**Methodology Bug Hunting**

**Enumeration methodology**

**Subdomains enumeration**

1. Launch Amass and Sublist3r to find subdomains using a complete subdomains wordlist.
2. Merge Amass and Sublist3r outputs.
3. Launch AltDNS using the previous merge as input to discover subdomains through alterations and permutations.
4. Merge all previous outputs.
5. Launch Aquatone using the merge as input to keep only web application-based subdomains.
6. Launch Nuclei using Aquatone’s output as input to find vulnerabilities defined by multiple templates.

**Subdomains enumeration**

1. Launch [dirsearch](https://github.com/maurosoria/dirsearch) using directories wordlists to make the process recursive.
2. Launch [dirsearch](https://github.com/maurosoria/dirsearch) using the output of the previous step as input using files wordlist.
3. Launch [dirsearch](https://github.com/maurosoria/dirsearch) using technologies wordlists.

**Enumeration resources**

**Dictionaries to identify subdomains**

* [all.txt](https://gist.github.com/jhaddix/f64c97d0863a78454e44c2f7119c2a6a) (composed of dictionaries of many tools for the DNS enumeration).
* [subdomains-top1million-110000.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/DNS/subdomains-top1million-110000.txt)

**Dictionaries to identify both directories and files**

* [dicc.txt](https://gist.github.com/Lopseg/33106eb13372a72a31154e0bbab2d2b3) (contains approximately 500 of %EXT% for specified extensions).
* [content\_discovery\_all.txt](https://gist.github.com/jhaddix/b80ea67d85c13206125806f0828f4d10) (composed of dictionaries of many tools for the enumeration of files and directories).

**Dictionaries to identify only directories**

* [raft-large-directories.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/raft-large-directories.txt)
* [KitchensinkDirectories.fuzz.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/KitchensinkDirectories.fuzz.txt)
* [directory-list-2.3-big.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/directory-list-2.3-big.txt)

**Dictionaries to identify only files**

* [raft-large-files.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/raft-large-files.txt)

**Dictionaries of extensions**

* [raft-large-extensions.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/raft-large-extensions.txt)

**Dictionaries for specific technologies**

* Adobe Experience Manager: [AdobeCQ-AEM.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/AdobeCQ-AEM.txt)
* Nginx: [nginx.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/nginx.txt)
* Oracle BlueKai: [oracle.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/oracle.txt)

**Enumeration reconnaissance tools**

**Tool:** [Subdomain & test](https://github.com/badBounty/TheFantasticSecDevOps/tree/master/DAST/Scripts/subdomain%20%26%20test).

**Internal work:** it’s composed of 7 other tools (Nuclei, Amass, MassDNS, AltDNS, HTTProbe, Aquatone, Slackcat) embedded in a script. Performs a complete subdomain reconnaissance and runs nuclei on them to find vulnerabilities.

**Allows finding:** information disclosure; sensible documents; general leaks.

**Use:** get a complete subdomain list that will be used in the active reconnaissance step.

**Commands to use:**

* ./start\_enum\_n\_test.sh [domain file list]
* ./subdomain\_enum.sh <TARGET>

**Tool:** [Amass](https://github.com/OWASP/Amass).

**Internal work:** performs network mapping of attack surfaces and external asset discovery using open-source information gathering and active reconnaissance techniques.

**Allows finding:** subdomains.

**Use:** get a complete subdomains list to search for vulnerabilities.

**Output:** multiple subdomains in (\*.domain.com) format.

**Commands to use:**

* amass enum -active -d <TARGET> -o <OUTPUT>

**Tool:** [Sublist3r](https://github.com/aboul3la/Sublist3r).

**Internal work:** is a python tool designed to enumerate subdomains of websites using OSINT. It also counts with a brute force module.

**Allows finding:** subdomains.

**Use:** get a complete subdomains list to search for vulnerabilities.

**Output:** multiple subdomains in (\*.domain.com) format.

**Commands to use:**

* sublist3r -b -d <TARGET> -o <OUTPUT>

**Tool:** [AltDNS](https://github.com/infosec-au/altdns).

**Internal work:** generates a massive output of "altered" or "mutated" potential subdomains that could be present.

**Allows finding:** subdomains.

**Use:** get a complete subdomains list to search for vulnerabilities.

**Output:** multiple subdomains in (\*.domain.com) format.

**Commands to use:**

* altdns -i <SUBDOMAINS LIST> -o <DATA OUTPUT> -w <WORDS FOR PERMUTATION> -r -s <RESULT OUTPUT> -t 10

**Tool:** [Aquatone](https://github.com/michenriksen/aquatone).

**Internal work:** is a tool for visual inspection of websites across a large number of hosts and is convenient for quickly gaining an overview of HTTP-based attack surface.

**Allows finding:** different reports for use in other tools.

**Use:** use the outputs as inputs of other tools.

**Commands to use:**

* aquatone in some outputs to have final reports.

**Tool:** [Slackcat](https://github.com/bcicen/slackcat).

**Internal work:** is a simple command line utility to post snippets to Slack.

**Allows finding:** -.

**Use:** send important notifications from other tools to Slack channel

**Commands to use:**

* slackcat -c <CHANNEL> <FILE>
* slackcat -c <CHANNEL> -s <MESSAGE>

**Tool:** [dirsearch](https://github.com/maurosoria/dirsearch).

**Internal work:** it’s an advanced command-line tool designed to brute force directories and files performing an intensive scan in web servers.

