



Sri Lanka Institute of Information Technology

VULNERABILITY ASSESSMENT - WEB AUDIT

<https://www.grammarly.com/>

Individual Assignment

IE2062 – Web Security

Submitted by:

Student Registration Number	Student Name
IT19976686	Jayasinghe D.A.

Date of submission: 31/05/2021

Table of Contents

Acknowledgement.....	5
Assesment Objectives.....	5
Application Creditionals and URL.....	5
What is Website Audit?	6
Domain Selection.....	6
Introduction.....	7
1. Information Gathering	9
I. Policy.....	9
II. Scope.....	10
2. Reconnaissance.....	12
I. Target	
Validation.....	12
i. Whois.....	12
II. Finding Sub-domains.....	13
i. Sublist3r.....	13
ii. Crt.sh.....	16
iii. Subfinder.....	16
iv. Amass.....	19
3. Discovery and Scanning.....	21
I. Nmap.....	21
II. Wafw00f.....	23
i. Solution:	25
4. Vulnerability Assessment.....	26
I. Nikto.....	26
i. X-Content-Type Options.....	31
a. Solution.....	31

ii.	X-XSS-Protection.....	31
a.	Solution.....	31
iii.	X-Content-Type Options.....	31
a.	Solution.....	32
II.	Uniscan.....	32
III.	Nessus.....	41
IV.	OWASP ZAP.....	49
i.	3 medium priority vulnerabilities:	52
1.	CSP Scanner: Wildcard Directive (1234)	52
2.	Referer Exposes Session ID (5)	53
3.	Session ID in URL Rewrite (5)	54
ii.	Other low and informational priority vulnerabilities:	55
1.	Absense of Anti-CRSF tokens (2688)	55
2.	Informational Disclosure – Sensitive Information in URL (491)	55
V.	Netsparker Proffessional.....	56
i.	Identified Vulnerabilities.....	63
1.	Vulnerability: Out-of-date Version (Nginx)	63
2.	Vulnerability: [Possible] Cross-site Scripting.....	65
3.	Vulnerability: HTTP Strict Security (HSTS) Errors and Warnings.....	67
4.	Vulnerability: Version Disclosure (Nginx)	68
ii.	Confirmed Vulnerabilities.....	69
1.	Vulnerability: Weak Cipher Enabled.....	69
2.	Vulnerability: Cookie Not Marked as HttpOnly.....	71
3.	Vulnerability: Cookie Not Marked as Secure.....	72
4.	Vulnerability: Insecure Frame (External)	73
5.	Vulnerability: Missing X-Frame-Options Heade.....	74
6.	Vulnerability: Robots.txt Detected.....	75
a.	Sitemaps:	77

VI. Burp Suite Professional.....	79
i. Crawl and Audit:	80
a. Crawl:	80
b. Audit:	80
ii. Identified Vulnerabilities.....	88
1. Vulnerability: Strict transport security not enforced...	88
2. Vulnerability: TLS cookie without secure flag set.....	90
3. Vulnerability: Cookie without HttpOnly flag set.....	91
Conclusion.....	92
References.....	93
Explanation Video URL.....	93

Acknowledgement

I would like to thank Ms. Chethana Liyanapathirana, the lecturer in charge of the Web Security module, for her insightful feedback and advice, which was crucial to the start of this web audit.

I would like to express my gratitude to Dr. Lakmal Rupasinghe, Ms. Lanessha Rukgahakotuwa, and Ms. Chathu Udagedara for their assistance and advice during this web audit.

Assessment Objectives

We were given the option of performing a web audit on any domain we like. The tests started on the 20th of April and ended on the 30th of May. This paper demonstrates how to perform a primary web audit that focuses on the topics of; information gathering, reconnaissance, discovering and scanning, and the vulnerability assessment.

Application Credentials and URL

The audit was done on the following URLs.

Target domain:

- ✓ <https://www.grammarly.com/>

Target sub-domains:

- ✓ <https://account.grammarly.com/>
- ✓ <https://app.grammarly.com/>

What is Website Audit?

A website audit is a process of assessing a web system which covers websites as well as web applications for vulnerabilities and loopholes. There are many explanations for a website audit, but the most important ones are usually SEO (Search Engine Optimization) and content marketing.

An SEO-based website audit identifies weak points in a website's SEO score and leads to a better understanding of its SEO status. The content audit is used to assess engagement and evaluate improvements to the content strategy in order to increase the website's performance. A systematic cyber protection audit examines all information infrastructure properties for safety protocols, security measures, and future risks. SQL Injections and Cross Site Scripting (XSS) are examples of cyber-attack tactics that pose a significant danger to data security and can result in the full compromise of computer networks. The biggest vulnerability risks associated with web-based applications are caused by web vulnerabilities. Web vulnerability audits involve any of a website's secret or undisclosed bugs, as well as the security infrastructure. Vulnerability scanners are the most popular method for evaluating a website's security.

Domain selection

Using “<https://www.bugcrowd.com>” website, I have selected this “Grammarly” for my web audit as shown in bellow.

The screenshot shows the Bugcrowd homepage. At the top, there is a navigation bar with links for "Researcher Portal" and "Customer Portal". Below the navigation bar, the Bugcrowd logo is displayed, followed by a subtext "A Crowdsourced Security Company". To the right of the logo are several menu items: "Why Bugcrowd", "Products", "Solutions", "Researchers", "Programs", "Resources", "Company", and a prominent orange "Get Started" button. Below the menu, there is a search bar with the placeholder text "Search Bugcrowd". Underneath the search bar, there is a section titled "Audit Selection" with the subtext "Select the domains you want to audit". A table lists three domains: "Grammarly" (selected), "Facebook" (selected), and "Twitter" (not selected). The "Grammarly" row has a green checkmark icon next to it. The "Facebook" and "Twitter" rows have orange checkmark icons next to them. The "Grammarly" row also contains a "Edit" link and a "Delete" link.

Introduction

You have already heard of Grammarly if you do some sort of online writing. It is the most common spelling and grammar checker on the internet, with over 10 million regular active users. Grammarly analyzes and scans a piece of text you have composed for mistakes. The free version adds the fundamentals (grammar, punctuation, and spelling), while the premium version goes a little further by providing stylistic and best-writing-practice recommendations. Grammarly will help you polish your tweets, social media articles, journals, and formal documents, no matter what kind of writing you are doing – professional, informal, or hobby writing.

To that end, I believe the following people would learn the most from Grammarly:

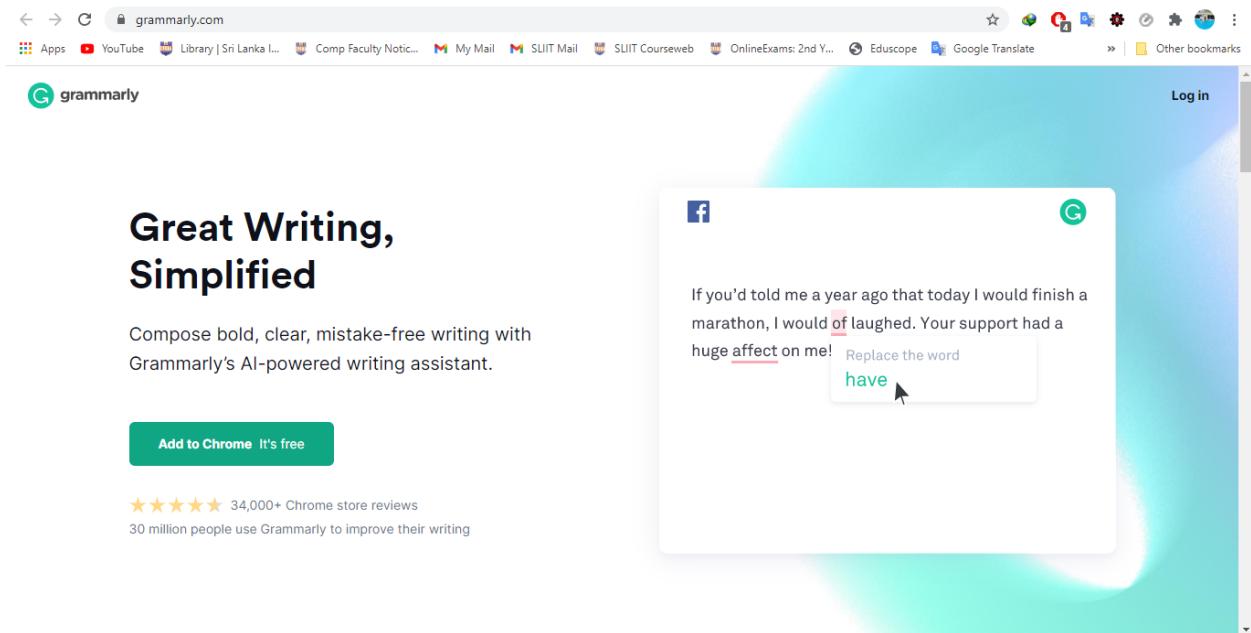
-  Students
-  Bloggers
-  Copywriters and content marketers
-  Professionals and business authors who need to create an accurate article, presentation, or email.

The different ways to use Grammarly:

Since Grammarly is an internet-based tool, in order to use it you need to be connected to the internet. Also, there are various ways to check your work through it.

- 1. The Grammarly web tool**
- 2. Desktop app**
- 3. Microsoft Word**
- 4. Browser extension**
- 5. On your phone**

The below figure shows the interface of the “grammarly.com”.



1. Information Gathering

I. Policy

As shown in the figure, there are some policies that defines some rules and conditions which are regarding for web auditors and the organization.

The screenshot shows a web page from hackerone.com. At the top right, there is a link to 'Last updated on January 20, 2021' and a 'View changes' button. Below this, there are three navigation links: 'SOLUTIONS' with a dropdown arrow, 'PRODUCTS' with a dropdown arrow, and 'WHY HACKERONE' with a dropdown arrow. The main content area has a light gray background. On the left, there is a vertical sidebar with a dark gray header containing the 'hackerone' logo. The main content area has a white header with the title 'Policy'. Below the header, there is a paragraph of text: 'Security and privacy are core concepts at Grammarly. Grammarly looks forward to working with the security community to find security vulnerabilities in order to keep our business and customers safe.' Underneath this paragraph, there are two sections: 'Rules for us' and 'Rules for you', each containing a bulleted list of guidelines.

Last updated on January 20, 2021 | View changes

SOLUTIONS ▾ PRODUCTS ▾ WHY HACKERONE ▾

Policy

Security and privacy are core concepts at Grammarly. Grammarly looks forward to working with the security community to find security vulnerabilities in order to keep our business and customers safe.

Rules for us

- We offer a **safe harbor (defined below)** to **all activities** that are consistent with this policy.
- We respect the time and effort of our researchers.
- We will do our best to keep you informed about our progress throughout the process.
- We will try to award a bounty for a successfully validated report in 3 days after the triage.
- We will not respond to threats or negotiate under duress.

Rules for you

- Be an ethical hacker.
- Respect privacy: Only interact with test Grammarly accounts you own.
- Avoid testing that would result in privacy violations, destruction of data, or interruption or degradation of our service.
- Contact us immediately if you inadvertently encounter user data. Do not view, alter, save, store, transfer, or otherwise access the data, and immediately purge any local information upon reporting the vulnerability to Grammarly.
- Follow [disclosure guidelines](#) during the triage and after the successful remediation of the vulnerability.

II. Scope

The scope is listed with few sub domains and the main domain as shown below.

Scopes			
In Scope			
Domain	www.grammarly.com	Critical	Eligible
Domain	auth.grammarly.com	Critical	Eligible
Domain	capi.grammarly.com	Critical	Eligible
Domain	dox.grammarly.com List, upload, download and edit user documents	Critical	Eligible
Domain	irbis.grammarly.com	Critical	Eligible
Domain	app.grammarly.com	Critical	Eligible

Domain	admin-panel.grammarly.com		Critical		Eligible
hackerone					
Domain	data.grammarly.com		Critical		Eligible
	Store various user settings and properties.				
Domain	*.grammarly.io		Critical		Eligible
Domain	*.grammarly.net		Critical		Eligible
Domain	datareport.grammarly.com		Critical		Eligible
	A web service storing "Personal Data Reports"				
Domain	subscription.grammarly.com		Critical		Eligible
Domain	institution.grammarly.com		Critical		Eligible
Domain	account.grammarly.com		Critical		Eligible
Domain	g-mail.grammarly.com		Critical		Eligible

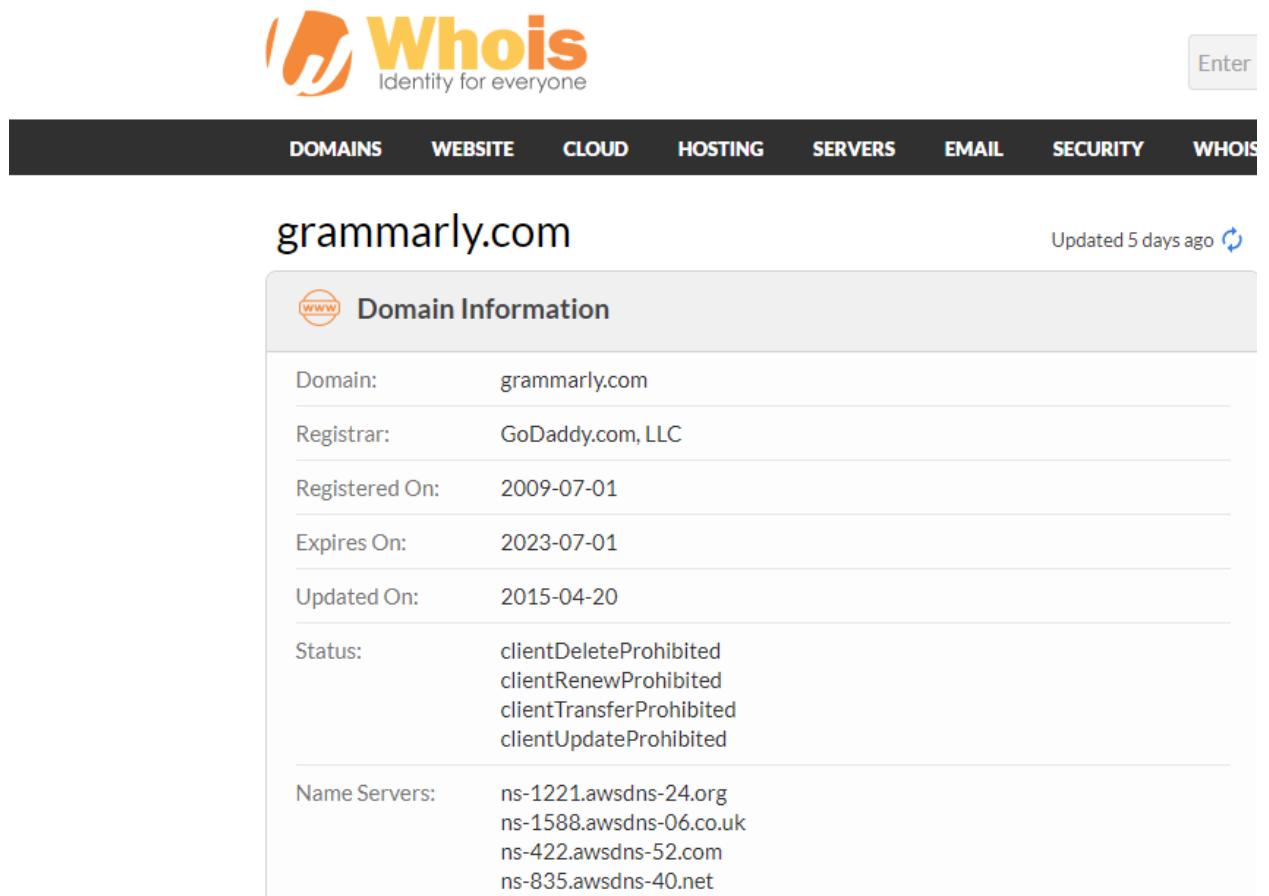
2. Reconnaissance

I. Target validation

Goal confirmation ensures that the target we are auditing is within reach and is the one provided by the customer. Several tools, such as Whois, nslookup, and dnsrecon, can be used to achieve this goal.

i. Whois

“Whois” is a popular Internet record listing that tells you who holds a domain and how to contact them. Domain name registration and possession was governed by the Internet Corporation for Assigned Names and Numbers (ICANN). “Whois” records have proved to be highly valuable and have evolved into an invaluable database for protecting the registration and integrity of domain names and websites.



The screenshot shows the Whois website interface. At the top, there's a logo with the text "Whois" and "Identity for everyone". Below the logo is a search bar with the placeholder "Enter" and a "WHOIS" button. A navigation bar below the search bar includes links for DOMAINS, WEBSITE, CLOUD, HOSTING, SERVERS, EMAIL, SECURITY, and WHOIS. The main content area displays domain information for "grammarly.com", which was updated 5 days ago. The information includes:

Domain Information	
Domain:	grammarly.com
Registrar:	GoDaddy.com, LLC
Registered On:	2009-07-01
Expires On:	2023-07-01
Updated On:	2015-04-20
Status:	clientDeleteProhibited clientRenewProhibited clientTransferProhibited clientUpdateProhibited
Name Servers:	ns-1221.awsdns-24.org ns-1588.awsdns-06.co.uk ns-422.awsdns-52.com ns-835.awsdns-40.net

 Registrant Contact
Organization: Grammarly, Inc.
State: California
Country: US
Email: Select Contact Domain Holder link at https://www.godaddy.com/whois/results.aspx?domain=GRAMMARLY.COM
 Administrative Contact
Email: Select Contact Domain Holder link at https://www.godaddy.com/whois/results.aspx?domain=GRAMMARLY.COM
 Technical Contact
Email: Select Contact Domain Holder link at https://www.godaddy.com/whois/results.aspx?domain=GRAMMARLY.COM

II. Finding Subdomains

Finding subdomains for a one or more domains is known as the sub-domain enumeration which is a very essential part in reconnaissance stage. Finding vulnerabilities on web domain is a very complex task since web developers and organizations are very keen on their website. But with the help of sub-domain enumeration can reveal lot of vulnerabilities which are in the scope.

Therefore, I have selected some of sub-domain enumeration tool to find the sub-domains of the selected domain.

