ETHICAL HACKING FOOTPRINTING

AMASS

Detail about Amass

BY: Nin Kanong

Amass

OWASP®

Beginners guide to amass

WASP Amass used mainly to find assets mapped to a particular domain, perform sub-domain enumeration, autonomous system numbers (ASNs) etc. Although there are many other tools that can enumerate sub-domains etc. (for example gobuster), this tool as you can see is backed by OWASP. Let's see how to use this tool to discover assets of an organization.

Overview of Amass

Amass performs in-depth attack surface mapping and asset discovery using open-source intelligence (OSINT) and active reconnaissance techniques. It supports features like:

- **Subdomain Enumeration**: Discovering subdomains of a target domain.
- **DNS Enumeration**: Collecting DNS records (hostnames, IP addresses, etc.).
- **Network Mapping**: Identifying network ranges and Autonomous System Numbers (ASNs).
- **Visualization**: Generating graphical representations of findings.
- **Data Correlation**: Integrating data from multiple sources (e.g., APIs, web scraping, certificates).
  Graph Database: Storing findings for reuse and analysis.

Amass has five primary subcommands: **intel, enum, viz, track**, and **db**. Each serves a specific purpose in the reconnaissance process. The -demo flag, available in some subcommands, obfuscates output for presentations without revealing sensitive target information.

Installation

Before using Amass, ensure it's installed. Here are common installation methods:

+ Linux:

- kali> go install -v github.com/owasp-amass/amass/v3/...@master

Amass is installed by default in almost all pen testing distros. For this blogpost, we will be using Kali Linux. It doesn't have a man page yet, but we can see all the options it supports using the help option.

Amass Subcommands and Common Commands

Below are the subcommands, their purposes, common flags, and example commands, including the -demo flag where applicable.

1. intel: Collects open-source intelligence to discover root domains, ASNs, and other assets.

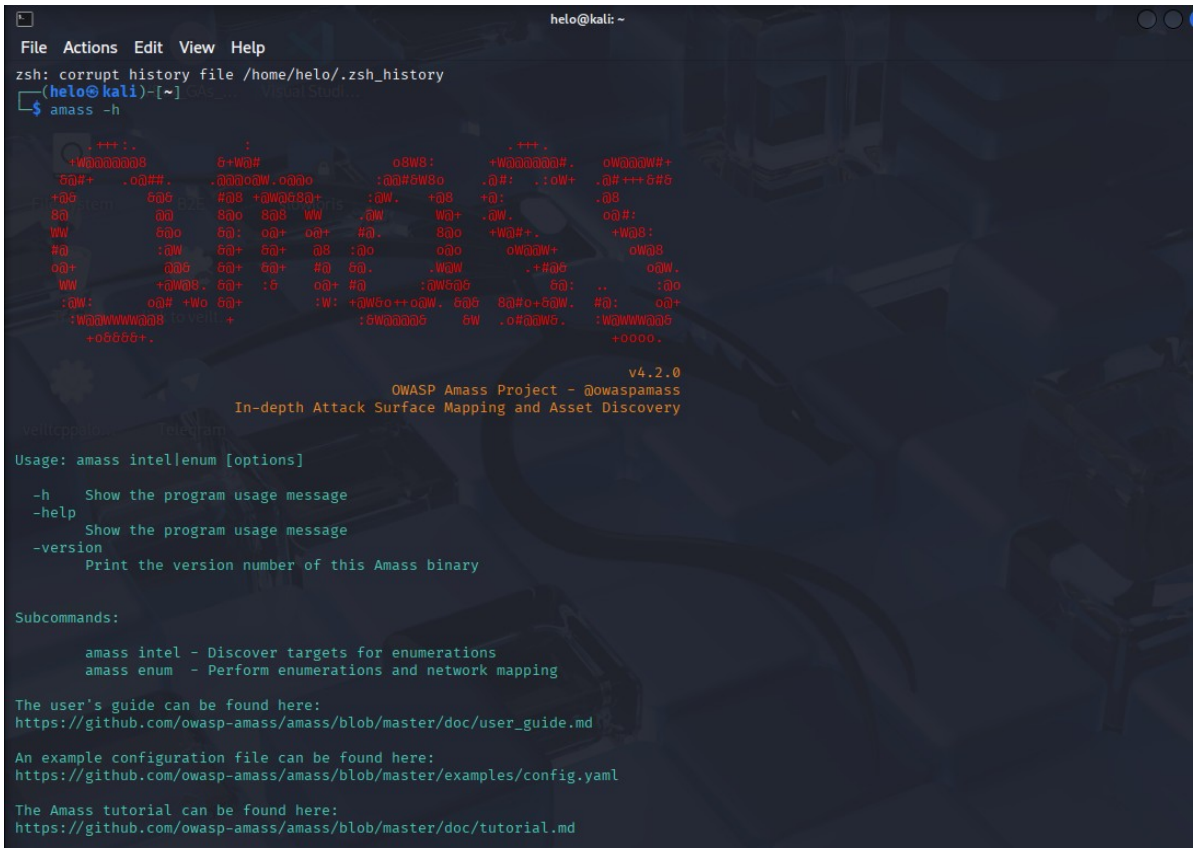
- **Purpose**: Identify targets (e.g., domains, IP ranges, ASNs) for further enumeration.
- **Common Flags**:
  - **-d** <domain>: Specify the target domain.
  - **-org** <organization>: Search by organization name.