**Allows finding:** directories; files.

**Use:** get a complete directory and file list to search for leaks.

**Output:** multiple URLs containing files or being directories.

**Commands to use:**

* dirsearch.py -l <SUBDOMAINS LIST> -e <EXTENSIONS LIST> -f -w <WORDLIST> --deep-recursive

**Tool:** Nuclei.

**Internal work:** it sends requests across targets based on a template leading to zero false positives.

**Allows finding:** vulnerabilities.

**Use:** it will use some of the following kinds of templates to obtain information about the target and identify a large number of possible vulnerabilities.

* **Exposures**:

Nuclei through its templates makes requests to different URLs in search of files, extensions and known directories.

For example, it looks for URLs like {{BaseURL}}/mysql.initial.sql, {{BaseURL}}/my.ppk, {{BaseURL}}/.build.sh, etc.

Some of the technologies that we can detect with this kind of templates are:

* APIs
  + OpenAPI
  + Strapi Page
  + Swagger
  + Wadl file disclosure
  + Wsdl detect
* Backups (search for common files or extensions)
  + MySQL
  + Php files
  + Php settings
  + SQL dump
  + Zip files
* Configs (search for common config files)
  + Airflow
  + Alibaba
  + Amazon
  + Ansible
  + Apache
  + Appspec Yml Disclosure
  + AWS
  + Circleci
  + Cisco network config
  + Codeigniter .env file
  + Sensitive Configuration Files Listing
  + docker
  + Nginx
  + Oracle
  + SFTP credentials
  + Ssh authorized keys
  + Ssh known hosts
* Files

Examples:

* Bower JSON file disclosure
* FileZilla
* Find GitHub page files(Gemfile / Gemfile.lock)
* Joomla: Searches for the pattern /libraries/joomla/database/ on passed URLs.
* Php User.ini Disclosure
* Putty private key
* Common shell scripts files (build.sh, jenkins.sh, run.sh, etc.)
* & more
* Tokens

Nuclei uses regex syntax to find tokens with known nomenclatures of the different technologies:

* + amazon
  + artifactory
  + bitly
  + cloudinary
  + discord
  + generic
  + google
  + mailchimp
  + Microsoft
  + Newrelic
  + Paypal
  + Picatic
  + Sendgrid
  + Slack
  + Sonarqube
  + Stripe
  + Zapier
  + zoho

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/exposures
* **Exposed-Panels**

Send requests and look in the response for exposed panels with which you can find admin pages, logins, and detect technologies used as for example some of them are:

* Aims
* Adobe
* Airflow
* Cisco
* Citrix
* Dell
* Django
* GitHub
* Jenkins
* Jira
* Kafka
* Magento
* Microsoft exchange
* mongoDB
* Nginx
* PhpMyAdmin
* Sap
* Tomcat
* VMware
* Zoho
* & more

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/exposed-panels
* **Fuzzing**
* Adminer Login Panel Fuzz
* Fuzzing headers for command injection: Requests are sent by sending payloads to the header (§header§: §payload§).
* IIS old .Net framework folder enumaration with symbol ~
* MDB database file leakage (enumeration)
* Prestashop Modules Enumeration
* WordPress Plugins Detection (Request: GET /wp-content/plugins/{{pluginSlug}}/readme.txt)
* WordPress Theme Detection
* X-Forwarded-For 403-forbidden bypass :Template to detect 403 forbidden endpoint bypass behind Nginx/Apache proxy & load balancers, based on X-Forwarded-For header.

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/fuzzing

* **DNS**

Nuclei through its templates searches if the targets belong to different DNS.

Some of the DNS technologies it can detect are:

* + Azure
  + AWS EC2 detection
  + 000domains
  + bizlnad
  + digitalocean
  + dnsmadeeasy
  + dsnsimple
  + dotster
  + easydns
  + googledomains
  + hurricane-electric
  + linode
  + mydomain
  + nsone
  + tierranet
  + yahoo
  + github.io
  + Zendesk
  + sanfor-shield
  + 360panyun
  + Baiduyun
  + Chuangyudun
  + Knownsec
  + Huaweicloud
  + Xinliuyun
  + Chinacache
  + Nscloudwaf
  + Wangsu
  + anquanbao
  + Aliyun
  + Xuanwudun
  + yundun
  + Knownsec-ns
  + Qianxin
  + wangsu
  + Baiduyunjiasue
  + Cloudflate
  + Edns

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/dns

* **Default-logins**

Nuclei through its templates tries to enter the login pages by testing default credentials for different technologies such as:

* UCMDB
* activemq
* aem
* alibaba
* ambari
* apache
* arl
* axis2
* azkaban
* chinaunicom
* dell
* dvwa
* Exacqvision
* E-mail service detector
* flir
* frps
* gitlab
* glpi
* grafana
* guacamole
* hongdian
* hortonworks
* ibm
* idemia
* iptime
* jenkins
* minio
* nagios
* nexus
* nps
* ofbiz
* oracle
* paloalto
* panabit
* rabbitmq
* ricoh
* rockmongo
* samsung
* showdoc
* solarwinds
* spectracom
* szhe
* viewpoint
* visionhub
* wifisky
* xxljob
* zabbix
* Zmand

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/default-logins

* **Misconfigurations:**

Nuclei through its templates searches for some of the most popular misconfigurations of different technologies. Some of them we can find:

* + Adobe
    - Username exposure
    - Version exposure
  + Aem
  + Airflow
    - Airflow debug trace
    - Unauthenticated Airflow Instance
  + Drupal
    - User enumeration (Ajax & Redirect)
  + gitLab
    - Public repositories
    - Public signup
    - Public snippets
    - User enumeration
  + Nginx
    - Nginx status page (/nginx\_status)
    - Nginx vhost traffic status (/status)
  + Sap
    - Directory listing
    - NetWeaver ICM page leak
  + Spingboot
    - Detect springboot autoconfig, configprops, beans, dump, Env, Health, Heapdump, mappings, metrics, trace actuator
    - Detect springboot httptrace, loggers
  + Akamai XSS
  + Alibaba Mongoshake unauth
  + Android debug DB exposed
  + Apache
    - Filename brute force
    - Page snoop disclosure
  + APCu Service information leakage
  + Artifactory anonymous deploy
  + Aspx
    - Debug mode
  + Aws
    - Object listing
    - Subdomain takeover AWS S3
  + CGI test page
  + Clockwork Dashboard Exposure
  + Cloudflare External Image Resizing Misconfiguration
  + CX Cloud Unauthenticated Upload Detect
  + D-Link Arbitrary File Read
  + Django debug detect method enabled
  + Docker
    - Registry Listing
    - Exposed API
  + Druid Monitor Unauthorized Access
  + ElasticSearch Information Disclosure
  + Exposed
    - Docker API
    - JQuery file upload
    - Kafdrop
    - Kivana
    - Service now
    - SQLite manager
  + Hadoop unauth
  + Ha Proxy statistics
  + Horde Groupware Unauthenticated
  + IIS Internal IP Disclosure Template
  + Java Melody
  + Jboss
  + Jetty show contexts enable
  + Jkstatus manager
  + Jupyter ipython unauth
  + Kubeflow dasboard unauth
  + Kubernetes
  + Laravel
  + Linkerd SSRF detection
  + Mikrotik
  + Docker
  + Office365 open redirect
  + Php
  + PhpMyAdmin
  + Private keys
  + Put method enabled
  + Salesforce
  + Sonarqube public projects
  + SSRF via oauth misconfiguration
  + Symfony debug mode
  + TCP config
  + Tomcat
  + Mongo
  + WAMP
    - Server configuration
    - Xdebug detect

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/misconfiguration

* **Takeover Detection:**

This allows an attacker to set up a page on the service that was being used and point their page to that subdomain. For example, if subdomain.example.com was pointing to a GitHub page and the user decided to delete their GitHub page, an attacker can now create a GitHub page, add a CNAME file containing subdomain.example.com, and claim subdomain.example.com.

Based on different templates, nuclei makes requests to the target and by checking the response with different parameters it can identify if the subdomain is available or not.

* Technologies that Nuclei can check to verify takeovers: acquia, aftership, agilecrm, aha, airee, anima, aws-bucket, bigcartel, bitbucket, brightcove, campaignmonitor, canny, cargo, cargocollective, ceros, fastly, feedpress, flywheel, freshdesk, freshservice, frontify, gemfury, getresponse, ghost, github, hatenablog, helpjuice, helprace, helpscout, heroku, hubspot, intercom, jazzhr, jetbrains, kinsta, landingi, launchrock, mashery, medium, netlify, ngrok, pantheon, pingdom, proposify, readme, readthedocs, shopify, simplebooklet, smartjob, smartling, smugmug, sprintful, strikingly, surveygizmo, tave, teamwork, tictail, tilda, tumblr, uberflip, uptimerobot, urge, vend, vercel, webflow, wishpond, wordpress, worksites, wufoo, zendesk.

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/takeovers

* **Technology Detection:**

Technology fingerprinting is one of the most important part of any workflow. Specific templates read the responses of different requests and look for parameters to detect and match with different types of technologies. Some of the ones that can be detected using this tool are adobe, aem, airflow, apache, aws buckets, Django, azure, oracle, tomcat, google bucket, kibana, nexus, NGinx, SAP, SQL server, WordPress and more.

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/technologies
* **Vulnerabilities**

Nuclei through its templates looks for different vulnerabilities in different technologies. For example it checks for:

* Generic
  + Basic CORS
  + Basic XSS
  + Top 15 XSS Parameter check (Searches for reflected XSS in the server response via GET-request)
  + Cache Poisoning
  + CRLF injection
  + Error based SQL injection
  + Linux LFI test
  + Windows LFI test
  + Open URL redirect detection (A user-controlled input redirect users to an external website)

In addition, check on the following technologies:

* Apache
  + Apache Flink Unauth RCE
  + Apache Solr <= 8.8.1 Arbitrary File Read
* Confluence
  + Confluence SSRF in sharelinks (Vulnerable should be Confluence versions released from 2016 November and older)
* GitLab
  + User enumeration
  + Information disclosure via open API
* IBM
  + Eclipse Help System RXSS Vulnerability
  + Directory traversal vulnerability on IBM InfoPrint 4247-Z03 Impact Matrix Printer.
* Jenkins
  + Jenkins panel async-people
  + Jenkins RCE due to accessible script functionality
  + Detect Jenkins in Debug Mode with Stack Traces Enabled
  + Unauthenticated Jenkins Dashboard
* Jira
  + Check for Jira Service Desk Signup
  + Check for unauthenticated admin projects
  + Check for unauthenticated dashboards (If public sharing is ON it allows users to share dashboards and filters with all users including those that are not logged in. Those dashboard and filters could reveal potentially sensitive information)
  + Check for Unauthenticated gadgets
  + Check for Unauthenticated project categories
  + Check for Unauthenticated projects
  + Check for Unauthenticated Resolutions
  + Check for Unauthenticated user picker
* Jolokia Information disclosure
* Check for CVE-2010-1870 (Struts based OGNL remote code execution in ListSERV Maestro before and including version 9.0-8)
* Magento
  + Check for exposed Magento 2 API
  + Check for Magento Cacheleak
  + Check for Magento unprotected development files
* Moodle
  + Local File Inclusion
  + XSS
* Oracle
  + Oracle EBS Bispgraph file access
  + XSS
* osCommerce
  + Check for Remote code execution (Exploiting the install.php finish process by injecting php payload into the db\_database parameter & read the system command output from configure.php)
* Samsung
  + Wlan AP LFI
  + Wlan AP RCE
  + Wlan AP XSS
* Springboot
  + Spring Boot Actuators (Jolokia) XXE
  + Spring Boot H2 Database RCE
* thinkCMF
  + Arbitrary code execution
  + LFI
  + RCE
* thinkPHP
  + RCE
  + Information disclosure
* VMware
  + VMware vCenter LFI for Linux appliances
  + Vmware vCenter Unauthenticated Arbitrary File Read
* Wordpress
  + Searches for sensitive directories present in the ALFA\_DATA
  + WordPress Plugin Media Gallery Pro Listing (Searches for sensitive directories present in the WordPress-plugins plugin)
  + SMTP WP Plugin Directory Listing enabled
  + Total Cache SSRF
  + Plugin listing
  + Backup listing
  + Directory listing
  + User enumeration
  + Woocommerce unauthenticated SQL injection
  + LFI
  + XSS
  + Log exposure
  + Wordfence information disclosure
  + Check for license file
  + Wpmudev Dashboard Pub Key
* & other vulnerabilities

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/vulnerabilities

* **CVE Identification**:

By sending different requests, and reading different parameters of the response Nuclei can identify if the target may be vulnerable with the following CVE's