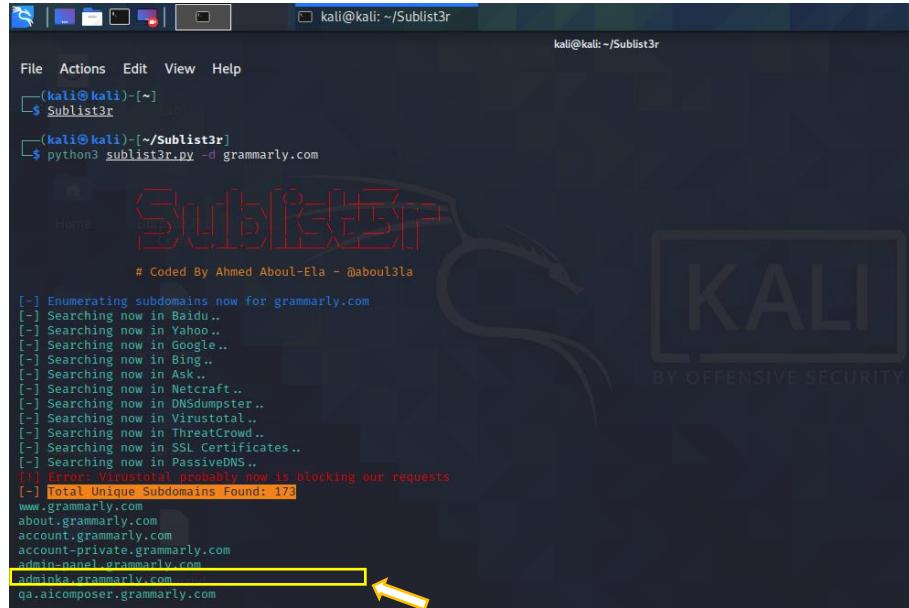
i. Sublist3r

Sublist3r is a python tool which uses OSINT to enumerate website subdomains. It assists penetration testers and bug hunters in gathering and collecting subdomains for the domain they are targeting. Google, Yahoo, Bing, Baidu, and Ask are variety of search engines that used by Sublist3r to find sub-domains.

Sublist3r can install in Kali Linux and any other Linux based distributors by running “[git clone https://github.com/aboul3la/Sublist3r.git](https://github.com/aboul3la/Sublist3r.git)” in the terminal. After

getting into the sublist3r directory we can find sub domain of the main domain by running the following command.

```
python3 sublist3r.py -d grammarly.com
```



```
kali@kali: ~/Sublist3r
File Actions Edit View Help
(kali㉿kali)-[~]
$ Sublist3r
(kali㉿kali)-[~/Sublist3r]
$ python3 sublist3r.py -d grammarly.com

SUBREDFTER
# Coded By Ahmed Aboul-Ela - @aboul3la

[-] Enumerating subdomains now for grammarly.com
[-] Searching now in Baidu..
[-] Searching now in Yahoo..
[-] Searching now in Google..
[-] Searching now in Bing..
[-] Searching now in Ask..
[-] Searching now in Netcraft..
[-] Searching now in DNSdumpster..
[-] Searching now in VirusTotal..
[-] Searching now in ThreatCrowd..
[-] Searching now in SSL Certificates..
[-] Searching now in PassiveDNS..
[!] Error: VirusTotal probably now is blocking our requests
[-] Total Unique Subdomains Found: 173
www.grammarly.com
about.grammarly.com
account.grammarly.com
account-private.grammarly.com
admin-panel.grammarly.com
adminka.grammarly.com
qa.alicomposer.grammarly.com
```

After running that command, I was able to be found **172 number of unique subdomains** in my target domain. Some of them are shown in the below figure.

```
kali@kali: ~/Sublist3r
File Actions Edit View Help
billing-ui.grammarly.com
blog.grammarly.com
www.blog.grammarly.com
calendar.grammarly.com
capi.grammarly.com
capi-msdk.grammarly.com
chef.grammarly.com
chipmunk.grammarly.com
chipmunk-private.grammarly.com
ci.grammarly.com
contenthub.grammarly.com
contenthub-private.grammarly.com
contenthub-static.grammarly.com
context.grammarly.com
corgi.grammarly.com
dc-01.corp.grammarly.com
dc-02.corp.grammarly.com
cv.grammarly.com
dashboard.grammarly.com
data.grammarly.com
data-internal.grammarly.com
datareport.grammarly.com
db.grammarly.com
demo.grammarly.com
denali-fb.grammarly.com
denali-static.grammarly.com
developer.grammarly.com
devoxx.grammarly.com
diagnostics.grammarly.com
dockerhub.grammarly.com
docproc.grammarly.com
download-bundle.grammarly.com
download-editor.grammarly.com
download-editor-predeploy.grammarly.com
download-mac.grammarly.com
```

```
spdmta3.send.grammarly.com
view.send.grammarly.com
sentinel.grammarly.com
sentinel-private.grammarly.com
snippets.grammarly.com
speak.grammarly.com
sso.grammarly.com
sso-institutions.grammarly.com
static.grammarly.com
stats-public.grammarly.com
status.grammarly.com
subscription.grammarly.com
support.grammarly.com
tech.grammarly.com
tokenism.grammarly.com
tokenism-private.grammarly.com
tokenism-relay.grammarly.com
tokenism-relay-private.grammarly.com
tr.grammarly.com
tracking-storage.grammarly.com
treatment.grammarly.com
trumpet.grammarly.com
ua-gec-dataset.grammarly.com
update.grammarly.com
update-officeaddin.grammarly.com
vox-static.grammarly.com
preprod.vox-static.grammarly.com
qa.vox-static.grammarly.com
vpn1.grammarly.com
wiki.grammarly.com
words.grammarly.com
wstest.grammarly.com
www2.grammarly.com
zoom-callbacks.grammarly.com
zoomarly.grammarly.com
```

ii. crt.sh

Simply, crt.sh can be defined as a web based tool to find subdomain, important things and also to do fingerprinting on a selected domain. There is specific column for crt.sh.ID and certificate issuer name and it will help to group the domains which are certified by a specific CA.

Certificates	crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
	4428355714	2021-04-25	2021-04-16	2022-05-15	sso.grammarly.com	sso.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4428333520	2021-04-25	2021-04-21	2022-05-20	capi.grammarly.com	capi.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4424956950	2021-04-24	2021-04-24	2021-07-23	go.grammarly.com	go.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4424957114	2021-04-24	2021-04-24	2021-07-23	go.grammarly.com	go.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4414881059	2021-04-22	2021-04-22	2022-05-21	speak.grammarly.com	speak.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4409498909	2021-04-21	2021-04-21	2022-05-20	capi.grammarly.com	capi.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4409393814	2021-04-21	2021-04-21	2022-05-20	capi.grammarly.com	capi.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4382706415	2021-04-16	2021-04-16	2022-05-15	sso.grammarly.com	sso.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4356296802	2021-04-10	2021-04-10	2021-07-09	api-platform-status.production.tax.service.gov.uk	status.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4356297093	2021-04-10	2021-04-10	2021-07-09	api-platform-download-windows.grammarly.com	status.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4350834518	2021-04-09	2020-11-12	2021-12-11	download-windows.grammarly.com	download-windows.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4341274392	2021-04-07	2021-04-07	2022-05-06	download-mac.grammarly.com	download-mac.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4331813260	2021-04-05	2021-04-05	2022-05-04	access-manager.private.grammarly.com	access-manager.private.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4303279170	2021-03-31	2021-03-31	2021-06-29	hello.grammarly.com	hello.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4303279432	2021-03-31	2021-03-31	2021-06-29	hello.grammarly.com	hello.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4301227622	2021-03-31	2021-03-30	2021-06-28	support.grammarly.com	support.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4301228601	2021-03-31	2021-03-30	2021-06-28	support.grammarly.com	support.grammarly.com	C=US,O=Let's Encrypt,CN=R3
	4281496439	2021-03-27	2021-02-09	2022-03-10	developer.grammarly.com	developer.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4273089963	2021-03-25	2020-11-24	2021-12-23	iosapp-beta.grammarly.com	iosapp-beta.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4234124424	2021-03-18	2021-03-09	2022-04-07	msync.grammarly.com	msync.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4230965805	2021-03-18	2021-03-18	2022-04-16	officeaddin.grammarly.com	officeaddin.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4227569928	2021-03-17	2021-03-17	2022-04-25	goldengate.grammarly.com	goldengate.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon
	4213964133	2021-03-14	2021-02-19	2022-03-20	iosanno.grammarly.com	iosanno.grammarly.com	C=US,O=Amazon,OU=Server CA 1B,CN=Amazon

Above figure shows the results of my target domain using crt.sh.

iii. Subfinder

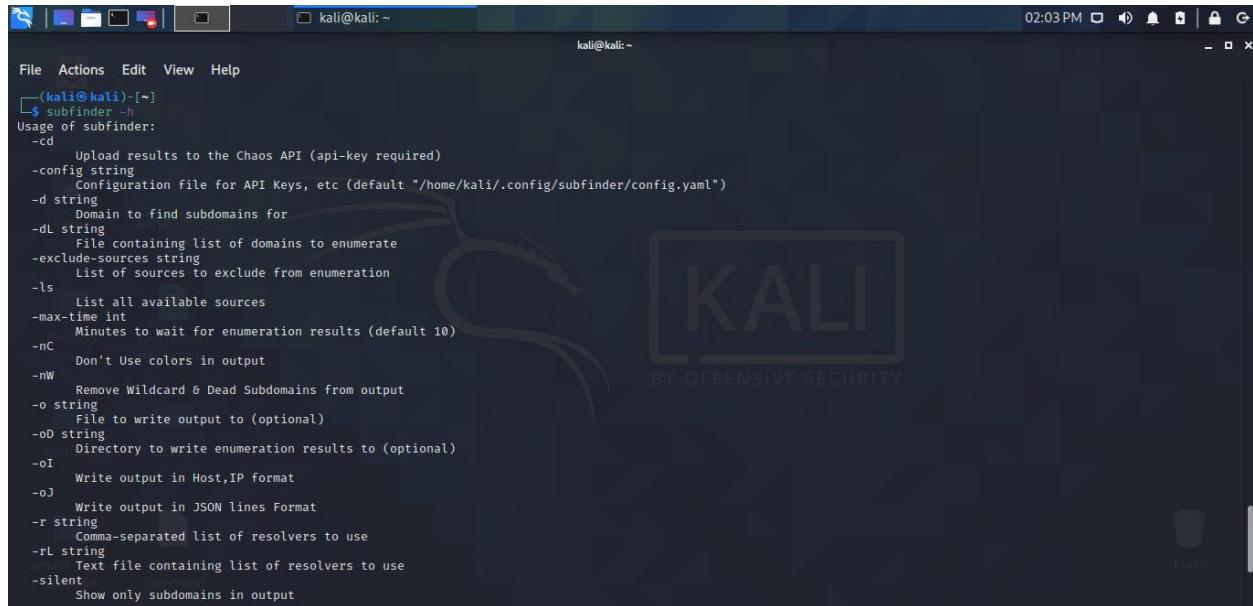
Subfinder is a subdomain exploration platform that uses passive online sources to locate relevant subdomains for websites. It has a straightforward modular design that is designed for speed. Subfinder was designed to do one thing and one thing well: passive subdomain enumeration. It is designed as a subfinder to adhere to all passive source permits and usage limitations while retaining a transparent passive model, making it useful for both penetration testers and bug bounty hunters.

There are different ways for the subfinder installation. Executing the following commands, we can simply install it from GitHub.

```
git clone https://github.com/projectdiscovery/subfinder.git  
cd subfinder/v2/cmd/subfinder  
go build .  
mv subfinder /usr/local/bin/
```

Using the following command, we can see the help for the tool, and it will clearly define what it does for each and every flag.

subfinder -h



The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal title is 'kali@kali: ~'. The command entered is 'subfinder -h'. The output is the usage information for the subfinder tool, which includes various flags and their descriptions. The usage text is as follows:

```
File Actions Edit View Help  
[kali㉿kali] -[~]  
$ subfinder -h  
Usage of subfinder:  
-td  
    Upload results to the Chaos API (api-key required)  
-config string  
    Configuration file for API Keys, etc (default "/home/kali/.config/subfinder/config.yaml")  
-d string  
    Domain to find subdomains for  
-dL string  
    File containing list of domains to enumerate  
-exclude-sources string  
    List of sources to exclude from enumeration  
-ls  
    List all available sources  
-max-time int  
    Minutes to wait for enumeration results (default 10)  
-nC  
    Don't Use colors in output  
-nW  
    Remove Wildcard & Dead Subdomains from output  
-o string  
    File to write output to (optional)  
-oD string  
    Directory to write enumeration results to (optional)  
-oI  
    Write output in Host,IP format  
-oJ  
    Write output in JSON lines Format  
-r string  
    Comma-separated list of resolvers to use  
-rL string  
    Text file containing list of resolvers to use  
-silent  
    Show only subdomains in output
```

As we already done in Sublitr3r we can find sub domain of the main domain by running the following command.

```
subfinder -d grammarly.com
```



```
kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ subfinder -d grammarly.com
projectdiscovery.io

[WRN] Use with caution. You are responsible for your actions
[WRN] Developers assume no liability and are not responsible for any misuse or damage.
[WRN] By using subfinder, you also agree to the terms of the APIs used.

[INF] Enumerating subdomains for grammarly.com
redirect.grammarly.com
tf.grammarly.com
image.send.grammarly.com
auth.grammarly.com
download-office.grammarly.com
update.grammarly.com
download-editor.grammarly.com
balancer0.grammarly.com
vpn1.grammarly.com
balancer1.grammarly.com
balancer2.grammarly.com
www2.grammarly.com
balancer3.grammarly.com
balancer4.grammarly.com
balancer5.grammarly.com
balancer6.grammarly.com
balancer7.grammarly.com
balancer8.grammarly.com
adminka.grammarly.com
jira.grammarly.com
```

As shown in the above figure, after running that command I was able to find several numbers of unique subdomains in my target domain.

iv. Amass

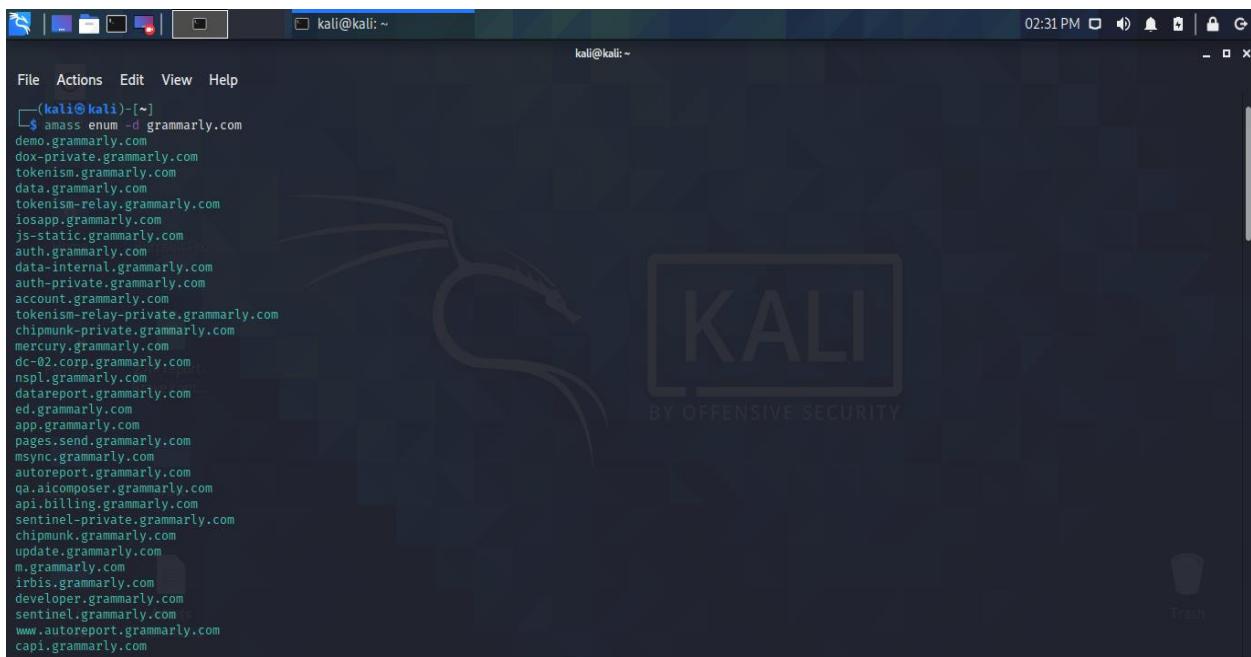
Amass is a great tool for gathering information on the attack surface of targets in multiple dimensions and it is based on OWASP. This tool really helps for penetration testers by mapping the organization's attack surface and with the use of open source information gathering and active reconnaissance techniques it allows to perform external asset discovery.

