"):
                yield response.follow(link, callback=self.parse)
```

After running the Scrapy spider, you'll have a file containing scraped data (e.g., **example_data.json**). You can analyze these results using standard command-line tools. For instance, to extract all links:

```
jq -r '[][.file != null]' | select(.file != null) | .file example_data.json | sort -u
```

This command uses **jq** to extract links, **awk** to isolate file extensions, **sort** to order them, and **uniq -c** to count their occurrences. By scrutinizing the extracted data, you can identify patterns, anomalies, or sensitive files that might be of interest for further investigation.

Search Engine Discovery

Leveraging search engines for reconnaissance involves utilizing their vast indexes of web content to uncover information about your target. This passive technique, often referred to as Open Source Intelligence (OSINT) gathering, can yield valuable insights without directly interacting with the target's systems.

By employing advanced search operators and specialized queries known as "Google Dorks," you can pinpoint specific information buried within search results. Here's a table of some useful search operators for web reconnaissance:

Operator	Description	Example
site:	Restricts search results to a specific website.	site:example.com "password reset"
inurl:	Searches for a specific term in the URL of a page.	inurl:admin login
filetype:	Limits results to files of a specific type.	filetype:pdf "confidential report"
intitle:	Searches for a term within the title of a page.	intitle:"index of" /backup
cache:	Shows the cached version of a webpage.	cache:example.com
"search term"	Searches for the exact phrase within quotation marks.	"internal error" site:example.com
OR	Combines multiple search terms.	inurl:admin OR inurl:login
-	Excludes specific terms from search results.	inurl:admin -intext:wordpress

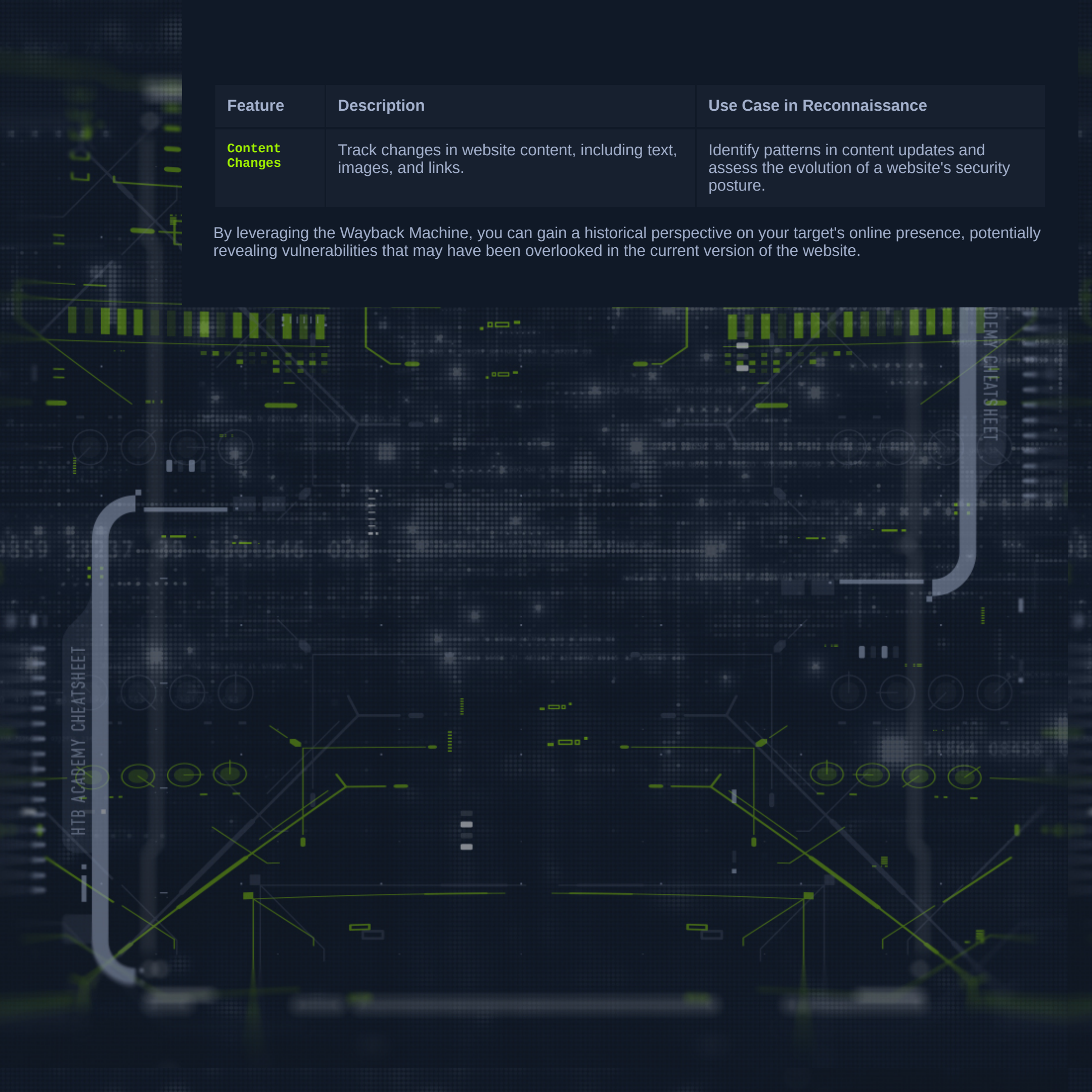
By creatively combining these operators and crafting targeted queries, you can uncover sensitive documents, exposed directories, login pages, and other valuable information that may aid in your reconnaissance efforts.

Web Archives

Web archives are digital repositories that store snapshots of websites across time, providing a historical record of their evolution. Among these archives, the Wayback Machine is the most comprehensive and accessible resource for web reconnaissance.

The Wayback Machine, a project by the Internet Archive, has been archiving the web for over two decades, capturing billions of web pages from across the globe. This massive historical data collection can be an invaluable resource for security researchers and investigators.

Feature	Description	Use Case in Reconnaissance
Historical Snapshots	View past versions of websites, including pages, content, and design changes.	Identify past website content or functionality that is no longer available.
Hidden Directories	Explore directories and files that may have been removed or hidden from the current version of the website.	Discover sensitive information or backups that were inadvertently left accessible in previous versions.

The background of the entire page is a dark blue-grey grid with various yellow and white geometric shapes, lines, and icons. On the left side, there is a vertical grey bar with the text "HTB ACADEMY CHEATSHEET" written vertically. On the right side, there is another vertical grey bar with the text "DEMY CHEATSHEET" written vertically. The table is positioned in the upper right quadrant of the page.

Feature	Description	Use Case in Reconnaissance
Content Changes	Track changes in website content, including text, images, and links.	Identify patterns in content updates and assess the evolution of a website's security posture.

By leveraging the Wayback Machine, you can gain a historical perspective on your target's online presence, potentially revealing vulnerabilities that may have been overlooked in the current version of the website.