* **2005**: CVE-2005-2428, CVE-2005-4385.
* **2006**: CVE-2006-1681
* **2007**: CVE-2007-0885, CVE-2007-4556
* **2008**: CVE-2008-2398, CVE-2008-2650, CVE-2008-4764, CVE-2008-6668
* **2009**: CVE-2009-0545, CVE-2009-0932, CVE-2009-1151, CVE-2009-1558, CVE-2009-1872, CVE-2009-4223, CVE-2009-5114
* **2010**: CVE-2010-0944, CVE-2010-1979, CVE-2010-1983, CVE-2010-2259, CVE-2010-2307, CVE-2010-2682, CVE-2010-2861, CVE-2010-4231, CVE-2010-4617
* **2011**: CVE-2011-0049, CVE-2011-1669, CVE-2011-3315, CVE-2011-4336, CVE-2011-4618, CVE-2011-4624,CVE-2011-4804, CVE-2011-4926, CVE-2011-5106, CVE-2011-5107, CVE-2011-5179, CVE-2011-5181, CVE-2011-5265
* **2012**: CVE-2012-0392, CVE-2012-0901, CVE-2012-0991, CVE-2012-1823, CVE-2012-1835, CVE-2012-2371, CVE-2012-3153, CVE-2012-4242, CVE-2012-4253, CVE-2012-4273, CVE-2012-4768, CVE-2012-4878, CVE-2012-4889, CVE-2012-5913
* **2013**: CVE-2013-1965, CVE-2013-2248, CVE-2013-2251, CVE-2013-2287, CVE-2013-3526, CVE-2013-3827, CVE-2013-4117, CVE-2013-4625, CVE-2013-5528, CVE-2013-5979, CVE-2013-7240
* **2014**: CVE-2014-2321, CVE-2014-2323, CVE-2014-2383, CVE-2014-2962, CVE-2014-3120, CVE-2014-3704, CVE-2014-3744, CVE-2014-4210, CVE-2014-4513, CVE-2014-4535, CVE-2014-4536, CVE-2014-4940, CVE-2014-5368, CVE-2014-6271, CVE-2014-6308, CVE-2014-8799, CVE-2014-9094
* **2015**: CVE-2015-1000012, CVE-2015-1427, CVE-2015-1880, CVE-2015-2080, CVE-2015-2807, CVE-2015-3306, CVE-2015-3337, CVE-2015-3648, CVE-2015-5688, CVE-2015-6477, CVE-2015-6544, CVE-2015-7297, CVE-2015-7823, CVE-2015-8349, CVE-2015-8399, CVE-2015-8813, CVE-2015-9414, CVE-2015-9480
* **2016**: CVE-2016-0957, CVE-2016-1000126, CVE-2016-1000127, CVE-2016-1000128, CVE-2016-1000129, CVE-2016-1000130, CVE-2016-1000131, CVE-2016-1000132, CVE-2016-1000133, CVE-2016-1000134, CVE-2016-1000135, CVE-2016-1000137, CVE-2016-1000138, CVE-2016-1000139, CVE-2016-1000140, CVE-2016-1000146, CVE-2016-1000148, CVE-2016-1000149, CVE-2016-1000152, CVE-2016-1000153, CVE-2016-1000154, CVE-2016-1000155, CVE-2016-10033, CVE-2016-10956, CVE-2016-10960, CVE-2016-10993, CVE-2016-2004, CVE-2016-2389, CVE-2016-3081, CVE-2016-5649, CVE-2016-7552, CVE-2016-7981
* **2017**: CVE-2017-1000028, CVE-2017-1000170, CVE-2017-1000486, CVE-2017-10075, CVE-2017-10271, CVE-2017-11444, CVE-2017-12149, CVE-2017-12542, CVE-2017-12611, CVE-2017-12615, CVE-2017-12629, CVE-2017-12635, CVE-2017-12637, CVE-2017-12794, CVE-2017-14535, CVE-2017-14537, CVE-2017-14651, CVE-2017-14849, CVE-2017-15647, CVE-2017-15715, CVE-2017-15944, CVE-2017-16806, CVE-2017-16877, CVE-2017-17043, CVE-2017-17059, CVE-2017-17451, CVE-2017-17562, CVE-2017-18024, CVE-2017-18536, CVE-2017-3506, CVE-2017-3528, CVE-2017-3881, CVE-2017-5487, CVE-2017-5521, CVE-2017-5638, CVE-2017-6090, CVE-2017-7269, CVE-2017-7391, CVE-2017-7615, CVE-2017-7921, CVE-2017-8917, CVE-2017-9140, CVE-2017-9288, CVE-2017-9506, CVE-2017-9791, CVE-2017-9805, CVE-2017-9822, CVE-2017-9841
* **2018**: CVE-2018-0296, CVE-2018-1000129, CVE-2018-1000130, CVE-2018-1000533, CVE-2018-1000600, CVE-2018-1000861, CVE-2018-10095, CVE-2018-10141, CVE-2018-10818, CVE-2018-10822, CVE-2018-11409, CVE-2018-11709, CVE-2018-11759, CVE-2018-11776, CVE-2018-11784, CVE-2018-12031, CVE-2018-1207, CVE-2018-1247, CVE-2018-12613, CVE-2018-12634, CVE-2018-1271, CVE-2018-1273, CVE-2018-1335, CVE-2018-13379, CVE-2018-13380, CVE-2018-14013, CVE-2018-14574, CVE-2018-14728, CVE-2018-15517, CVE-2018-15745, CVE-2018-16059, CVE-2018-16167, CVE-2018-16283, CVE-2018-16288, CVE-2018-16299, CVE-2018-16341, CVE-2018-16668, CVE-2018-16670, CVE-2018-16671, CVE-2018-16763, CVE-2018-16836, CVE-2018-17246, CVE-2018-17254, CVE-2018-17431, CVE-2018-18069, CVE-2018-18775, CVE-2018-18777, CVE-2018-18778, CVE-2018-19386, CVE-2018-19439, CVE-2018-19458, CVE-2018-20462, CVE-2018-20470, CVE-2018-20824, CVE-2018-20985, CVE-2018-2392, CVE-2018-2628, CVE-2018-2791, CVE-2018-2893, CVE-2018-2894, CVE-2018-3167, CVE-2018-3714, CVE-2018-3760, CVE-2018-3810, CVE-2018-5230, CVE-2018-5233, CVE-2018-5316, CVE-2018-6910, CVE-2018-7251, CVE-2018-7422, CVE-2018-7490, CVE-2018-7600, CVE-2018-7700, CVE-2018-8006, CVE-2018-8033, CVE-2018-8715, CVE-2018-8770, CVE-2018-9118, CVE-2018-9995