We can simply install it from GitHub using git clone command as shown below.

```
git clone https://github.com/OWASP/Amass.git
```

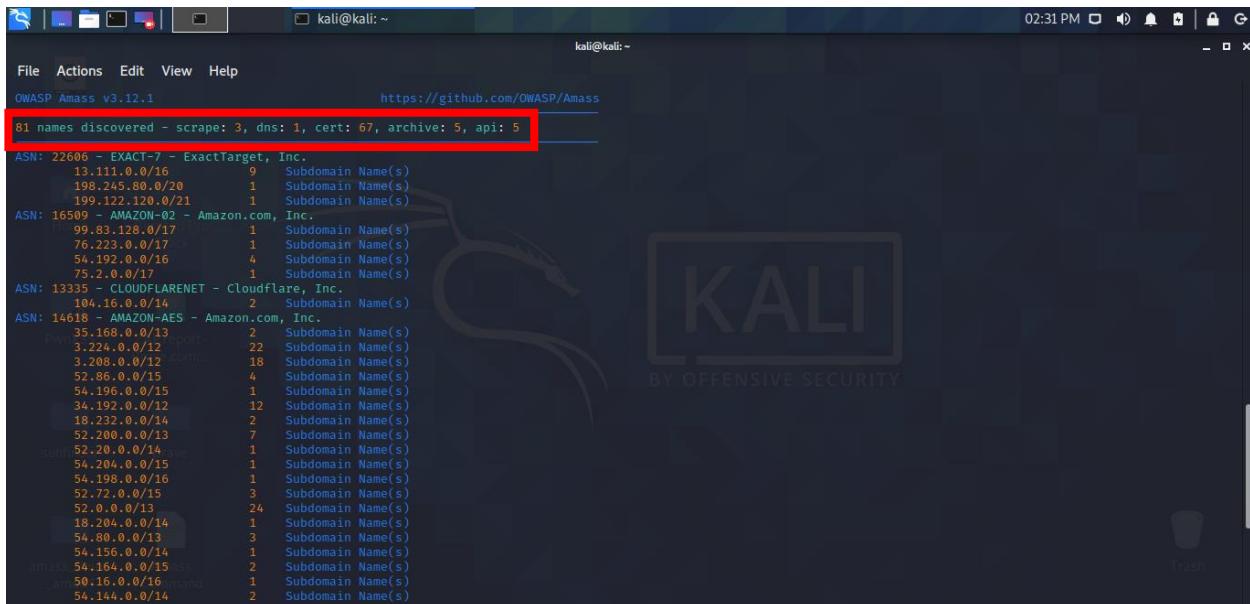
Here I used Amass tool in order to gather more and relevant information regarding my sub-domains of my target domain of grammarly.com.

```
amass enum -d grammarly.com
```



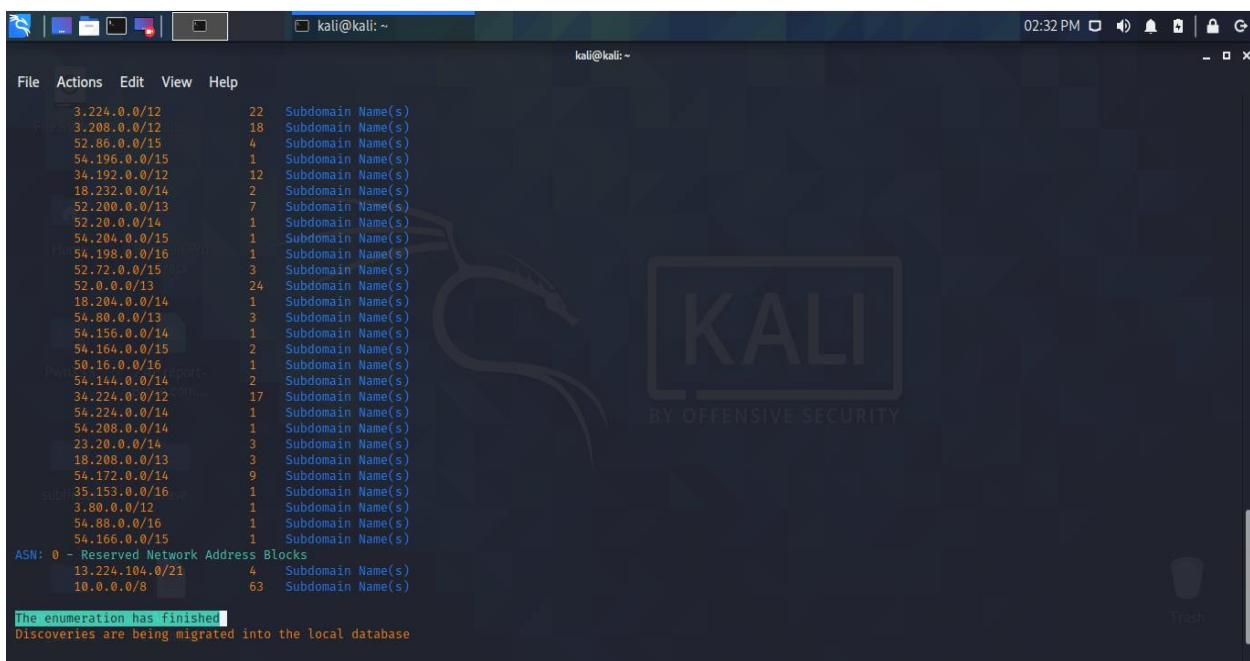
```
(kali㉿kali)-[~] $ amass enum -d grammarly.com
demo.grammarly.com
dox-private.grammarly.com
tokenism.grammarly.com
data.grammarly.com
tokenism-relay.grammarly.com
iosapp.grammarly.com
js-static.grammarly.com
auth.grammarly.com
data-internal.grammarly.com
auth-private.grammarly.com
account.grammarly.com
tokenism-relay-private.grammarly.com
chimpunk-private.grammarly.com
mercury.grammarly.com
dc-02.corp.grammarly.com
nspl.grammarly.com
datareport.grammarly.com
ed.grammarly.com
app.grammarly.com
pages.send.grammarly.com
msync.grammarly.com
autoreport.grammarly.com
qa.aicomposer.grammarly.com
api.billing.grammarly.com
sentinel-private.grammarly.com
chimpunk.grammarly.com
update.grammarly.com
m.grammarly.com
irbis.grammarly.com
developer.grammarly.com
sentinel.grammarly.com
www.autoreport.grammarly.com
capi.grammarly.com
```

As shown in below, at the end of subdomain scan, it generates a report which contains all web server details, number of names discovered, IP address ranges of subdomains and their certificate issuers. Even it takes more time than the other sub domain scanners, at the end of it provide this categorized report and it is really helping to get a clear understand of the target domain and its sub domains.



```
File Actions Edit View Help https://github.com/OWASP/Amass
81 names discovered - scrape: 3, dns: 1, cert: 67, archive: 5, api: 5
ASN: 22606 - EXACT-7 - ExactTarget, Inc.
13.111.0.0/16 9 Subdomain Name(s)
198.245.80.0/20 1 Subdomain Name(s)
199.122.120.0/21 1 Subdomain Name(s)
ASN: 16509 - AMAZON-02 - Amazon.com, Inc.
99.83.128.0/17 1 Subdomain Name(s)
76.223.0.0/17 1 Subdomain Name(s)
54.192.0.0/16 4 Subdomain Name(s)
75.2.0.0/17 1 Subdomain Name(s)
ASN: 13335 - CLOUDFLARENET - Cloudflare, Inc.
104.16.0.0/14 2 Subdomain Name(s)
ASN: 14618 - AMAZON-AES - Amazon.com, Inc.
35.168.0.0/13 2 Subdomain Name(s)
3.224.0.0/12 22 Subdomain Name(s)
3.208.0.0/12 18 Subdomain Name(s)
52.86.0.0/15 4 Subdomain Name(s)
54.196.0.0/15 1 Subdomain Name(s)
54.192.0.0/12 12 Subdomain Name(s)
18.232.0.0/14 2 Subdomain Name(s)
52.200.0.0/13 7 Subdomain Name(s)
52.20.0.0/14 1 Subdomain Name(s)
54.204.0.0/15 1 Subdomain Name(s)
54.198.0.0/16 1 Subdomain Name(s)
52.72.0.0/15 3 Subdomain Name(s)
52.0.0.0/13 24 Subdomain Name(s)
18.204.0.0/14 1 Subdomain Name(s)
54.80.0.0/13 3 Subdomain Name(s)
54.156.0.0/14 1 Subdomain Name(s)
54.164.0.0/15 2 Subdomain Name(s)
50.16.0.0/16 1 Subdomain Name(s)
54.144.0.0/14 2 Subdomain Name(s)
34.224.0.0/12 17 Subdomain Name(s)
54.224.0.0/14 1 Subdomain Name(s)
54.208.0.0/14 1 Subdomain Name(s)
23.20.0.0/14 3 Subdomain Name(s)
18.208.0.0/13 3 Subdomain Name(s)
54.172.0.0/14 9 Subdomain Name(s)
35.153.0.0/16 1 Subdomain Name(s)
3.80.0.0/12 1 Subdomain Name(s)
54.88.0.0/16 1 Subdomain Name(s)
54.166.0.0/15 1 Subdomain Name(s)
ASN: 0 - Reserved Network Address Blocks
13.224.104.0/21 4 Subdomain Name(s)
10.0.0.0/8 63 Subdomain Name(s)

The enumeration has finished.
Discoveries are being migrated into the local database
```



```
File Actions Edit View Help https://github.com/OWASP/Amass
81 names discovered - scrape: 3, dns: 1, cert: 67, archive: 5, api: 5
ASN: 22606 - EXACT-7 - ExactTarget, Inc.
13.111.0.0/16 9 Subdomain Name(s)
198.245.80.0/20 1 Subdomain Name(s)
199.122.120.0/21 1 Subdomain Name(s)
ASN: 16509 - AMAZON-02 - Amazon.com, Inc.
99.83.128.0/17 1 Subdomain Name(s)
76.223.0.0/17 1 Subdomain Name(s)
54.192.0.0/16 4 Subdomain Name(s)
75.2.0.0/17 1 Subdomain Name(s)
ASN: 13335 - CLOUDFLARENET - Cloudflare, Inc.
104.16.0.0/14 2 Subdomain Name(s)
ASN: 14618 - AMAZON-AES - Amazon.com, Inc.
35.168.0.0/13 2 Subdomain Name(s)
3.224.0.0/12 22 Subdomain Name(s)
3.208.0.0/12 18 Subdomain Name(s)
52.86.0.0/15 4 Subdomain Name(s)
54.196.0.0/15 1 Subdomain Name(s)
54.192.0.0/12 12 Subdomain Name(s)
18.232.0.0/14 2 Subdomain Name(s)
52.200.0.0/13 7 Subdomain Name(s)
52.20.0.0/14 1 Subdomain Name(s)
54.204.0.0/15 1 Subdomain Name(s)
54.198.0.0/16 1 Subdomain Name(s)
52.72.0.0/15 3 Subdomain Name(s)
52.0.0.0/13 24 Subdomain Name(s)
18.204.0.0/14 1 Subdomain Name(s)
54.80.0.0/13 3 Subdomain Name(s)
54.156.0.0/14 1 Subdomain Name(s)
54.164.0.0/15 2 Subdomain Name(s)
50.16.0.0/16 1 Subdomain Name(s)
54.144.0.0/14 2 Subdomain Name(s)
34.224.0.0/12 17 Subdomain Name(s)
54.224.0.0/14 1 Subdomain Name(s)
54.208.0.0/14 1 Subdomain Name(s)
23.20.0.0/14 3 Subdomain Name(s)
18.208.0.0/13 3 Subdomain Name(s)
54.172.0.0/14 9 Subdomain Name(s)
35.153.0.0/16 1 Subdomain Name(s)
3.80.0.0/12 1 Subdomain Name(s)
54.88.0.0/16 1 Subdomain Name(s)
54.166.0.0/15 1 Subdomain Name(s)
ASN: 0 - Reserved Network Address Blocks
13.224.104.0/21 4 Subdomain Name(s)
10.0.0.0/8 63 Subdomain Name(s)

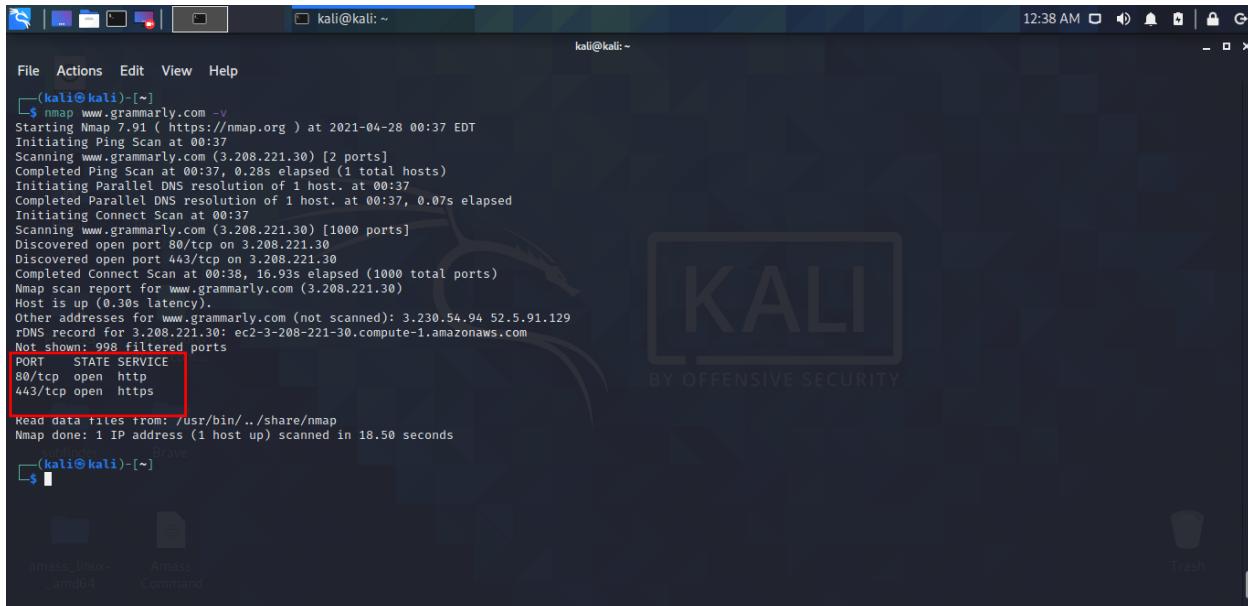
The enumeration has finished.
Discoveries are being migrated into the local database
```

3. Discovery and Scanning

I. Nmap

Nmap (Network Mapper) is a network transparency and security examination tool that is free and open source (permit). Nmap makes use of raw IP packets in creative ways to find out what hosts are available on the network, what administrations (application name and form) those hosts are providing, what working systems (and OS renditions) they are using, what kind of parcel channels/firewalls they are using, and a variety of other things. It was designed to easily check large entities, but it still works well for single hosts. The Nmap suite includes a severe GUI and results watcher (Zenmap), an adaptable information move, redirection, and investigation equipment (Ncat), a utility for looking at search results (Ndifff), and a package age and reaction analysis unit, aside from the well-known order-line Nmap executable (Nping).

First, by running the command of `nmap www.grammarly.com -v` we were able to find two open ports which is common but informative.



```
(kali㉿kali)-[~]
└─$ nmap www.grammarly.com -v
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-28 00:37 EDT
Initiating Ping Scan at 00:37
Scanning www.grammarly.com (3.208.221.30) [2 ports]
Completed Ping Scan at 00:37, 0.28s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 00:37
Completed Parallel DNS resolution of 1 host. at 00:37, 0.07s elapsed
Initiating Connect Scan at 00:37
Scanning www.grammarly.com (3.208.221.30) [1000 ports]
Discovered open port 80/tcp on 3.208.221.30
Discovered open port 443/tcp on 3.208.221.30
Completed Connect Scan at 00:38, 16.93s elapsed (1000 total ports)
Nmap scan report for www.grammarly.com (3.208.221.30)
Host is up (0.30s latency).
Other addresses for www.grammarly.com (not scanned): 3.230.54.94 52.5.91.129
rDNS record for 3.208.221.30: ec2-3-208-221-30.compute-1.amazonaws.com
Not shown: 998 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 18.50 seconds
└─$
```

Using above shown scan we can find the IP address of our target domain as **3.208.221.30** and it can be used for further scans regarding my target domain. Furthermore, we can see there are two open ports which are currently open with specific service.

PORT	STATE	SERVICE
80/tcp	open	http
443/tcp	open	https

When we scanned port 80 specifically using the command **nmap -v -iR 1000 -Pn 80** as shown in bellow it takes more time than the others and it gives more information about discovered open port on tcp with the specific IP address individually.



```
(kali㉿kali)-[~]
$ nmap -v -iR 1000 -Pn 80
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-28 00:51 EDT
Initiating Parallel DNS resolution of 1000 hosts. at 00:51
Completed Parallel DNS resolution of 1000 hosts. at 00:52, 73.37s elapsed
Initiating Connect Scan at 00:52
Scanning 64 hosts [1000 ports/host]
Discovered open port 23/tcp on 119.189.168.0
Discovered open port 21/tcp on 7.78.140.2
Discovered open port 21/tcp on 219.163.106.4
Discovered open port 21/tcp on 100.175.194.5
Discovered open port 21/tcp on 173.162.73.9
Discovered open port 21/tcp on 88.120.65.11
Discovered open port 21/tcp on 27.227.98.18
Discovered open port 21/tcp on 77.38.223.19
Discovered open port 21/tcp on 98.243.241.22
Discovered open port 21/tcp on 66.38.254.25
Discovered open port 21/tcp on 122.109.133.34
Discovered open port 21/tcp on 16.129.166.34
Discovered open port 21/tcp on 215.54.132.37
Discovered open port 21/tcp on 103.204.218.42
Discovered open port 21/tcp on 111.130.119.43
Discovered open port 21/tcp on 38.74.220.46
Discovered open port 21/tcp on 116.168.87.61
Discovered open port 21/tcp on 62.199.92.61
Discovered open port 21/tcp on 8.235.245.62
Discovered open port 21/tcp on 2.206.31.63
Discovered open port 21/tcp on 135.109.204.64
Discovered open port 21/tcp on 143.110.104.65
Discovered open port 21/tcp on 46.189.29.73
Discovered open port 21/tcp on 83.210.154.87
Discovered open port 21/tcp on 186.202.40.89
Discovered open port 21/tcp on 182.137.6.93
Discovered open port 21/tcp on 168.68.0.101
```