- **-asn** <ASN>: Search by Autonomous System Number.
- **-cidr** <CIDR>: Search by IP range.
- **-whois**: Perform reverse WHOIS lookup.
- **-active**: Enable active reconnaissance (e.g., pulling SSL certificates).
- **-ip**: Include IP addresses in output.
- **-demo**: Obfuscate output for demonstrations.
- **-config** <file>: Specify configuration file with API keys.

2. enum: Performs DNS enumeration and network mapping to discover subdomains and assets.

- **Purpose**: Core enumeration tool for finding subdomains and mapping network infrastructure.
- **Common Flags**:
  - **-d** <domain>: Target domain for enumeration.
  - **-df** <file>: File containing list of domains.
  - **-active**: Perform active reconnaissance (e.g., DNS queries, certificate pulls).
  - **-passive**: Use only passive sources (faster but less accurate).
  - **-brute**: Enable brute-forcing of subdomains.
  - **-src**: Show data sources for each finding.
  - **-aw** <wordlist>: Use a custom wordlist for brute-forcing.
  - **-p** <ports>: Specify ports to scan (e.g., 443,8080).
  - **-o** <file>: Save output to a file.
  - **-demo**: Obfuscate output for demonstrations.
  - **-max-dns-queries** <n>: Limit DNS queries for rate control.

- kali> amass -h



Amass has 5 subcommands as shown below.

```
Subcommands:

        amass intel - Discover targets for enumerat
ions
        amass enum  - Perform enumerations and netw
ork mapping
        amass viz   - Visualize enumeration results
        amass track - Track differences between enu
merations
        amass db    - Manipulate the Amass graph da
tabase
```


Each subcommand has its own help section. For example, let's see the "intel" subcommand first.

Amass intel

- kali> amass intel -h

```
helo@kali: ~
File  Actions  Edit  View  Help
┌──(helo㉿kali)-[~]
└─$ amass intel -h

                                                    v4.2.0
                                OWASP Amass Project - @owaspamass
                    In-depth Attack Surface Mapping and Asset Discovery

Usage: amass intel [options] [-whois -d DOMAIN] [-addr ADDR -asn ASN -cidr CIDR]

  -active
        Attempt certificate name grabs
  -addr value
        IPs and ranges (192.168.1.1-254) separated by commas
  -asn value
        ASNs separated by commas (can be used multiple times)
  -cidr value
        CIDRs separated by commas (can be used multiple times)
  -config string
        Path to the YAML configuration file. Additional details below
  -d value
        Domain names separated by commas (can be used multiple times)
  -demo
        Censor output to make it suitable for demonstrations
  -df value
        Path to a file providing root domain names
  -dir string
        Path to the directory containing the output files
  -ef string
        Path to a file providing data sources to exclude
  -exclude value
        Data source names separated by commas to be excluded
  -h    Show the program usage message
  -help
        Show the program usage message
  -if string
        Path to a file providing data sources to include
  -include value
        Data source names separated by commas to be included
  -ip
        Show the IP addresses for discovered names
```

The ‘intel’ subcommand is used to discover targets to perform enumeration later. We can specify an IP address, IP address range, domain etc as targets to this command.