* **2019**: CVE-2019-0193, CVE-2019-0221, CVE-2019-0230, CVE-2019-10068, CVE-2019-10092, CVE-2019-1010287, CVE-2019-10475, CVE-2019-11248, CVE-2019-11510, CVE-2019-11580, CVE-2019-11581, CVE-2019-11869, CVE-2019-12276, CVE-2019-12314, CVE-2019-12461, CVE-2019-12593, CVE-2019-12616, CVE-2019-12725, CVE-2019-13101, CVE-2019-13462, CVE-2019-14205, CVE-2019-14223, CVE-2019-14312, CVE-2019-14322, CVE-2019-14470, CVE-2019-14696, CVE-2019-14974, CVE-2019-15043, CVE-2019-15107, CVE-2019-15713, CVE-2019-15858, CVE-2019-15859, CVE-2019-15889, CVE-2019-16097, CVE-2019-16278, CVE-2019-16332, CVE-2019-16525, CVE-2019-1653, CVE-2019-16662, CVE-2019-16759, CVE-2019-16920, CVE-2019-17270, CVE-2019-17382, CVE-2019-17506, CVE-2019-17538, CVE-2019-17558, CVE-2019-18393, CVE-2019-18394, CVE-2019-19134, CVE-2019-19368, CVE-2019-19781, CVE-2019-19908, CVE-2019-19985, CVE-2019-20085, CVE-2019-20141, CVE-2019-2588, CVE-2019-2616, CVE-2019-2725, CVE-2019-2767, CVE-2019-3396, CVE-2019-3401, CVE-2019-3402, CVE-2019-3403, CVE-2019-3799, CVE-2019-5127, CVE-2019-5418, CVE-2019-6112, CVE-2019-6340, CVE-2019-6715, CVE-2019-7219, CVE-2019-7238, CVE-2019-7254, CVE-2019-7256, CVE-2019-7481, CVE-2019-7609, CVE-2019-8442, CVE-2019-8446, CVE-2019-8449, CVE-2019-8451, CVE-2019-8903, CVE-2019-8982, CVE-2019-9041, CVE-2019-9618, CVE-2019-9670, CVE-2019-9733, CVE-2019-9955, CVE-2019-9978
* **2020**: CVE-2019-9618, CVE-2020-0618, CVE-2020-10148, CVE-2020-10546, CVE-2020-10547, CVE-2020-10548, CVE-2020-10549, CVE-2020-11034, CVE-2020-11110, CVE-2020-11455, CVE-2020-11710, CVE-2020-11738, CVE-2020-11853, CVE-2020-11854, CVE-2020-11930, CVE-2020-11978, CVE-2020-11991, CVE-2020-12054, CVE-2020-12116, CVE-2020-12720, CVE-2020-13117, CVE-2020-13167, CVE-2020-13483, CVE-2020-13700, CVE-2020-13927, CVE-2020-13937, CVE-2020-13942, CVE-2020-14092, CVE-2020-14179, CVE-2020-14181, CVE-2020-14413, CVE-2020-14864, CVE-2020-14882, CVE-2020-14883, CVE-2020-15129, CVE-2020-15148, CVE-2020-15227, CVE-2020-15500, CVE-2020-15505, CVE-2020-15568, CVE-2020-15920, CVE-2020-16139, CVE-2020-16846, CVE-2020-16952, CVE-2020-17362, CVE-2020-17453, CVE-2020-17496, CVE-2020-17505, CVE-2020-17506, CVE-2020-17518, CVE-2020-17519, CVE-2020-17530, CVE-2020-1938, CVE-2020-1943, CVE-2020-19625, CVE-2020-2036, CVE-2020-2096, CVE-2020-21224, CVE-2020-2140, CVE-2020-22840, CVE-2020-23517, CVE-2020-23972, CVE-2020-24148, CVE-2020-24186, CVE-2020-24223, CVE-2020-24312, CVE-2020-24550, CVE-2020-24571, CVE-2020-24579, CVE-2020-24949, CVE-2020-25078, CVE-2020-25213, CVE-2020-25495, CVE-2020-25506, CVE-2020-2551, CVE-2020-25540, CVE-2020-26073, CVE-2020-26153, CVE-2020-26214, CVE-2020-26919, CVE-2020-26948, CVE-2020-27361, CVE-2020-27735, CVE-2020-27866, CVE-2020-27982, CVE-2020-27986, CVE-2020-28188, CVE-2020-28208, CVE-2020-28871, CVE-2020-29164, CVE-2020-29227, CVE-2020-29395, CVE-2020-3187, CVE-2020-3452, CVE-2020-35338, CVE-2020-35476, CVE-2020-35489, CVE-2020-35580, CVE-2020-35598, CVE-2020-35713, CVE-2020-35729, CVE-2020-35736, CVE-2020-35774, CVE-2020-3580, CVE-2020-35846, CVE-2020-35847, CVE-2020-35848, CVE-2020-35951, CVE-2020-36112, CVE-2020-36289, CVE-2020-4463, CVE-2020-5284, CVE-2020-5307, CVE-2020-5405, CVE-2020-5410, CVE-2020-5412, CVE-2020-5776, CVE-2020-5777, CVE-2020-5847, CVE-2020-5902, CVE-2020-6171, CVE-2020-6207, CVE-2020-6287, CVE-2020-6308, CVE-2020-6637, CVE-2020-7209, CVE-2020-7247, CVE-2020-7318, CVE-2020-7796, CVE-2020-7961, CVE-2020-8091, CVE-2020-8115, CVE-2020-8163, CVE-2020-8191, CVE-2020-8193, CVE-2020-8194, CVE-2020-8209, CVE-2020-8512, CVE-2020-8515, CVE-2020-8771, CVE-2020-8813, CVE-2020-8982, CVE-2020-9036, CVE-2020-9047, CVE-2020-9054, CVE-2020-9315, CVE-2020-9344, CVE-2020-9376, CVE-2020-9402, CVE-2020-9425, CVE-2020-9483, CVE-2020-9484, CVE-2020-9490, CVE-2020-9496, CVE-2020-9757
* **2021**: CVE-2021-1497, CVE-2021-20090, CVE-2021-20091, CVE-2021-20092, CVE-2021-21234, CVE-2021-21307, CVE-2021-21315, CVE-2021-21389, CVE-2021-21402, CVE-2021-21479, CVE-2021-21801, CVE-2021-21802, CVE-2021-21803, CVE-2021-21816, CVE-2021-21972, CVE-2021-21975, CVE-2021-21978, CVE-2021-21985, CVE-2021-22122, CVE-2021-22214, CVE-2021-22873, CVE-2021-22986, CVE-2021-23241, CVE-2021-24146, CVE-2021-24176, CVE-2021-24210, CVE-2021-24235, CVE-2021-24237, CVE-2021-24285, CVE-2021-24291, CVE-2021-24298, CVE-2021-24316, CVE-2021-24320, CVE-2021-24335, CVE-2021-24340, CVE-2021-24387, CVE-2021-24389, CVE-2021-24406, CVE-2021-24472, CVE-2021-24495, CVE-2021-24498, CVE-2021-25281, CVE-2021-25646, CVE-2021-26295, CVE-2021-26475, CVE-2021-26710, CVE-2021-26722, CVE-2021-26723, CVE-2021-26812, CVE-2021-26855, CVE-2021-27132, CVE-2021-27330, CVE-2021-27561, CVE-2021-27651, CVE-2021-27850, CVE-2021-27905, CVE-2021-28073, CVE-2021-28149, CVE-2021-28150, CVE-2021-28151, CVE-2021-28164, CVE-2021-28169, CVE-2021-28854, CVE-2021-28937, CVE-2021-29156, CVE-2021-29203, CVE-2021-29441, CVE-2021-29442, CVE-2021-29484, CVE-2021-29622, CVE-2021-30151, CVE-2021-3019, CVE-2021-30461, CVE-2021-30497, CVE-2021-31249, CVE-2021-31250, CVE-2021-3129, CVE-2021-31537, CVE-2021-31581, CVE-2021-31755, CVE-2021-3223, CVE-2021-32305, CVE-2021-32820, CVE-2021-3297, CVE-2021-33221, CVE-2021-33544, CVE-2021-33564, CVE-2021-3374, CVE-2021-3377, CVE-2021-3378, CVE-2021-33904, CVE-2021-34429, CVE-2021-34473, CVE-2021-34621, CVE-2021-35336, CVE-2021-35464, CVE-2021-36380, CVE-2021-37216