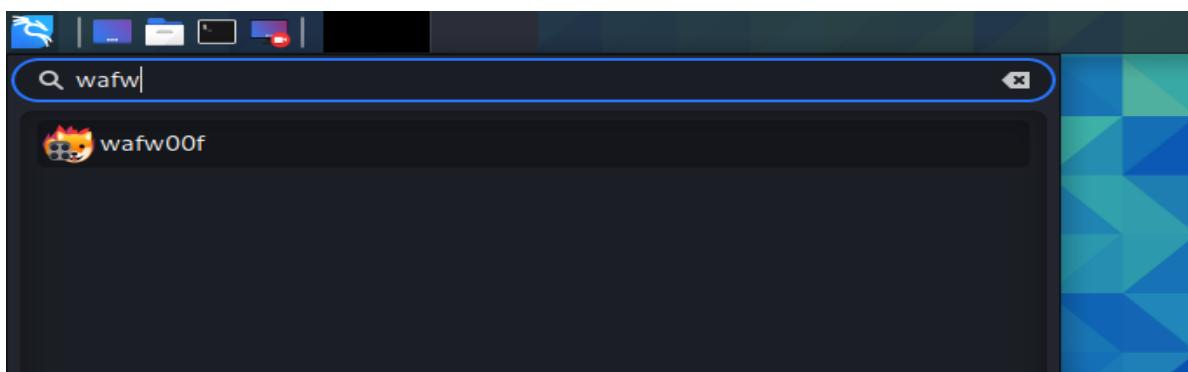


```
kali@kali: ~
File Actions Edit View Help
Discovered open port 113/tcp on 111.130.119.43
Discovered open port 256/tcp on 38.74.220.46
Discovered open port 445/tcp on 8.235.245.62
Discovered open port 25/tcp on 197.67.242.254
Discovered open port 3389/tcp on 119.190.190.141
Discovered open port 113/tcp on 109.246.10.106
Discovered open port 256/tcp on 3.187.254.107
Discovered open port 22/tcp on 133.22.76.130
Discovered open port 113/tcp on 167.8.9.135
Discovered open port 139/tcp on 74.235.219.139
Discovered open port 25/tcp on 168.68.0.101
Discovered open port 111/tcp on 54.43.38.175
Discovered open port 1025/tcp on 86.68.202.158
Discovered open port 3306/tcp on 155.111.111.171
Discovered open port 53/tcp on 182.137.6.93
Discovered open port 22/tcp on 37.114.208.205
Discovered open port 25/tcp on 216.255.35.208
Discovered open port 1723/tcp on 182.137.6.93
Discovered open port 199/tcp on 182.137.6.93
Discovered open port 110/tcp on 23.81.245.210
Discovered open port 1025/tcp on 190.245.169.226
Discovered open port 256/tcp on 58.225.125.223
Connect Scan Timing: About 1.82% done; ETC: 01:20 (0:27:55 remaining)
Connect Scan Timing: About 2.77% done; ETC: 01:29 (0:35:41 remaining)
Connect Scan Timing: About 31.23% done; ETC: 00:57 (0:03:20 remaining)
Connect Scan Timing: About 31.37% done; ETC: 00:58 (0:04:25 remaining)
Connect Scan Timing: About 31.51% done; ETC: 01:00 (0:05:28 remaining)
Connect Scan Timing: About 31.66% done; ETC: 01:01 (0:06:31 remaining)
Connect Scan Timing: About 31.80% done; ETC: 01:03 (0:07:33 remaining)
Connect Scan Timing: About 31.94% done; ETC: 01:04 (0:08:34 remaining)
Connect Scan Timing: About 32.08% done; ETC: 01:06 (0:09:34 remaining)
Connect Scan Timing: About 32.23% done; ETC: 01:07 (0:10:33 remaining)
Connect Scan Timing: About 32.37% done; ETC: 01:09 (0:11:31 remaining)
Connect Scan Timing: About 32.51% done; ETC: 01:10 (0:12:30 remaining)
```

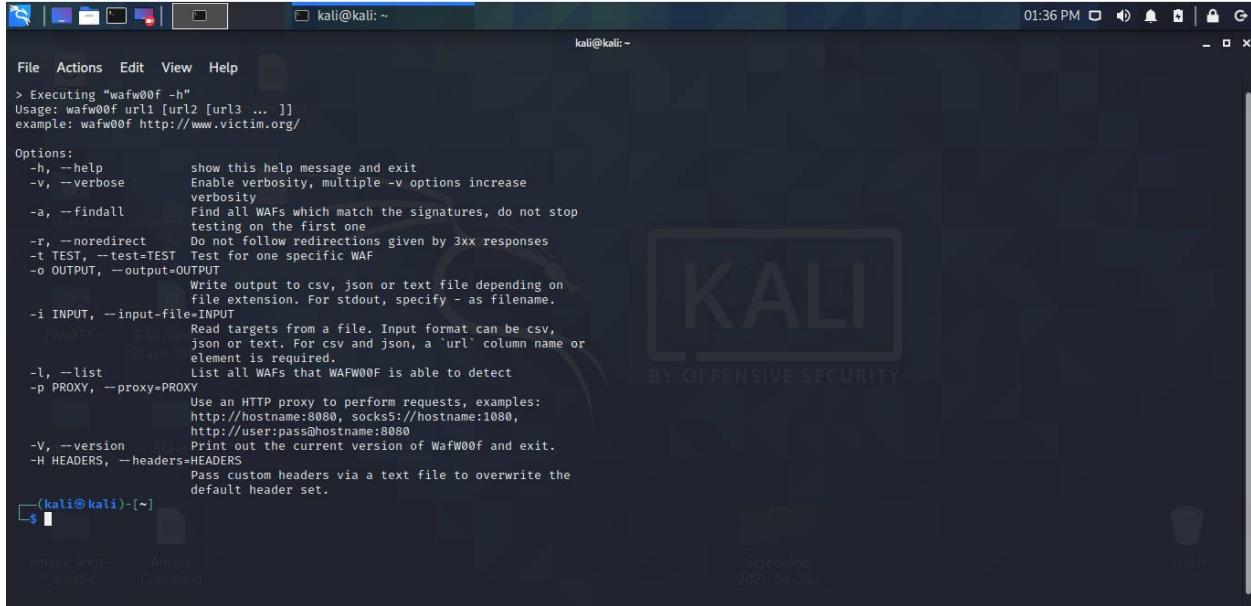
II. Wafw00f

Wafw00f is a well-known Python application that automates the process of fingerprinting a website's firewall. Wafw00f will evaluate the underlying firewall used by a provider it probes based on the answers to a set of precisely designed web requests. Wafw00f is pre-installed in Kali Linux, but it can also be installed on any Python-enabled device. Wafw00f consistently provided more complete and reliable results during testing.

Since Wafw00f is used in the Kali Linux Penetration Testing OS by default, it can be accessed by simply searching and selecting the Wafw00f in the search bar in Kali Linux as shown below.

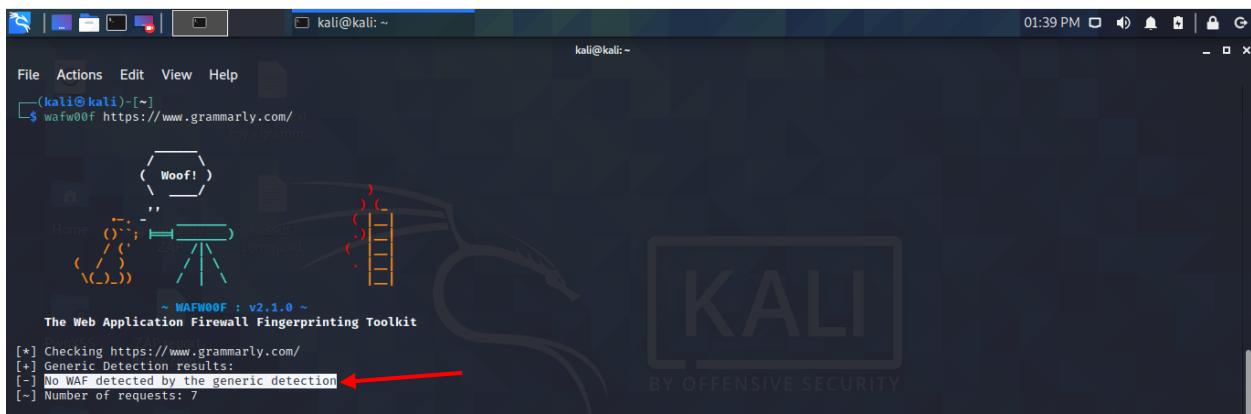


Then it will execute the command of “**wafw00f -h**” by default and it will display the available options as shown below.



```
kali@kali: ~
File Actions Edit View Help
> Executing "wafw00f -h"
Usage: wafw00f url1 [url2 [url3 ... ]]
example: wafw00f http://www.victim.org/
Options:
-h, --help      show this help message and exit
-v, --verbose   Enable verbosity, multiple -v options increase
                verbosity
-a, --findall   Find all WAFs which match the signatures, do not stop
                testing on the first one
-r, --noredirect Do not follow redirections given by 3xx responses
-t TEST, --test=TEST Test for one specific WAF
-o OUTPUT, --output=OUTPUT
                Write output to csv, json or text file depending on
                file extension. For stdout, specify - as filename.
-i INPUT, --input=INPUT
                Read targets from a file. Input format can be csv,
                json or text. For csv and json, a 'url' column name or
                element is required.
-l, --list       List all WAFs that WAFW00F is able to detect
-p PROXY, --proxy=PROXY
                Use an HTTP proxy to perform requests, examples:
                http://hostname:8080, socks5://hostname:1080,
                http://user:pass@hostname:8080
-V, --version    Print out the current version of WafW00f and exit.
-H HEADERS, --headers=HEADERS
                Pass custom headers via a text file to overwrite the
                default header set.
(kali㉿kali)-[~]
```

In order to scan our target, I have executed the command with the target URL.
wafw00f <https://www.grammarly.com/>



```
(kali㉿kali)-[~]
$ wafw00f https://www.grammarly.com/
```

The terminal output shows the following:

```
[*] Checking https://www.grammarly.com/
[*] Generic Detection results:
[+] No WAF detected by the generic detection
[-] Number of requests: 7
```

A red arrow points to the line "[+] No WAF detected by the generic detection".

Scan was completed by displaying that the target domain has not a Web Application Firewall (WAF). This can be considered as a critical vulnerability because we can reduce intruder attacks by 20% with the help of WAF.

i. Solution:

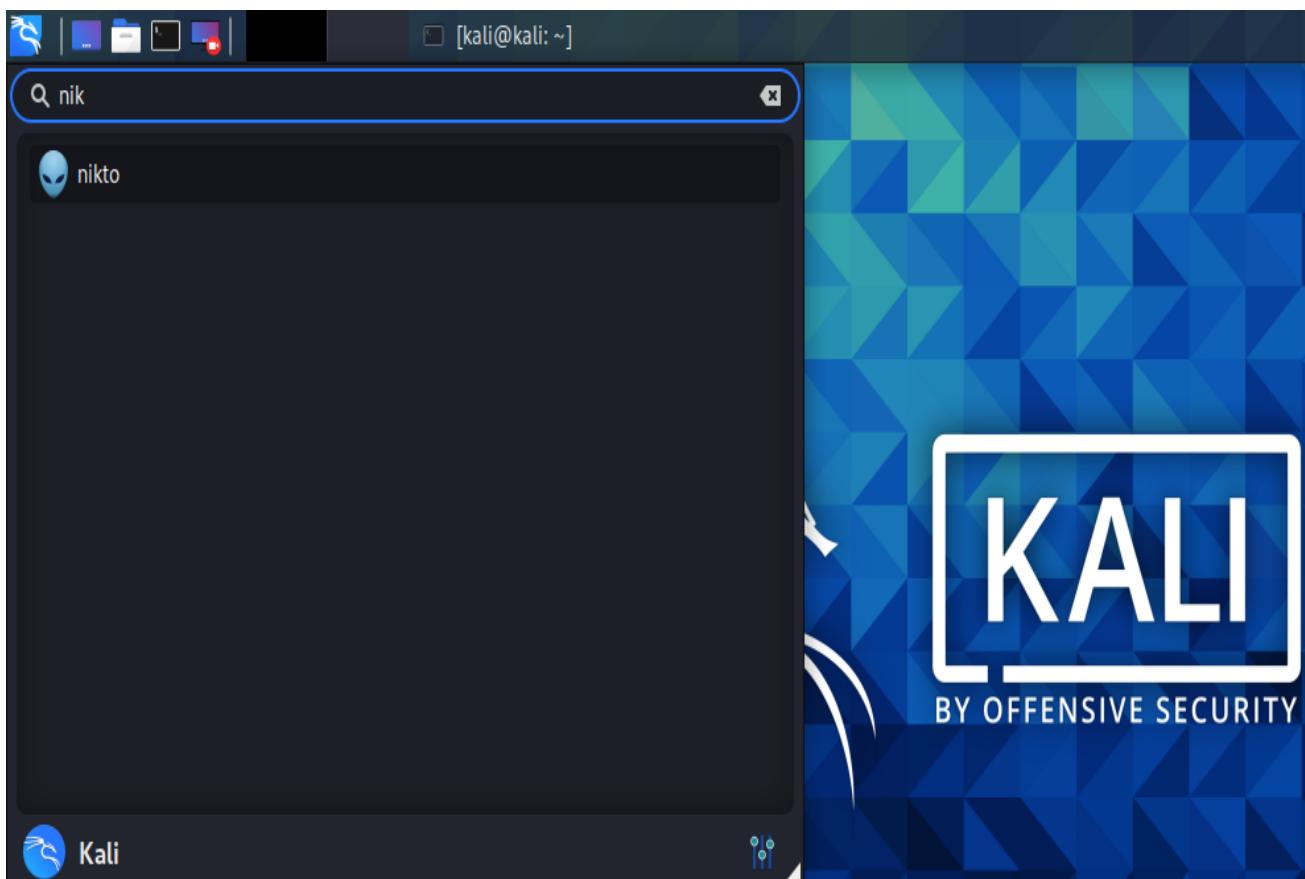
Through filtering and tracking HTTP traffic between a web application and the Internet, a WAF, or Web Application Firewall, aids in the protection of web apps. It usually defends web applications from threats like cross-site forgery, cross-site scripting (XSS), file inclusion, and SQL injection. A WAF (in the OSI model) is a protocol layer 7 security that is not intended to protect against all forms of attacks. This type of attack prevention is typically part of a larger set of techniques that work together to provide a comprehensive response against a variety of threats. A WAF acts as a barrier between a web application and the Internet when it is deployed in front of it.

4. Vulnerability Assessment

I. Nikto

Nikto is an Open Source (GPL) web worker scanner that runs comprehensive tests against web workers for a variety of items, including over 6700 potentially dangerous documents/programs, scans for outdated adaptations on over 1250 servers, and type explicit problems on over 270 servers. It also looks for server architecture elements like the inclusion of multiple record documentation and HTTP server options, as well as attempting to understand installed web servers and programming. Nikto can detect a wide variety of problems as well as look at configuration issues. This tool is also pre-installed in Kali.

We can run it by simply searching in Kali environment as shown below.





```
[kali㉿kali: ~] kali@kali: ~ 01:13 AM
```

```
File Actions Edit View Help
> Executing "nikto -h"
Option host requires an argument
  -config+      Use this config file
  -Display+     Turn on/off display outputs
  -dbcheck       check database and other key files for syntax errors
  -Format+      save file (-o) format
  -Help          Extended help information
  -host+        target host/URI
  -id+          Host authentication to use, format is id:pass or id:pass:realm
  -list-plugins List all available plugins
  -output+      Write output to this file
  -nssl          Disables using SSL
  -no404        Disables 404 checks
  -Plugins+     List of plugins to run (default: ALL)
  -port+        Port to use (default 80)
  -root+        Prepend root value to all requests, format is /directory
  -ssl          Force ssl mode on port
  -Tuning+      Scan tuning
  -timeout+    Timeout for requests (default 10 seconds)
  -update       Update databases and plugins from CIRT.net
  -Version      Print plugin and database versions
  -vhost+      Virtual host (for Host header)

  + requires a value
Note: This is the short help output. Use -H for full help text.
```

We can find all the options which are available in Nikto tool by running “**nikto -Help**” command in open window as shown below.

nikto -Help



```
[kali㉿kali: ~] kali@kali: ~ 01:14 AM
```

```
File Actions Edit View Help
└─(kali㉿kali)-[~]
$ nikto -Help

Options:
  -ask+      Whether to ask about submitting updates
            yes  Ask about each (default)
            no   Don't ask, don't send
            auto Don't ask, just send
  -Cgidirs+  Scan these CGI dirs: 'none', 'all', or values like "/cgi/ /cgi-a/"
  -config+   Use this config file
  -Display+  Turn on/off display outputs:
            1   Show redirects
            2   Show cookies received
            3   Show all 200/OK responses
            4   Show URLs which require authentication
            D   Debug output
            E   Display all HTTP errors
            P   Print progress to STDOUT
            S   Scrub output of IPs and hostnames
            V   Verbose output
  -dbcheck    Check database and other key files for syntax errors
  -evasion+  Encoding technique:
            1   Random URI encoding (non-UTF8)
            2   Directory self-reference (./.)
            3   Premature URL ending
            4   Prepend long random string
            5   Fake parameter
            6   TAB as request spacer
            7   Change the case of the URL
            8   Use Windows directory separator (\)
            A   Use a carriage return (0xd) as a request spacer
            B   Use binary value 0xb as a request spacer
  -Format+   Save file (-o) format:
            csv  Comma-separated-value
```



```
[kali㉿kali:~] kali@kali:~
```

File Actions Edit View Help

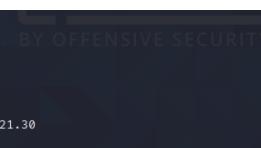
```
-port+ Port to use (default 80)
-RSACert+ Client certificate file
-root+ Prepend root value to all requests, format is /directory
-Save Save positive responses to this directory ('.' for auto-name)
-ssl Force ssl mode on port
-Tuning+ Scan tuning:
    1 Interesting File / Seen in logs
    2 Misconfiguration / Default File
    3 Information Disclosure
    4 Injection (XSS/Script/HTML)
    5 Remote File Retrieval - Inside Web Root
    6 Denial of Service
    7 Remote File Retrieval - Server Wide
    8 Command Execution / Remote Shell
    9 SQL Injection
    0 File Upload
    a Authentication Bypass
    b Software Identification
    c Remote Source Inclusion
    d WebService
    e Administrative Console
    x Reverse Tuning Options (i.e., include all except specified)

-timeout+ Timeout for requests (default 10 seconds)
-Userdbs Load only user databases, not the standard databases
    all Disable standard dbs and load only user dbs
    tests Disable only db_tests and load udb_tests

-useragent Over-rides the default useragent
-until Run until the specified time or duration
-update Update databases and plugins from CIRT.net
-url+ Target host/URL (alias of -host)
-useproxy Use the proxy defined in nikto.conf, or argument http://server:port
-Version Print plugin and database versions
-vhost+ Virtual host (for Host header)

+ and - + requires a value
```

Using the “`nikto -h grammarrly.com`” we can find that the X-XSS-Protection header is not defined, and server name can be found as shown below.



```
(kali㉿kali:[~]) ~]$ nikto -h grammarrly.com
- Nikto v2.1.6

+ Target IP:      52.5.91.129
+ Target Hostname: grammarrly.com
+ Target Port:     80
+ Message:        Multiple IP addresses found: 52.5.91.129, 3.230.54.94, 3.208.221.30
+ Start Time:    2021-04-28 01:19:38 (GMT-4)

+ Server: awselb/2.0
[+] Server: awselb/2.0 [+] Header: X-Frame-Options header is not present. [Yellow Box]
[+] The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS [Yellow Arrow]
[+] The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://www.grammarrly.com:443/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```



```
(kali㉿kali:[~]) ~]$ nikto -h grammarrly.com
- Nikto v2.1.6

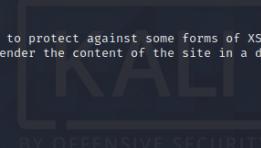
+ Target IP:      52.5.91.129
+ Target Hostname: grammarrly.com
+ Target Port:     80
+ Message:        Multiple IP addresses found: 52.5.91.129, 3.230.54.94, 3.208.221.30
+ Start Time:    2021-04-28 01:19:38 (GMT-4)

+ Server: awselb/2.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://www.grammarrly.com:443/
[+] No CGI Directories found (use '-C all' to force check all possible dirs)
[+] 7786 requests: 0 error(s) and 3 item(s) reported on remote host
+ End time:       2021-04-28 02:03:10 (GMT-4) (2612 seconds)

+ 1 host(s) tested
```

In order to complete this scan, it takes more than 40 minutes, but it cannot find any error requests within all 7786 requests.

Then I scanned my target domain with the number of 1 option for the Interesting File / Seen in logs command as shown below, and it does not give any vulnerability and end up with 0 errors within all of 2139 total requests.