- kali> amass intel -whois -d example.com

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d google.com
```

Command Breakdown

- **Subcommand**: intel – Collects open-source intelligence, such as domains tied to WHOIS data.
- **Flag**: -whois – Performs reverse WHOIS lookup to find domains registered with similar WHOIS details.
- **Flag**: -d youtube.com – Specifies the target domain (youtube.com).

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -o youtube_domains.txt
```

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -demo
```

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -ip
```

Include IP addresses in the output

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -config config.ini
```

Use a config file for API keys

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -demo -o demo_output.txt
```

```
┌──(helo㉿kali)-[~]
└─$ amass intel -whois -d youtube.com -v
```

amass enum

This sub command is used to perform enumeration and network mapping of the discovered targets.

```
┌──(helo㉿kali)-[~]
└─$ amass enum -h
```

Using it, we can perform DNS enumeration too. All the findings of “amass enum” command are stored in a graph database, which is located in the amass’s default output folder. To enumerate subdomains of a domain using amass enum, this is the command.

- kali> amass enum -d example.com -whois

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org
owasp.org (FQDN) --> mx_record --> alt3.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> alt4.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> alt2.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> alt1.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> ns_record --> fay.ns.cloudflare.com (FQDN)
owasp.org (FQDN) --> ns_record --> west.ns.cloudflare.com (FQDN)
owasp.org (FQDN) --> a_record --> 104.22.27.77 (IPAddress)
owasp.org (FQDN) --> a_record --> 172.67.10.39 (IPAddress)
owasp.org (FQDN) --> a_record --> 104.22.26.77 (IPAddress)
owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::6816:1b4d (IPAddress)
owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::ac43:a27 (IPAddress)
owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::6816:1a4d (IPAddress)
www.owasp.org (FQDN) --> a_record --> 172.67.10.39 (IPAddress)
www.owasp.org (FQDN) --> a_record --> 104.22.27.77 (IPAddress)
www.owasp.org (FQDN) --> a_record --> 104.22.26.77 (IPAddress)
www.owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::ac43:a27 (IPAddress)
www.owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::6816:1a4d (IPAddress)
www.owasp.org (FQDN) --> aaaa_record --> 2606:4700:10::6816:1b4d (IPAddress)
172.67.0.0/16 (Netblock) --> contains --> 172.67.10.39 (IPAddress)
13335 (ASN) --> managed_by --> CLOUDFLARENET - Cloudflare, Inc. (RIROrganization)
13335 (ASN) --> announces --> 172.67.0.0/16 (Netblock)
```

Command Breakdown

- **Subcommand**: enum – Performs DNS enumeration to discover subdomains and related network assets.
- **Flag**: -d owasp.org – Specifies the target domain (owasp.org) for enumeration.

- kali> amass enum -d example.com -o subdomains.txt

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -o subdomains.txt
owasp.org (FQDN) --> mx_record --> alt3.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> alt4.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> alt2.aspmx.l.google.com (FQDN)
owasp.org (FQDN) --> mx_record --> aspmx.l.google.com (FQDN)
```

- kali> amass enum -d owasp.org -demo

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -demo
```

+ Passive Enumeration (faster, uses only OSINT sources):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -passive -d owasp.org
```

+ Active Enumeration (includes DNS queries, certificate pulls):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -active -d owasp.org
```

+ Brute-Force Subdomains (guesses subdomains using a wordlist):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -brute -d owasp.org
```

+ Include Source Information (shows which data source found each subdomain):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -src
```

+ Specify Ports (scans for services on specified ports):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -p 80,443,8080
```

+ Use a Custom Wordlist (for brute-forcing):

```
┌──(helo㉿kali)-[~]
└─$ amass enum -brute -aw pwdlist.txt -d owasp.org
```

+ Demo Mode with Output File:

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -demo -o demo_subdomains.txt
```

Notes

- **Output Accuracy**: The actual subdomains depend on owasp.org's DNS configuration and Amass's data sources. Public organizations like OWASP may have many subdomains (e.g., for projects, events, or documentation).
- **Performance**: Passive mode (-passive) is faster but less thorough. Active mode (-active) may yield more results but takes longer and may trigger rate limits.
- **API Keys**: Free-tier API keys (e.g., VirusTotal, Censys) enhance results. Without them, Amass relies on public sources, which may be limited.
- **Verification**: Validate subdomains using tools like dig or nslookup:

```
┌──(helo㉿kali)-[~]
└─$ dig www.owasp.org

; <<>> DiG 9.20.9-1-Debian <<>> www.owasp.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40890
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.owasp.org.                 IN      A

;; ANSWER SECTION:
www.owasp.org.          289     IN      A       104.22.27.77
www.owasp.org.          289     IN      A       172.67.10.39
www.owasp.org.          289     IN      A       104.22.26.77

;; Query time: 0 msec
;; SERVER: 10.10.9.101#53(10.10.9.101) (UDP)
;; WHEN: Wed Jul 02 07:09:53 CDT 2025
;; MSG SIZE  rcvd: 90
```

+ Graph Database: Results are stored in ~/.config/amass/. Use -dir to specify a custom directory:

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -dir owasp_output
```

Troubleshooting

- **No Subdomains Found**: Check API key configuration or try -active or -brute flags.
- **Rate Limits**: Use -max-dns-queries <n> to limit DNS queries:

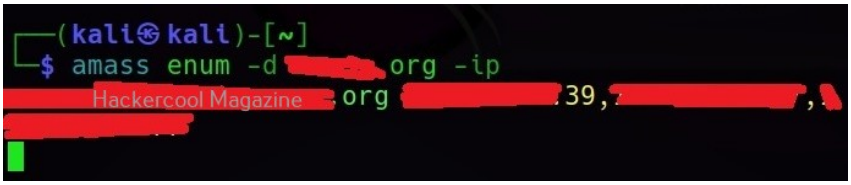
```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -max-dns-queries 100
```

+ Verbose Output: Add -v for debugging:

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d owasp.org -v
```

Adding, “-ip” option to the above command, we can also get IP addresses for the sub domains discovered.

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d google.com -ip
```



Amass queries more than 80 sources to collect information. All the sources it queries can be seen using the list flag.

Key Features and Capabilities

1. DNS Enumeration

- Brute force subdomain discovery
- Recursive DNS lookups
- Zone transfers
- Certificate transparency logs analysis
- DNS wildcard detection
- Alterations and permutations of names

2. Data Sources Integration

Amass can collect data from numerous external sources, including:

- DNS databases
- Search engines

•SSL/TLS certificate logs

•API integration with various services

•Web archives

•WHOIS records

3. Advanced Features

•Graph database support for storing and analyzing results

•Visualization capabilities for better understanding of network relationships

•Custom scripting support

•Active and passive information gathering methods

•Output in multiple formats (JSON, CSV, GraphML)

Best Practices and Optimization

1. Resource Management

Amass can be resource-intensive, especially during large scans. Consider these optimization techniques:

•Use the `-max-dns-queries` flag to limit concurrent DNS queries

•Implement appropriate timeouts using `-timeout`

•Utilize the `-df` flag for specific domain scope

2. Output Management

Properly managing and analyzing results is crucial:

```
┌──(helo㉿kali)-[~]
└─$ amass enum -d example.com -o output.txt -json output.json
```

3. Configuration File Usage

Create a config file for consistent scanning parameters:

config.yaml

resolvers:

  - 8.8.8.8

  - 8.8.4.4

scope:

  domains:

- example.com

Advanced Usage Scenarios

1. Database Integration

Amass can integrate with graph databases for complex analysis:

- kali> amass db -names -d example.com

2. Visualization

Generate visual representations of discovered networks

- kali> amass viz -d3 -d example.com

3. Custom Scripts

Implement custom scripts for specialized enumeration:

- kali> amass enum -script custom_script.ads -d example.com

Security Considerations and Legal Compliance

When using Amass, it’s crucial to:

1.Obtain proper authorization before scanning any networks

2.Respect rate limits and scanning policies

3.Be aware of local and international cybersecurity laws

4.Document all testing activities

5.Handle discovered information responsibly

Limitations and Considerations

While Amass is powerful, users should be aware of its limitations:

•Resource intensity during large scans

•Potential false positives in results

•Dependency on external data sources

•Need for proper configuration for optimal results

Integration with Other Tools

Amass works well with other security tools:

•Nmap for port scanning

•Burp Suite for web application testing

•Metasploit for exploitation

•Custom scripts through API integration