For more information, please refer to the following link to the official Nuclei template GitHub repository:

* https://github.com/projectdiscovery/nuclei-templates/tree/master/cves
* **Workflows**

Nuclei uses different workflows for the detection of Technologies & vulnerabilities. This is performed using templates such as the previously mentioned.

Some of the workflows available are:

* Aem
* Airflow
* Artica web proxy
* Azkaban
* Bigip
* Cacti
* Cisco
* Cockpit
* Dell
* Gitlab
* Grafana
* Harbor
* Jellyfin
* Jira
* Liferay
* Lotus
* Lucee
* Magento
* Magmi
* Micro focus
* Mida eframework
* Netsweeper
* Phpmyadmin
* Rabbitmq
* Ruijie
* Samsung
* Sap netweaver
* Solarwinds orion
* Springboot
* Thinkcmf
* Thinkphp
* Vbulletin
* Weblogic
* Wordpress
* Worksite takeover

**Scanning methodology**

*TODO*

**Scanning resources**

**Passive reconnaissance tools**

**Tool:** [Goohak](https://github.com/1N3/Goohak).

**Internal work:** it uses advanced Google search to find security holes in the configuration or source code of a website.

**Allows finding:** information disclosure; sensitive documents; emails; users and passwords.

**Use:** multiple.

**Commands to use:**

* ./goohak domain.com

**Output:** multiple URLs containing sensitive information or documents.

**Tool:** [Censys](https://search.censys.io/).

**Internal work:** it performs an internet wide scanning by searching how devices or websites are configured and deployed.

**Allows finding:** unknown internet assets; certificates; open ports and services.

**Use:** check security of those assets; check SSL certificates.

**Commands to use:** works through an API.

**Output:** multiple hosts connected to the target with some information about them, such as ip, open ports and SSL certificates.

**Tool:** [Hunter](https://hunter.io/).

**Internal work:** it performs an internet wide scanning for finding professional emails associated to a domain/organization.

**Allows finding:** emails.

**Use:** brute force.

**Commands to use:** works through an API.

**Output:** list of emails with the specified domain.

**Tool:** [CloudFail](https://github.com/m0rtem/CloudFail).

**Internal work:** gather enough information about a target protected by Cloudflare in the hopes of discovering the real server behind it.

**Allows finding:** ip addresses.

**Use:** scan the IPs found.

**Commands to use:**

* python3 cloudfail.py --target <TARGET>

**Output:** list of hosts found behind CloudFlare.

**Tool:** [Shodan](https://www.shodan.io/).

**Internal work:** search engine for internet connected devices of an organization.

**Allows finding:** devices.

**Use:** check for insecure devices connected to the target network.

**Commands to use:** works through an API.

**Output:** multiple hosts connected to the target with some information about them, such as IPs, open ports and SSL certificates.

**Tool:** [Wappalyzer](https://www.wappalyzer.com/).

**Internal work:** reveals all the technologies stack of any website.

**Allows finding:** technologies.

**Use:** check all domains to get technologies associated and perform a vulnerability scan for each.

**Commands to use:** works through an API.

**Output:** list of different technologies used by the target

**Active reconnaissance tools**

**Tool:** [Nikto](https://github.com/sullo/nikto).

**Internal work:** performs comprehensive tests against web servers for multiple items.

**Allows finding:** files; programs; versions; configurations.

**Use:** check for files or programs to be insecure; check for outdated software by knowing the versions; check for software misconfigurations.

**Commands to use:**

* ./nikto.pl -h <TARGET>

**Output:** list of files or programs with their respective versions.

**Tool:** [Sn1per](https://github.com/1N3/Sn1per).

**Internal work:** performs an intensive scan to discover the attack surface.

**Allows finding:** IPs; domain names; HTTP headers; open ports; WAF; vulnerabilities linked to latest CVEs.

**Use:** check for insecure IPs and domain names; check for insecure HTTP headers resulting in XSS or CSRF attacks; check if any technology used is linked to a CVE.

**Commands to use:**

* sniper -t <TARGET> -m webscan

**Output:** list of different possible findings

**Tool:** [Datasploit](https://github.com/DataSploit/datasploit).

**Internal work:** performs OSINT on a domain and finds out information from different sources.

**Allows finding:** credentials; API-keys; tokens; subdomains; domain history; legacy portals.

**Use:** check for privilege credentials; check for insecure tokens and API-keys.

**Commands to use:**

* python3 domainOsint.py -d <TARGET>

**Output:** list of possible API-keys and tokens related to the target.

**Tool:** [Nmap](https://nmap.org/download.html) .

**Internal work:** performs an intensive scan in order to gather information of multiple nature.

**Allows finding:** versions; vulnerabilities; open ports; services.

**Use:** check nmap web-based scripts to find all above.

**Commands to use:**

* nmap -Pn -sSV -p <PORT> --script=banner,vuln,vulscan/vulscan.nse,http-enum,http-webdav-scan,http-backup-finder,http-trace <TARGET>