```
(kali㉿kali)-[~]
$ nikto -h grammarly.com -Tuning 1
- Nikto v2.1.6

+ Target IP:          3.208.221.30
+ Target Hostname:   grammarly.com
+ Target Port:        80
+ Message:           Multiple IP addresses found: 3.208.221.30, 3.230.54.94, 52.5.91.129
+ Start Time:        2021-04-28 02:28:41 (GMT-4)

+ Server: awselb/2.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://www.grammarly.com:443/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 2139 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:          2021-04-28 02:42:26 (GMT-4) (825 seconds)

+ 1 host(s) tested
```

Then I scanned my target domain with the number of 4 option for the Injection (XSS/Script/HTML) command as shown below, and it does not give any vulnerability and end up with 0 errors.

```
(kali㉿kali)-[~]
$ nikto -h grammarly.com -Tuning 4
- Nikto v2.1.6

+ Target IP:          52.5.91.129
+ Target Hostname:   grammarly.com
+ Target Port:        80
+ Message:           Multiple IP addresses found: 52.5.91.129, 3.230.54.94, 3.208.221.30
+ Start Time:        2021-04-28 02:02:25 (GMT-4)

+ Server: awselb/2.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not set. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://www.grammarly.com:443/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 779 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:          2021-04-28 02:07:16 (GMT-4) (291 seconds)

+ 1 host(s) tested
```

Next, I scanned my target domain with the number of 9 option for the SQL Injection command as shown below, and it does not give any vulnerability and end up with 0 errors as previous.

```
(kali㉿kali)-[~]
$ nikto -h grammarly.com -Tuning 9
- Nikto v2.1.6

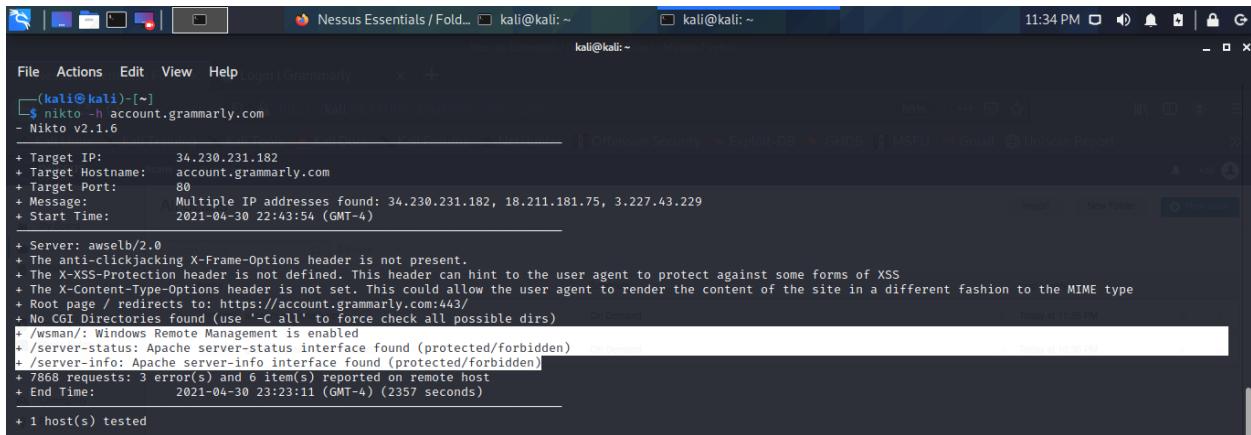
+ Target IP:          3.230.54.94
+ Target Hostname:   grammarly.com
+ Target Port:        80
+ Message:           Multiple IP addresses found: 3.230.54.94, 52.5.91.129, 3.208.221.30
+ Start Time:        2021-04-28 02:13:30 (GMT-4)

+ Server: awselb/2.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: https://www.grammarly.com:443/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 524 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:          2021-04-28 02:17:19 (GMT-4) (229 seconds)

+ 1 host(s) tested
```

Furthermore, I have scanned my sub-domains in order to find IP addresses and errors using Nikto tool as shown below. Those sub-domains were already fallen under the scope of bug-bounty rules and regulations.

As the first sub-domain I have scanned account.grammarly.com using the command of “**nikto -h account.grammarly.com**” as shown below.



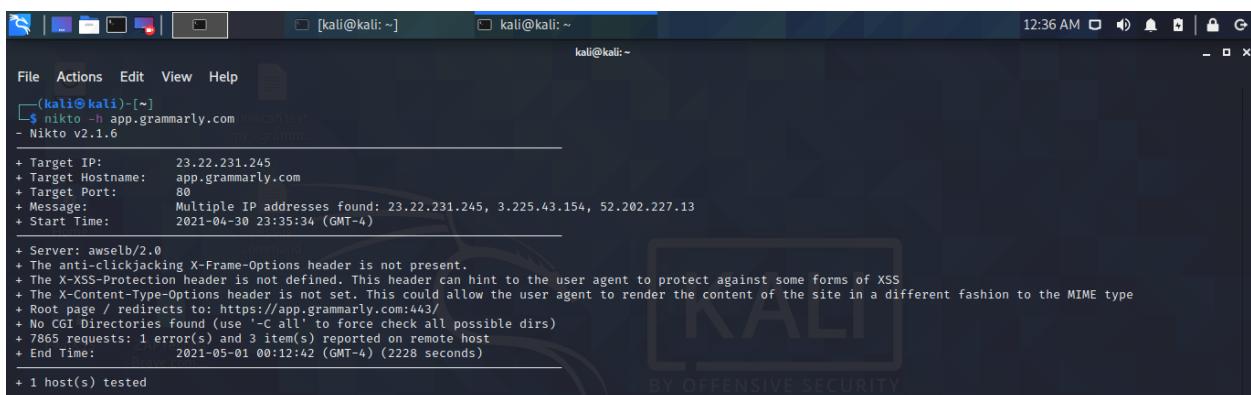
```
(kali㉿kali)-[~] $ nikto -h account.grammarly.com
[+] Nikto v2.1.6
[+] Target IP: 34.230.231.182
[+] Target Hostname: account.grammarly.com
[+] Target Port: 80
[+] Message: Multiple IP addresses found: 34.230.231.182, 18.211.181.75, 3.227.43.229
[+] Start Time: 2021-04-30 22:43:54 (GMT-4)

[+] Server: awselb/2.0
[+] The anti-clickjacking X-Frame-Options header is not present.
[+] The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
[+] The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
[+] Root page / redirects to: https://account.grammarly.com:443/
[+] No CGI Directories found (use '-C all' to force check all possible dirs)
[+] /wsmn/: Windows Remote Management is enabled
[+] /server-status: Apache server-status interface found (protected/forbidden)
[+] /server-info: Apache server-info interface found (protected/forbidden)
[+] 7868 requests: 3 error(s) and 6 item(s) reported on remote host
[+] End Time: 2021-04-30 23:23:11 (GMT-4) (2357 seconds)

[+] 1 host(s) tested
```

As the above figure shows, I found 3 errors within 7868 requests and those errors can be only consider as the info vulnerabilities since all of them were protected and forbidden under the states of the vulnerability.

As the second sub-domain I have scanned app.grammarly.com using the command of “**nikto -h app.grammarly.com**” as shown below.



```
(kali㉿kali)-[~] $ nikto -h app.grammarly.com
[+] Nikto v2.1.6
[+] Target IP: 23.22.231.245
[+] Target Hostname: app.grammarly.com
[+] Target Port: 80
[+] Message: Multiple IP addresses found: 23.22.231.245, 3.225.43.154, 52.202.227.13
[+] Start Time: 2021-04-30 23:35:34 (GMT-4)

[+] Server: awselb/2.0
[+] The anti-clickjacking X-Frame-Options header is not present.
[+] The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
[+] The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
[+] Root page / redirects to: https://app.grammarly.com:443/
[+] No CGI Directories found (use '-C all' to force check all possible dirs)
[+] 7865 requests: 1 error(s) and 3 item(s) reported on remote host
[+] End Time: 2021-05-01 00:12:42 (GMT-4) (2228 seconds)

[+] 1 host(s) tested
```

That scan also finished as the previous sub-domain scan with one error within all 7865 requests and it did not give any vulnerabilities apart from the previous scan results.

All the scan I have done using Nikto, we can find three vulnerabilities listed as “The **X-Content-Type-Options** header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type” and “The **X-XSS-Protection** header is not defined. This header can hint to the user agent to protect against some forms of XSS” and “Anti-clickjacking **X-Frame-options** header is not present”

i. **X-Content-Type-Options**

In order to reduce exposure to drive-by-download attacks and sites serving uploaded content they (organization) should set this header correctly.

a. Solution

Set the X-Content-Type-Options: nosniff header

ii. **X-XSS-Protection**

Entering data to a web application across an unstructured source will end up with Cross-Site-Scripting (XSS). This data which included in a dynamic content is sent to a web client apart from any filtering or validation of malicious code

a. Solution

Set the X-XSS-Protection "1; mode=block"

iii. **X-Frame-Options**

This X-Frame-Options header can be used to indicate whether or not a browser should be allowed to render a page in a `<frame>`, `<iframe>`, `<embed>` or `<object>`. This will be used by websites to prevent click-jacking attacks by ensuring that their content is not hidden in other websites.

a. Solution

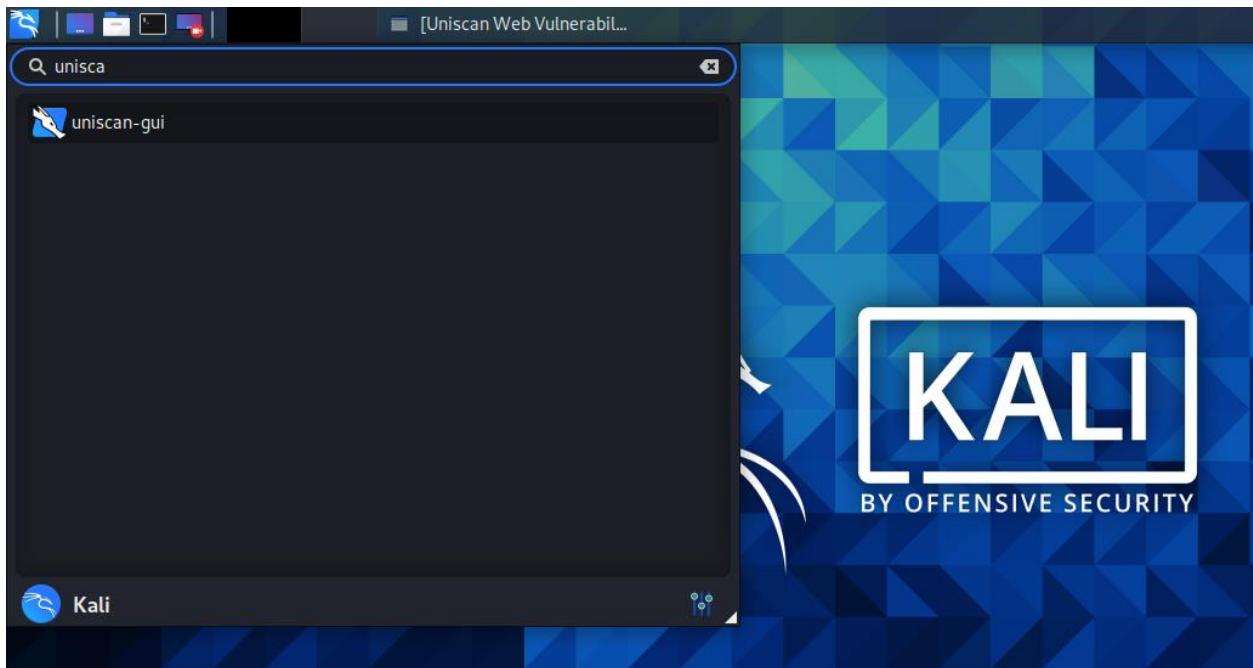
Set the X-Frame-Options: deny

II. Uniscan

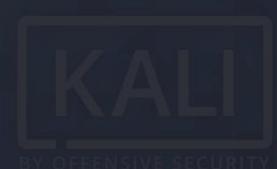
Uniscan is a simple web vulnerability scanner that searches for common flaws like neighboring document include, far off order execution, and far off record include flaws. It can also be used to label and list web administrations, interesting records and registries, and employee information. This instrument is written in Perl and can be used as a natural order line computer or as a graphical user interface. When the GUI and the command line are compared, the command line is quicker.

There is no need to install anything if you are using Kali Linux as your pen testing station because Uniscan is well-known for it.

We can run it by simply searching in Kali environment as shown below.



Using “**uniscan -h**” command we can show the options available with Uniscan tool.



```
kali@kali: ~
File Actions Edit View Help
[(kali㉿kali)-~]
$ sudo uniscan -h
[sudo] password for kali:
#####
# Uniscan project
# http://uniscan.sourceforge.net/
#####
V. 6.3
      Home   Burp Suite   Tools
OPTIONS:
-h    help
-u    <url> example: https://www.example.com/
-f    <file> list of url's
-b    Uniscan go to background
-q    Enable Directory checks
-w    Enable File checks
-e    Enable robots.txt and sitemap.xml check
-d    Enable Dynamic checks
-s    Enable Static checks
-r    Enable Stress checks
-i    <dork> Bing search
-o    <dork> Google search
-g    Web fingerprint
-gf   Server fingerprint
usage:
[1] perl ./uniscan.pl -u http://www.example.com/ -qweds
[2] perl ./uniscan.pl -f sites.txt -bqweds
[3] perl ./uniscan.pl -i uniscan
[4] perl ./uniscan.pl -i "ip:xxx.xxx.xxxx.xxxx"
[5] perl ./uniscan.pl -o "inurl:test"
[6] perl ./uniscan.pl -u https://www.example.com/ -r
└── andrea └── Command
```

In order to scan the main domain, we need to set the “-u” flag followed by the target domain as shown below and it will end up with giving some brief information about the server and the IP address of that target domain.

sudo uniscan -u https://www.grammarly.com



```
kali@kali: ~
File Actions Edit View Help
[(kali㉿kali)-~]
$ sudo uniscan -u https://www.grammarly.com
#####
# Uniscan project
# http://uniscan.sourceforge.net/
#####
V. 6.3

Scan date: 28-4-2021 2:59:09
Domain: https://www.grammarly.com/
IP: 3.208.221.30

Scan end date: 28-4-2021 2:59:16
PwnXSS ZAPreport Brav.com
HTML report saved in: report/www.grammarly.com.html
└── andrea └── Command
```

By using the “-q” flag, we can search for directories on the target. It appears that it discovered some directories including what appears to be documentation and PHP configuration data.

sudo uniscan -u https://www.grammarly.com -q



```
(kali㉿kali)-[~]
$ sudo uniscan -u https://www.grammarly.com -q
#####
# Uniscan project
# http://uniscan.sourceforge.net/
#####
V. 6.3

Scan date: 28-4-2021 3:3:3
Domain: https://www.grammarly.com/
IP: 3.208.221.30

Directory check:
[+] CODE: 200 URL: https://www.grammarly.com/about/
[+] CODE: 200 URL: https://www.grammarly.com/android/
[+] CODE: 200 URL: https://www.grammarly.com/bang/
[+] CODE: 200 URL: https://www.grammarly.com/blog/
[+] CODE: 200 URL: https://www.grammarly.com/boost/
[+] CODE: 200 URL: https://www.grammarly.com/box/
[+] CODE: 200 URL: https://www.grammarly.com/browsers/
[+] CODE: 200 URL: https://www.grammarly.com/careers/
[+] CODE: 200 URL: https://www.grammarly.com/chrome/
[+] CODE: 200 URL: https://www.grammarly.com/contact/
[+] CODE: 200 URL: https://www.grammarly.com/contacts/
[+] CODE: 200 URL: https://www.grammarly.com/download/
_____
Scan end date: 28-4-2021 3:12:47
Amass Linux - Amass
amd64 Command
HTML report saved in: report/www.grammarly.com.html
```

All the directories are in secured manner with code of 200 as shown above and we cannot find any vulnerable directory regarding the target domain.

Then I scanned domain by setting the flag with the options of “-wd” in order to find vulnerable information. Using -w flag allow to file check and gives some URLs, and which could yield to vulnerable and -d flag allow to conduct dynamic tests on the aim, such as email recognition, backdoor detection, and SQL and other injection point discovery. To end up this scan takes more time and in order to verify, I went through each of results, but they were not in vulnerable manner and can consider as informative result.

sudo uniscan -u https://www.grammarly.com -wd

```
kali@kali:~
```

```
File Actions Edit View Help
(kali㉿kali)-[~]
└─$ sudo uniscan -u https://www.grammarly.com -wd0
[sudo] password for kali:
Unknown option: n
#####
# Uniscan project      #
# http://uniscan.sourceforge.net/ #
#####
V. 6.3
Scan date: 28-4-2021 3:21:26
Domain: https://www.grammarly.com/
IP: 3.208.221.30

File check: ./WPPlugins
[+] CODE: 200 URL: https://www.grammarly.com/favicon.ico
[+] CODE: 200 URL: https://www.grammarly.com/index.html
[+] CODE: 200 URL: https://www.grammarly.com/index.php
[+] CODE: 200 URL: https://www.grammarly.com/robots.txt
[+] CODE: 200 URL: https://www.grammarly.com/sitemap.xml

Crawler Started:
Plugin name: FCKeditor upload test v.1 Loaded.
Plugin name: External Host Detect v.1.2 Loaded.
Plugin name: Timthumb < 1.32 vulnerability v.1 Loaded.
Plugin name: phpinfo() Disclosure v.1 Loaded.
Plugin name: E-mail Detection v.1.1 Loaded.
Plugin name: Web Backdoor Disclosure v.1.1 Loaded.
Plugin name: Code Disclosure v.1.1 Loaded.
Plugin name: Upload Form Detect v.1.1 Loaded.
[+] Crawling finished, 1354 URL's found!
```

```
kali@kali:~
```

```
File Actions Edit View Help
External hosts:
[+] External Host Found: https://www.thesun.co.uk
[+] External Host Found: https://www.ted.com
[+] External Host Found: https://guides.skylinecollege.edu
[+] External Host Found: https://t.co
[+] External Host Found: https://www.forbes.com
[+] External Host Found: https://giphy.com
[+] External Host Found: http://www.bbc.com
[+] External Host Found: https://www.jobacle.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 159.
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 161.
| [+] External Host Found: http://www.proofreadinglondon.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 496.
| [+] External Host Found: http://emilypost.com
| [+] External Host Found: https://www.themuse.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 159.
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 161.
| [+] External Host Found: http://fivethirtyeight.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 496.
| [+] External Host Found: https://app.appsflyer.com
| [+] External Host Found: http://nymag.com
| [+] External Host Found: https://www.twowanderingsoles.com
| [+] External Host Found: https://jeffmagnusonconsulting.com
| [+] External Host Found: http://www.newyorker.com
| [+] External Host Found: http://grammarly.com
| [+] External Host Found: https://www.youtube.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 159.
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 161.
| [+] External Host Found: http://mentalflloss.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 496.
| [+] External Host Found: https://owl.english.purdue.edu
| [+] External Host Found: https://www.mindtools.com
| [+] External Host Found: https://www.kaplaninternational.com
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 159.
Wide character in print at /usr/share/uniscan/Uniscan/Functions.pm line 161.
```

```
kali@kali: ~
```

```
File Actions Edit View Help
```

```
E-mails:  
[+] E-mail Found: timeline@2x.png,  
[+] E-mail Found: instagram@2x.png  
[+] E-mail Found: support@grammarly.com  
[+] E-mail Found: marketing@2x.png  
[+] E-mail Found: upport@2x.png  
[+] E-mail Found: comparably_outlook@2x.png  
[+] E-mail Found: accesible@2x.png  
[+] E-mail Found: updates@2x.png  
[+] E-mail Found: privacy@grammarly.com  
[+] E-mail Found: facebook@2x.png  
[+] E-mail Found: comparably_outlook@2x.png  
[+] E-mail Found: icon-checkmark@2x.png  
[+] E-mail Found: desktop@2x.png  
[+] E-mail Found: reg@2x.png  
[+] E-mail Found: f3@2x.png  
[+] E-mail Found: jose@2x.png  
[+] E-mail Found: demo_doc_screenshot@2x.png  
[+] E-mail Found: utton@2x.png  
[+] E-mail Found: hiring@domain.com  
[+] E-mail Found: video@2x.png  
[+] E-mail Found: linkedin@2x.png  
[+] E-mail Found: larity@2x.png  
[+] E-mail Found: ames@2x.jpg  
[+] E-mail Found: illinois@2x.png  
[+] E-mail Found: infrastructure@2x.png  
[+] E-mail Found: utton@2x-70af3377-c572-4543-9850-3154da31f82f.png  
[+] E-mail Found: cta@2x.jpg  
[+] E-mail Found: icle@2x.png  
[+] E-mail Found: teaser-poster@2x.jpg  
[+] E-mail Found: browser-screenshot@2x.jpg  
[+] E-mail Found: names_matter_mobile_2@2x.png  
[+] E-mail Found: browser-screenshot2@2x.jpg  
[+] E-mail Found: sf@2x.jpg  
[+] E-mail Found: proof@2x.jpg
```

```
[Uniscan Report - Mozilla... kali@kali: /usr/share/uniscan...]
```

```
File Actions Edit View Help
```

```
Web Backdoors:  
[+] File System: /tmp/uni-labs
```

```
Source Code Disclosure:
```

```
File Upload Forms:  
[+] Upload Form Found: https://www.grammarly.com/plagiarism-checker
```

```
Ignored Files:  
https://www.grammarly.com/blog/wp-content/uploads/2014/09/scrivener-logo.tif.
```

```
Dynamic tests:  
plugin name: Learning New Directories v.1.2 Loaded.  
plugin name: FCKeditor tests v.1.1 Loaded.  
plugin name: Timthumb < 1.32 vulnerability v.1 Loaded.  
Plugin name: Find Backup Files v.1.2 Loaded.  
Plugin name: Blind SQL-injection tests v.1.3 Loaded.  
Plugin name: Local File Include tests v.1.1 Loaded.  
Plugin name: PHP CGI Argument Injection v.1.1 Loaded.  
Plugin name: Remote Command Execution tests v.1.1 Loaded.  
Plugin name: Remote File Include tests v.1.2 Loaded.  
Plugin name: SQL-injection tests v.1.2 Loaded.  
Plugin name: Cross-Site Scripting tests v.1.2 Loaded.  
Plugin name: Web Shell Finder v.1.3 Loaded.  
[+] 26 New directories added
```

```
subfinder - Brute
```

```
FCKeditor tests:
```

```
Timthumb < 1.33 vulnerability:
```

```
Backup Files: Amass
```

```
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/category.bkp  
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/mobile.bkp
```

```
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/business-writing-tools.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/digital-communication-strategies.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/leading-web-hosting-platform-case-study.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/how-technology-improves-customer-service.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/how-consistency-is-critical-to-work-communication.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/customer-experience.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/hr.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/marketing.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/business-writing-vocabulary.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/5internal-business-communication-best-practices.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/best-practices-for-customer-communication.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/business-communication-with-customers.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/part-1-how-grammarly-tackles-hidden-complexity-in-front-end-applications.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/declarative-programming-net-framework.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/part-2-how-grammarly-tackles-hidden-complexity-in-front-end-applications.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/swiftui-uikit.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/building-microsoft-word-add-in-mac.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/the-value-of-live-agents.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/new-standard-for-customer-communication-ebook.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/author/vitalii-braslavskyi.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/author/bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/author/vgaidylo.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/author/victor-pavlychko.bkp
[+] CODE: 200 URL: https://www.grammarly.com/blog/engineering/author/grammarly.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/frostandsullivan-case-study.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/technology-customer-experience-goals.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/customer-trust-consistency.bkp
[+] CODE: 200 URL: https://www.grammarly.com/business/learn/upskill-more-efficient-cx-teams.bkp

Blind SQL Injection:
Local File Include:
[*] Remaining tests: 24847
```

Uniscan also saves each scan as an HTML file under **/usr/share/uniscan/report/** directory and we can simply examine the results at a later time in graphical manner.

```
[kali㉿kali)-[/usr/share/uniscan/report]
└─$ ls
brave.com.html css.css  images  index.php  uniscan.html  www.grammarly.com.html

[kali㉿kali)-[/usr/share/uniscan/report]
└─$ firefox uniscan.html
```

Uniscan Report - Mozilla Firefox

Uniscan Report - Mozilla... kali:kali:/usr/share/uniscan...

05:38 AM

Uniscan Report

Uniscan Report

file:///usr/share/uniscan/report/uniscan.html

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFU Gmail Uniscan Report

 Uniscan
Web Vulnerability Scanner

SCAN TIME

Scan Started: 28/4/2021 3:21:26

TARGET

Domain https://www.grammarly.com/
Target IP: 3.208.221.30

CRAWLING

File check:

CODE: 200 URL: https://www.grammarly.com/favicon.ico
CODE: 200 URL: https://www.grammarly.com/index.html
CODE: 200 URL: https://www.grammarly.com/index.php
CODE: 200 URL: https://www.grammarly.com/robots.txt
CODE: 200 URL: https://www.grammarly.com/sitemap.xml

Crawling finished, found: 1354 URL's

CKEditor File Upload:

External hosts:

- https://www.theguardian.co.uk
- https://www.ted.com
- https://guides.skylinecollege.edu
- https://it.co
- https://www.forbes.com
- https://giphy.com
- http://www.bbc.com
- https://www.jobacle.com
- http://www.proofreadinglondon.com
- http://jemilypost.com
- https://www.themuse.com
- http://www.wetpixelight.com
- https://futuraplayer.com
- http://nymag.com
- https://www.twowanderingsoles.com
- https://jeffmagnusonconsulting.com
- http://www.newyorker.com
- http://grammarly.com
- https://www.youtube.com
- http://mentalfloss.com
- https://owl.english.purdue.edu
- https://www.mindtools.com
- http://www.koreaninternational.com
- http://www.countryliving.com
- https://stanhandicrafts.com
- https://martechseries.com
- https://twitter.com
- https://www.cv-library.co.uk
- https://wridea.com
- https://vanstartupweek.ca
- http://personal.lse.ac.uk
- https://pdfs.semanticscholar.org
- https://blog.hubspot.com
- https://www.naplesnews.com

Timthumb:

PHPInfo() Disclosure:

Web Backdoors:

Source Code Disclosure:

File Upload Forms:

Upload Form Found: https://www.grammarly.com/plagiarism-checker

Ignored Files:

https://www.grammarly.com/blog/wp-content/uploads/2014/09/scrivener-logo.tif.

DYNAMIC TESTS

Learning New Directories: 26 New directories added.

CKEditor tests:

Timthumb < 1.33 vulnerability:

Backup Files:

```

CODE: 200 URL: https://www.grammarly.com/blog/engineering/category.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/category/mobile.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/category/ai.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/category/product.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/category/mobile/ai.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/product-mindset-for-engineers.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/category/infrastructure.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/gec-tag-not-rewrite.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/nlp-building-future-communication.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/announcing-ua-gec.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/adversarial-grammatical-error-correction.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/accepting-multiple-suggestions.bkp
CODE: 200 URL: https://www.grammarly.com/blog/engineering/product-oriented-eng-management.bkp
CODE: 200 URL: https://www.grammarly.com/business/learn/professional-writing-skills.bkp
CODE: 200 URL: https://www.grammarly.com/business/learn/business-writing-tools.bkp
CODE: 200 URL: https://www.grammarly.com/business/learn/digital-communication-strategies.bkp
CODE: 200 URL: https://www.grammarly.com/business/learn/lexicon-web-hosting-nla/fnrm-case-study.hkn

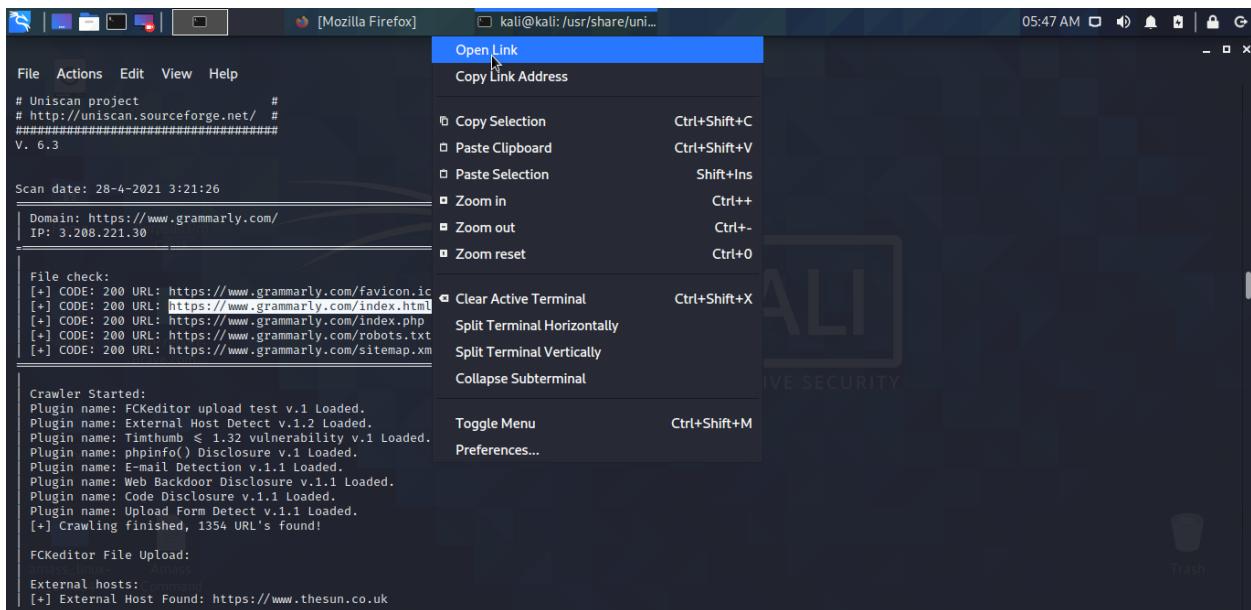
```

Blind SQL Injection:

Local File Include:

As shown above figures, we did not find any Web Backdoors, Source file disclosure, PHP info disclosure details and Blind SQL injection vulnerabilities. With the help of this tool even we found some external host addresses, e-mail addresses and backup files; all of them are informative. Under the crawling, it scans for files and all of those files are in secured manner with the code of 200 as shown in the figure.

Furthermore, I have checked some outputted URL under the file check category as shown below, and even it shows the path of the link as “index.html”, it simply redirected to the main domain of grammarly.com when it opens through the browser. Steps are shown below.



```
# Uniscan project
# http://uniscan.sourceforge.net/
#####
V. 6.3

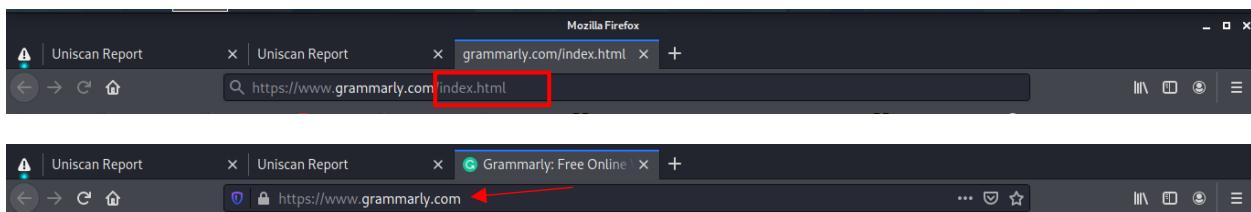
Scan date: 28-4-2021 3:21:26
Domain: https://www.grammarly.com/
IP: 3.208.221.30

File check:
[+] CODE: 200 URL: https://www.grammarly.com/favicon.ico
[+] CODE: 200 URL: https://www.grammarly.com/index.html
[+] CODE: 200 URL: https://www.grammarly.com/index.php
[+] CODE: 200 URL: https://www.grammarly.com/robots.txt
[+] CODE: 200 URL: https://www.grammarly.com/sitemap.xml

Crawler Started:
Plugin name: FCKeditor upload test v.1 Loaded.
Plugin name: External Host Detect v.1.2 Loaded.
Plugin name: Timthumb < 1.32 vulnerability v.1 Loaded.
Plugin name: phpinfo() Disclosure v.1 Loaded.
Plugin name: E-mail Detection v.1.1 Loaded.
Plugin name: Web Backdoor Disclosure v.1.1 Loaded.
Plugin name: Code Disclosure v.1.1 Loaded.
Plugin name: Upload Form Detect v.1.1 Loaded.
[+] Crawling finished, 1354 URL's found!

FCKeditor File Upload:

External hosts:
[+] External Host Found: https://www.thesun.co.uk
```



Using that tool, I was able to find a file uploading form under the File Upload Forms checking category and it is a simple plagiarism checking form which belongs to the same domain and that also can be considered as an informational result.

| File Upload Forms:

| [+] Upload Form Found: <https://www.grammarly.com/plagiarism-checker>

A screenshot of a Kali Linux desktop environment. A Mozilla Firefox window is open, showing the Grammarly Plagiarism Checker page at <https://www.grammarly.com/plagiarism-checker>. The page title is "Plagiarism Checker by Grammarly". Below it, a sub-headline reads: "Grammarly's plagiarism checker detects plagiarism in your text and checks for other writing issues." There is a central input field with placeholder text "Enter text or upload file to check for plagiarism and writing errors." To the left of the input field is a magnifying glass icon over a document, and to the right is a document icon with a checkmark. At the bottom of the page, there is a blue banner with the text "Catch plagiarism from 100+ sources" on the left, "Write better with Grammarly's app." in the center, a "Add to Firefox It's free" button, and "Get feedback on grammar, punctuation, and spelling" on the right.

Also, there was an ignored URL under the Ignored File category and as it says that page is not exist now. There was only a direction to the main domain as previous.

| Ignored Files:

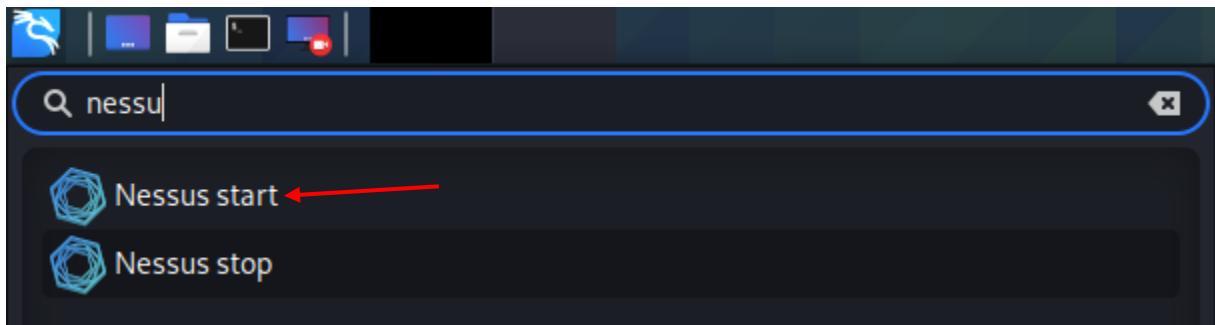
| <https://www.grammarly.com/blog/wp-content/uploads/2014/09/scrivener-logo.tif>.

A screenshot of a Kali Linux desktop environment. A Mozilla Firefox window is open, showing a 404 error page for the URL <https://www.grammarly.com/blog/wp-content/uploads/2014/09/scrivener-logo.tif>. The page features a cartoon illustration of a blue character holding a sign that says "404". Below the illustration, the text "Sorry, this page doesn't exist anymore." is displayed. Further down, a message reads: "But never fear! You can still find a lot of useful writing tips on the Grammarly Blog." A "Visit Grammarly Blog" button is present. The top of the browser window shows the Grammarly blog header and navigation links.

III. Nessus

Nessus is one of the security scanners used in vulnerability tests and penetration testing, as well as malware attacks. Plug-ins are used by the Nessus scanning engine to find new vulnerabilities. Within 24 hours of a flaw being public, Tenable distributes plug-ins with the most up-to-date information to customers' systems. Customers get regular plug-in feeds to keep current and new bugs occur virtually every day. While Nessus does not have penetration testing capability, administrators can use the results of Nessus scans in conjunction with common penetration testing tools like Metasploit, Core IMPACT, and Immunity CANVAS to gain insight into risk.

The downloadable installer can be found for Linux-based systems in the Nessus official webpage, and we need to install it using terminal. After the install we need to start Nessus daemon by terminal or simply searching in Kali environment as shown below.



After login to the Nessus with valid credentials, we can start our scans as we prefer. Then I was done Web Application Tests scan to my target domain by filling the relevant information as shown below. We need to enter our target domain IP separately as the target or we can enter several IPs regarding to our target domain.

First, I have done Web Application Test separate scan regarding my target domain as shown below.

The screenshot shows the Nessus Essentials interface for creating a new scan. The left sidebar includes sections for FOLDERS (My Scans, All Scans, Trash), RESOURCES (Policies, Plugin Rules), and TENABLE (Tenable News). The main area is titled "New Scan / Web Application Tests" and contains tabs for Settings, Credentials, and Plugins. Under the Settings tab, the "BASIC" section is selected, showing fields for Name ("grammarly web scan - (Report)"), Description ("scan for web audit report"), Folder ("My Scans"), and Targets ("3.208.221.80").

The screenshot shows the results of the "grammarly web scan - (Report)". The interface includes a sidebar with FOLDERS, RESOURCES, and TENABLE sections. The main content area displays the scan results for one host (3.208.221.80) with one vulnerability found. A "Scan Details" panel provides information about the scan: Policy: Web Application Tests, Status: Running, Severity Base: CVSS v3.0, Scanner: Local Scanner, and Start: Today at 10:30 PM. A "Vulnerabilities" section shows a pie chart with three segments: Critical (red), High (orange), and Medium (yellow).

grammarly web scan - (Report)

Scan Details

Policy:	Web Application Tests
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 10:30 PM
End:	Today at 10:36 PM
Elapsed:	6 minutes

Vulnerabilities

It was completed within 6 minutes and it did not give any low, medium, high, and critical vulnerabilities. It only gave one vulnerability which falls under the info category as shown below.

Sev	Name	Family	Count
INFO	Nessus SYN scanner	Port scanners	1

Scan Details

Policy:	Web Application Tests
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 10:30 PM
End:	Today at 10:36 PM
Elapsed:	6 minutes

The screenshot shows the Nessus Essentials interface. On the left, there's a sidebar with 'Folders' (My Scans, All Scans, Trash), 'Resources' (Policies, Plugin Rules), and 'Tenable' (Community, Research, Plugin Release Notes). The main area displays a 'grammarly web scan - (Report) / Plugin #11219'. It includes sections for 'Description' (SYN scanner), 'Solution' (protect target with IP filter), 'Output' (listing port 25/tcp as open), 'Plugin Details' (Severity: Info, ID: 11219, Version: 1.37, Type: remote, Family: Port scanners, Published: February 4, 2009, Modified: April 20, 2021), and 'Risk Information' (Risk Factor: None). Below this, another screenshot shows a scan report for 'grammarly web scan - (Report)' with an 'Assessed Threat Level: None' and 'Scan Details' (Policy: Web Application Tests, Status: Completed, Severity Base: CVSS v3.0, Scanner: Local Scanner, Start: Today at 10:30 PM, End: Today at 10:36 PM, Elapsed: 6 minutes). The 'VPR Top Threats' tab indicates 'No prioritized vulnerabilities found.'

As shown above figure, no vulnerabilities have been found as prioritized by Tenable's patented Vulnerability Priority Rating (VPR) system and Assessed Treat Level was none.

Then I separately scanned one of sub-domain under my target domain, but it took more time to finished than the previous scan of main-domain. My target sub-domain was, **account.grammarly.com**.

account.grammarly.com

Scan Details

Policy:	Web Application Tests
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 10:47 PM
End:	Today at 11:25 PM
Elapsed:	38 minutes

Vulnerabilities

Sev	Name	Family	Count
INFO	HTTP (Multiple Issues)	Web Servers	7
INFO	Web Server (Multiple Issues)	Web Servers	3
INFO	Nessus SYN scanner	Port scanners	3
INFO	Nessus Scan Information	Settings	1

Scan Details

Policy:	Web Application Tests
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 10:47 PM
End:	Today at 11:25 PM
Elapsed:	38 minutes

Even that scan end up with giving several vulnerabilities, all of them were under the info category and we could not find any low, medium, high, or critical vulnerabilities as the previous scan.

nessus
Essentials

Scans Settings

account.grammarly.com / HTTP (Multiple Issues)

[Back to Vulnerabilities](#)

Hosts 1 Vulnerabilities 4 VPR Top Threats 0 History 1

Search Vulnerabilities 4 Vulnerabilities

Sev	Name	Family	Count	Actions
INFO	HTTP Methods Allowed (per directory)	Web Servers	2	Edit Audit
INFO	HTTP Server Type and Version	Web Servers	2	Edit Audit
INFO	HyperText Transfer Protocol (HTTP) In...	Web Servers	2	Edit Audit
INFO	HSTS Missing From HTTPS Server	Web Servers	1	Edit Audit

nessus
Essentials

Scans Settings

account.grammarly.com / Web Server (Multiple Issues)

[Back to Vulnerabilities](#)

Hosts 1 Vulnerabilities 4 VPR Top Threats 0 History 1

Search Vulnerabilities 2 Vulnerabilities

Sev	Name	Family	Count	Actions
INFO	Web Server No 404 Error Code Check	Web Servers	2	Edit Audit
INFO	Web Server Crafted Request Vendor/...	Web Servers	1	Edit Audit

Description

The remote web server is configured such that it does not return '404 Not Found' error codes when a nonexistent file is requested, perhaps returning instead a site map, search page or authentication page.

Nessus has enabled some counter measures for this. However, they might be insufficient. If a great number of security holes are produced for this port, they might not all be accurate.

Output

```
CGI scanning will be disabled for this host because the host responds
to requests for non-existent URLs with HTTP code 301
rather than 404. The requested URL was :

http://ec2-34-230-231-182.compute-1.amazonaws.com/nmp1WpyeXTPw.html
```

Port	Hosts
80 / tcp / www	34.230.231.182

Description

The web server running on the remote host appears to be hiding its version or name, which is a good thing. However, using a specially crafted request, Nessus was able to discover the information.

Solution

No generic solution is known. Contact your vendor for a fix or a workaround.

Output