**Output:** information about vulnerabilities found by each script.

**Tool:** [getJS](https://github.com/003random/getJS).

**Internal work:** extract all the JavaScript files from a set of given URLs.

**Allows finding:** JavaScript files.

**Use:** check all domains to get JS files.

**Commands to use:**

* getJS --url <TARGET> --complete --output <OUTPUT> --resolve

**Output:** list of URLs that lead to JavaScript files.

**Tool:** [Photon](https://github.com/s0md3v/Photon).

**Internal work:** performs crawling to find data from a given domain.

**Allows finding:** files; URLs with parameters; auth keys; API keys; hashes; endpoints.

**Use:** check if parameters have a vulnerability associated; try with secret keys and hashes given by the tool.

**Commands to use:**

* python photon.py -u <TARGET> -o <OUTPUT>

**Output:** multiple files saved in a directory because of the crawling performed.

**Tool:** [h2t](https://github.com/gildasio/h2t).

**Internal work:** check the website headers to find misconfigurations.

**Allows finding:** misconfigurations.

**Use:** use findings to perform specific attacks such as XSS.

**Commands to use:**

* ./h2t.py scan -h <TARGET> -o <OUTPUT>

**Output:** list of headers with their state (good or bad).

**Tool:** [gowitness](https://github.com/sensepost/gowitness).

**Internal work:** uses Chrome Headless to generate screenshots of web interfaces using the command line, with a handy report viewer to process results.

**Allows finding:** -

**Use:** use the screenshots as proof of the vulnerabilities found by the tool.

**Commands to use:**

* gowitness single <TARGET>

**Output:** a screenshot saved in a directory.

**Tool:** [See-SURF](https://github.com/In3tinct/See-SURF).

**Internal work:** it performs an intensive scan to find potential SSRF parameters in a web application.

**Allows finding:** SSRF parameters.

**Use:** check for vulnerabilities in the SSRF parameters found.

**Commands to use:**

* python3 see-surf.py -H <TARGET>

**Output:** list of potential vulnerabilities containing the method, parameter and url implicated.

**Tool:** [XSStrike](https://github.com/s0md3v/XSStrike).

**Internal work:** analyses the response with multiple parsers and then crafts payloads that are guaranteed to work by context analysis integrated with a fuzzing engine.

**Allows finding:** parameters; DOM XSS vulnerabilities.

**Use:** check if parameters have a vulnerability associated.

**Commands to use:**

* python3 xsstrike.py -u <TARGET> -seeds <SUBDOMAINS FILE>

**Output:** list of objects and webpages vulnerable to XSS based on the subdomains found.

**Tool:** [FockCache](https://github.com/tismayil/fockcache).

**Internal work:** tries to make cache poisoning by trying X-Forwarded-Host and X-Forwarded-Scheme headers on web pages.

**Allows finding:** cache vulnerabilities.

**Use:** check if cache can be poisoned.

**Commands to use:**

* ./FockCache --hostname <TARGET>

**Output:** list of tests performed by the tool without adding parameters and adding parameters.

**Tool:** [AEM Hacker](https://github.com/0ang3el/aem-hacker).

**Internal work:** identifies vulnerable Adobe Experience Manager (AEM) webapps.

**Allows finding:** vulnerabilities.

**Use:** check if AEM version and configuration has any vulnerability associated.

**Commands to use:**

* python3 aem\_discoverer.py -h <SUBDOMAINS FILE>
* python3 aem\_hacker.py -u <TARGET IN DISCOVERED> --host <IP FOR BACK CONNECTIONS>

**Output:** list of vulnerabilities associated with the current AEM software running on the website and its configuration.

**Tool:** [Subzy](https://github.com/LukaSikic/subzy).

**Internal work:** subdomain takeover tool which works based on matching response fingerprints from a list.

**Allows finding:** HTTP vulnerabilities.

**Use:** check HTTP vulnerabilities.

**Commands to use:**

* subzy –targets <SUBDOMAINS FILE> --hide\_fails

**Output:** list of vulnerabilities and their location based on matching response fingerprints from a specific list.

**Tool:** [Retire.js](https://github.com/retirejs/retire.js/)

**Internal work:** performs a scan to try detecting the use of JS-library versions with known vulnerabilities.

**Allows finding:** JS-library vulnerabilities.

**Use:** check JS-library vulnerabilities.

**Commands to use:**

* retire --jspath <PATH> --outputformat <FORMAT> --outputpath <PATH>

**Output:** list of JavaScript files and their versions.

**Tool:** [DumpsterDriver](https://github.com/securing/DumpsterDiver).

**Internal work:** analyzes big volumes of data in search of hardcoded secrets like keys.

**Allows finding:** leaks.

**Use:** use those leaks to perform different attacks on the website.

**Commands to use:**

* python3 DumpsterDiver.py -p <PATH> -o <OUTFILE>

**Output:** list of potential passwords leaked.

**Tool:** [wafw00f](https://github.com/EnableSecurity/wafw00f).

**Internal work:** it sends a normal HTTP request and analyses the response.

**Allows finding:** WAF solutions.

**Use:** know and identify if there is any known WAF running.

**Commands to use:**

* wafw00f <TARGET>

**Output:** tells if there is any WAF running on the website.

**Tool:** [CRLF-Injection-Scanner](https://github.com/random-robbie/CRLF-Injection-Scanner)

**Internal work:** command line tool for testing CRLF injection on list of domains.

**Allows finding:** CRLF vulnerabilities.

**Use:** check if there are any CRLF vulnerabilities on the complete website.

**Commands to use:**

* crlf\_scan.py -i <inputfile> -o <outputfile>

**Output:** list of vulnerabilities found.