```
After sending this request :
HELP

Nessus was able to gather the following information from the web server :
awselb/2.0
```

Port	Hosts
443 / tcp / www	34.230.231.182

Above figures shows the result of sub-domain info vulnerabilities and the descriptions and the solutions which were already given by the Nessus tool.

Furthermore, I have done another Nessus web application test regarding the sub-domain of app.grammarly.com as previous sub-domain scan since both of those sub-domains fallen under the scope of bug-bounty rules and regulation which were given by the Grammarly.

Severity	Count
Critical	0
High	0
Medium	0
Low	0
Info	9

Description
By calling the OPTIONS method, it is possible to determine which HTTP methods are allowed on each directory.

The following HTTP methods are considered insecure:
PUT, DELETE, CONNECT, TRACE, HEAD

Many frameworks and languages treat 'HEAD' as a 'GET' request, albeit one without any body in the response. If a security constraint was set on 'GET' requests such that only 'authenticatedUsers' could access GET requests for a particular servlet or resource, it would be bypassed for the 'HEAD' version. This allowed unauthorized blind submission of any privileged GET request.

As this list may be incomplete, the plugin also tests - if 'Thorough tests' are enabled or 'Enable web applications tests' is set to 'yes' in the scan policy - various known HTTP methods on each directory and considers them as unsupported if it receives a response code of 400, 403, 405, or 501.

Note that the plugin output is only informational and does not necessarily indicate the presence of any security vulnerabilities.

See Also

- <http://www.nessus.org/u?d9c03a9a>
- <http://www.nessus.org/u?b019cbdb>
- [https://www.owasp.org/index.php/Test_HTTP_Methods_\(OTG-CONFIG-006\)](https://www.owasp.org/index.php/Test_HTTP_Methods_(OTG-CONFIG-006))

As previous two scans, this was also end up with several vulnerabilities which were fallen under the info category. Even though I have done several no of

separate scans regarding my target domain and its sub-domains, I could not find any low, medium, high, or critical vulnerabilities.

The below figure shows all the scans including main-domain and its sub-domains using Tenable's Nessus tool.

The screenshot shows the Nessus Essentials web interface. The left sidebar has sections for FOLDERS (My Scans, All Scans, Trash), RESOURCES (Policies, Plugin Rules), and TENABLE (Tenable News). The main area is titled 'My Scans' and shows a table with three entries:

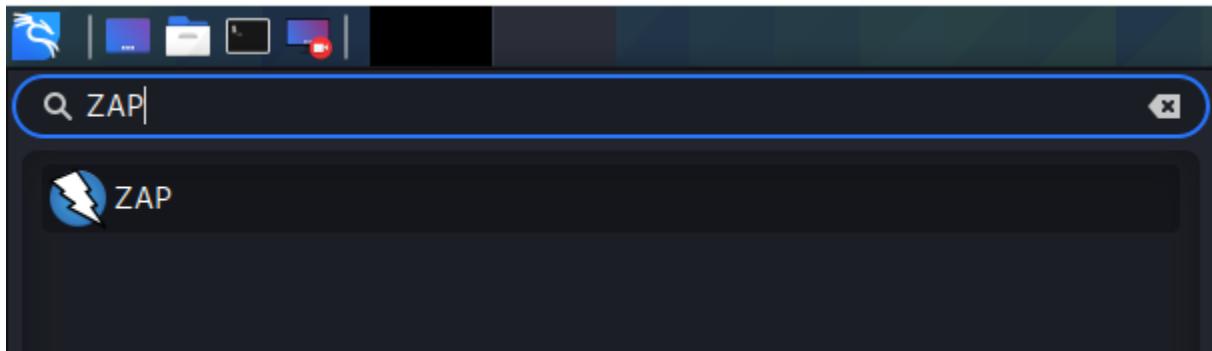
Name	Schedule	Last Modified
app.grammarly.com	On Demand	Today at 12:12 AM
account.grammarly.com	On Demand	April 30 at 11:25 PM
grammarly web scan - (Report)	On Demand	April 30 at 10:36 PM

The first two rows are highlighted with a yellow box, and the third row is highlighted with a red box.

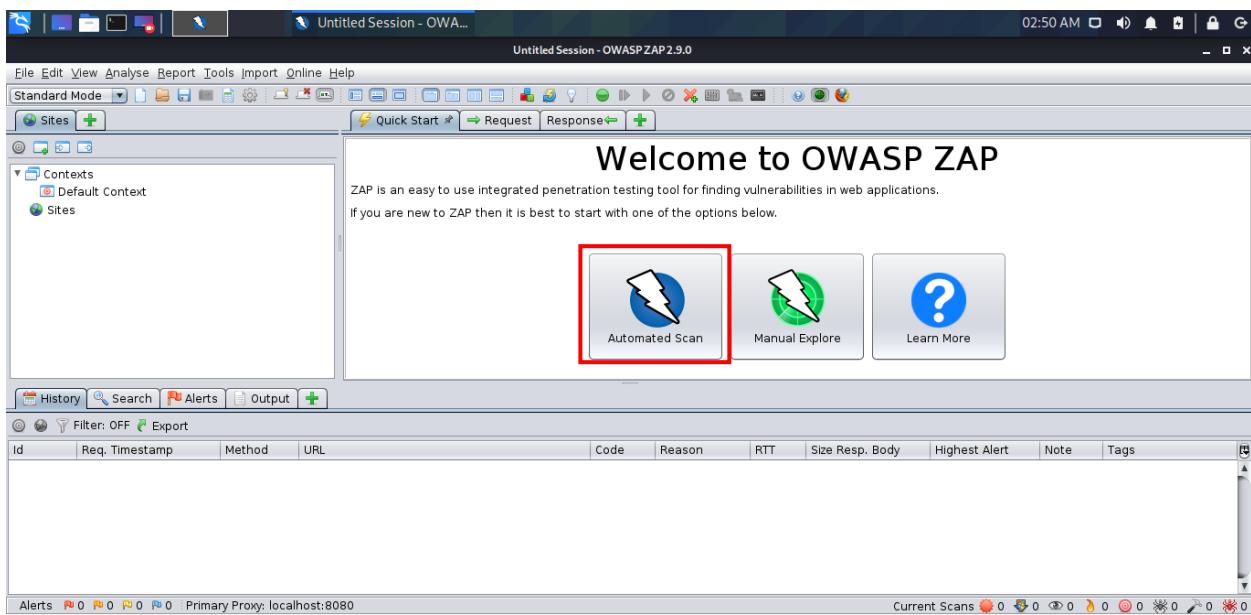
III. OWASP Zap

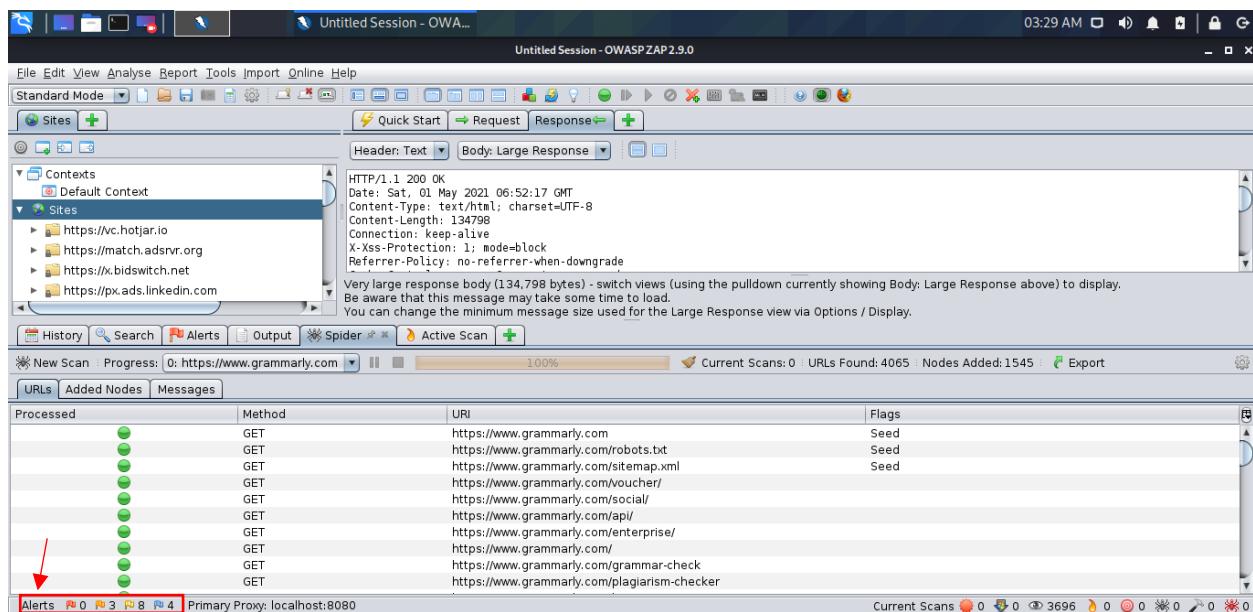
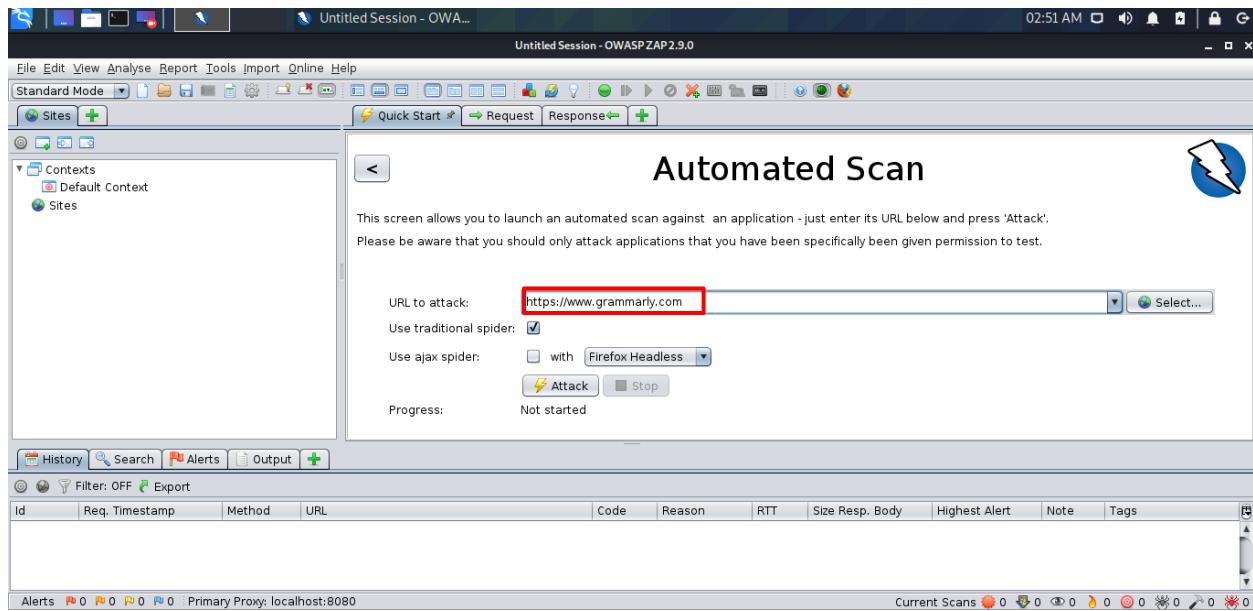
One of the most widely used web application security monitoring techniques is the OWASP Zed Attack Proxy (ZAP). OWASP contributes to and maintains it as an open source project, which is available for free. The Open Web Application Security Project (OWASP) is a vendor-neutral, non-profit organization of people devoted to improving the security of web applications. Web application security testers may use the OWASP Zed Attack Proxy, a Java-based platform with an elegant graphical GUI, to perform fuzzing, scripting, spidering, and proxying in order to attack web applications. Since it is a Java tool, it can operate on almost every operating system that supports Java. ZAP is used in the Kali Linux Penetration Testing OS by default, but it can also be downloaded and installed on any OS that has Java installed.

Since ZAP is used in the Kali Linux Penetration Testing OS by default, it can be accessed by simply searching and selecting the ZAP in the search bar in Kali Linux as shown below.



First, I have selected the scan type as automated and enter my target domain's URL as shown below.





After several hours of scanning the target domain, it was end up with finding 3 medium, 8 low and 4 informational priority alerts as shown in below figures.

i. 3 medium priority vulnerabilities:

1. CSP Scanner: Wildcard Directive (1234)

The screenshot shows the OWASP ZAP 2.9.0 interface. In the 'Alerts' tab, there is one alert titled 'CSP Scanner: Wildcard Directive (1234)' which is highlighted with a yellow box. This alert is categorized under 'Content-Security-Policy'. Below the alert, a detailed description is provided, also enclosed in a yellow box. At the bottom of the 'Alerts' tab, there is a red arrow pointing to the page number '4' in the navigation bar.

CSP Scanner: Wildcard Directive (1234)

URL: https://www.grammarly.com
Risk: Medium
Confidence: Medium
Parameter: Content-Security-Policy
Attack:
Evidence: frame-ancestors 'self' *.grammarly.com
CWE ID: 16
WASC ID: 15
Source: Passive (10055 - CSP Scanner)
Description:
The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined:
script-src, script-src-elem, script-src-attr, style-src, style-src-elem, style-src-attr, img-src, connect-src, frame-src, font-src, media-src, object-src, manifest-src, worker-src, prefetch-src

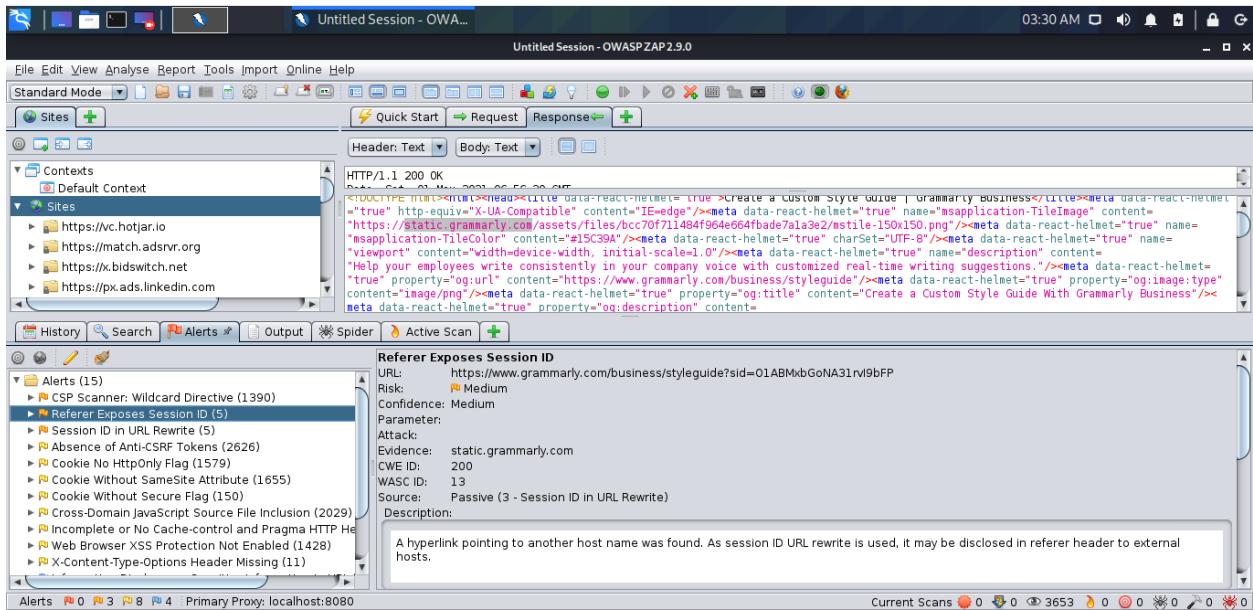
Alerts 0 1 2 3 4 Primary Proxy: localhost:8080

As the figure shows, this vulnerability was in under the medium priority alert with the Content-Security-Policy (CSP) parameter.

Solution:

As the solution, grammarly need to configure the CSP header properly with the web server, application server, load balancer, etc.

2. Referer Exposes Session ID (5)

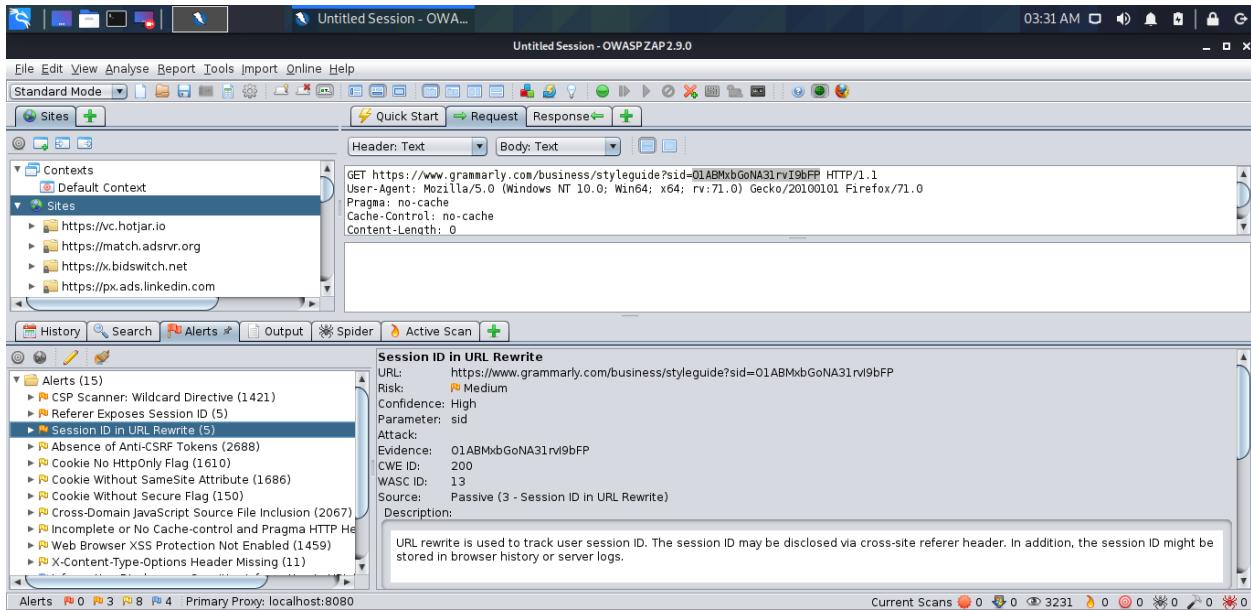


Here we can see that a hyperlink which was pointing to another host was reviled and this might be disclosed in referer header to external hosts as shown above figure. This vulnerability is very sensitive when this session ID and the hyperlink refers to an external or third party host.

Solution:

In order to ensure security, session ID need to include in a secured session cookie.

3. Session ID in URL Rewrite (5)



The user session ID is tracked using URL rewrite. Cross-site referer headers can reveal the session ID. The session ID can also be saved in the browser history or server logs.

Solution:

In order to ensure security, need to include the session ID in a cookie for protected material. Furthermore, site can be more safe with combination of cookie and URL rewrite.

ii. Other low and informational priority vulnerabilities:

1. Absense of Anti-CRSF tokens (2688)

The screenshot shows the OWASP ZAP 2.9.0 interface with an 'Untitled Session - OWA...' tab open. In the 'Alerts' panel, under the 'Alerts (15)' section, the 'Absence of Anti-CSRF Tokens (2688)' item is selected. The details pane displays the following information:

- Evidence: <form id="file-upload-form" enctype="multipart/form-data">
- CWE ID: 352
- WASC ID: 9
- Source: Passive (10202 - Absence of Anti-CSRF Tokens)
- Description: No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URLs or forms.
- Other Info: No known Anti-CSRF token (anticsrf, CSRFToken, _RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret) was found in the following HTML form: [Form 1: "source_file"].
- Solution: Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

At the bottom, the status bar shows 'Current Scans 0 0 0 2494 0 0 0 0 0 0 0 0'.

2. Information Disclosure - Sensitive Information in URL (491)

The screenshot shows the OWASP ZAP 2.9.0 interface with an 'Untitled Session - OWA...' tab open. In the 'Alerts' panel, under the 'Alerts (15)' section, the 'Information Disclosure - Sensitive Information in URL (491)' item is selected. The details pane displays the following information:

- Evidence: foo-bar@example.com
- CWE ID: 200
- WASC ID: 13
- Source: Passive (10024 - Information Disclosure - Sensitive Information in URL)
- Description: The request appeared to contain sensitive information leaked in the URL. This can violate PCI and most organizational compliance policies. You can configure the list of strings for this check to add or remove values specific to your environment.
- Other Info: The URL contains email address(es).
- Solution: Do not pass sensitive information in URIs.

At the bottom, the status bar shows 'Current Scans 0 0 0 2415 0 0 0 0 0 0 0 0'.

Sent Messages Filtered Messages										
Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	
6,275	5/1/21, 3:17:34 AM	5/1/21, 3:17:36 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	946 ms	1,229 bytes	52,034 bytes	
6,276	5/1/21, 3:17:34 AM	5/1/21, 3:17:36 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	589 ms	1,229 bytes	55,274 bytes	
6,277	5/1/21, 3:17:35 AM	5/1/21, 3:17:36 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	674 ms	1,229 bytes	52,034 bytes	
6,278	5/1/21, 3:17:36 AM	5/1/21, 3:17:36 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	610 ms	1,229 bytes	55,274 bytes	
6,279	5/1/21, 3:17:36 AM	5/1/21, 3:17:37 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	643 ms	1,229 bytes	52,034 bytes	
6,280	5/1/21, 3:17:36 AM	5/1/21, 3:17:37 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	582 ms	1,229 bytes	55,274 bytes	
6,281	5/1/21, 3:17:36 AM	5/1/21, 3:17:37 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	651 ms	1,229 bytes	52,034 bytes	
6,282	5/1/21, 3:17:37 AM	5/1/21, 3:17:38 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	782 ms	1,229 bytes	55,274 bytes	
6,283	5/1/21, 3:17:37 AM	5/1/21, 3:17:38 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	683 ms	1,229 bytes	52,034 bytes	
6,284	5/1/21, 3:17:38 AM	5/1/21, 3:17:38 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	587 ms	1,229 bytes	55,274 bytes	
6,285	5/1/21, 3:17:38 AM	5/1/21, 3:17:39 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	643 ms	1,229 bytes	52,034 bytes	
6,286	5/1/21, 3:17:38 AM	5/1/21, 3:17:39 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	589 ms	1,229 bytes	55,274 bytes	
6,287	5/1/21, 3:17:39 AM	5/1/21, 3:17:39 AM	GET	https://www.grammarly.com/blog/connect-with-c...	200	OK	645 ms	1,229 bytes	52,034 bytes	
6,288	5/1/21, 3:17:39 AM	5/1/21, 3:17:39 AM	GET	https://www.grammarly.com/blog/conjunctions-f...	200	OK	596 ms	1,229 bytes	55,274 bytes	
6,289	5/1/21, 3:17:40 AM	5/1/21, 3:17:40 AM	GET	https://www.grammarly.com/blog/connecting-se...	200	OK	381 ms	1,229 bytes	49,988 bytes	

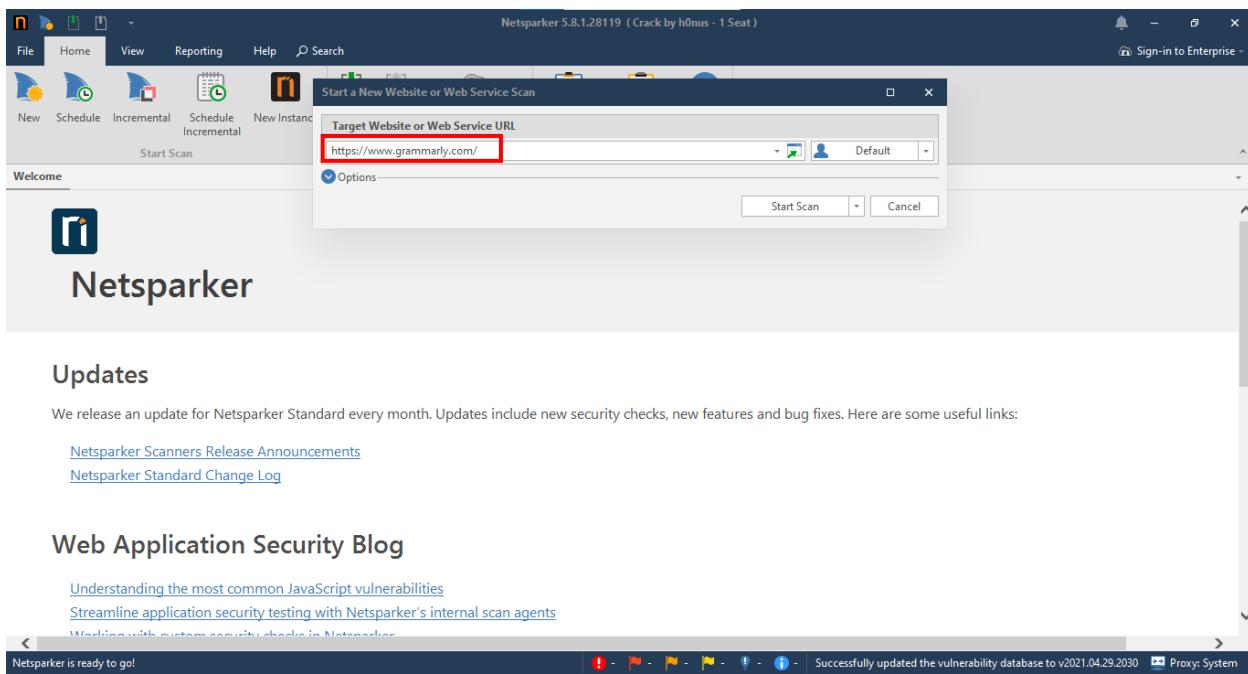
Finally, as I mentioned above, using this OWASP ZAP tool I was able to find 3 medium, 8 low and 4 informational priority vulnerabilities. In order to ensure the security, server should configure those headers and other solutions regarding those vulnerabilities securely.

V. Netsparker Professional

Netsparker is an Enterprise DAST (Dynamic Application Security Testing) tool that allows you to search websites, web applications, and web services for security vulnerabilities in an automatic and completely configurable manner. Netsparker is extremely extensible since it can search any kind of web app, regardless of the medium or language used to create it. Netsparker uses "Proof-Based Scanning," which instantly verifies discovered bugs and determines if they are false positives by exploiting them in a secure, read-only fashion.

I used Netsparker Professional Version to scan my target domain and I was able to get detailed report and more vulnerabilities than the other tools. I installed in on my Windows environment, and it is quite easy to work on there.

After installed it and starts it, first we need to enter our target website or web service URL in the given pop-up box as shown below.



Updates

We release an update for Netsparker Standard every month. Updates include new security checks, new features and bug fixes. Here are some useful links:

[Netsparker Scanners Release Announcements](#)

[Netsparker Standard Change Log](#)

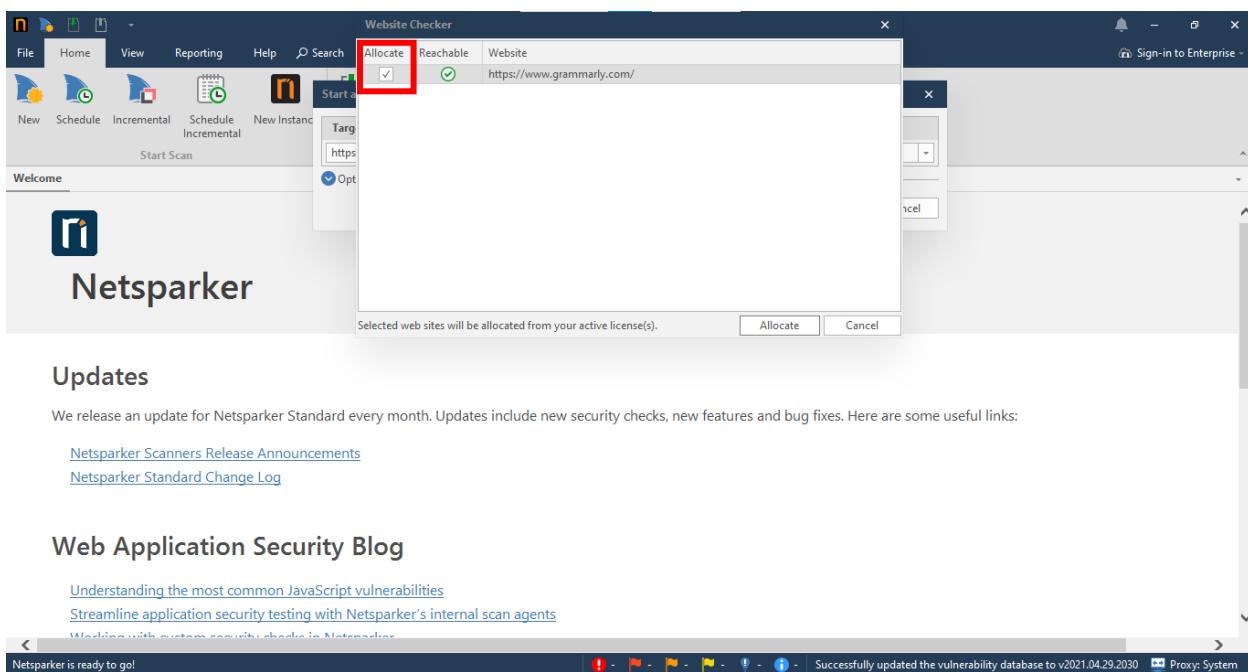
Web Application Security Blog

[Understanding the most common JavaScript vulnerabilities](#)

[Streamline application security testing with Netsparker's internal scan agents](#)

[Working with custom security checks in Netsparker](#)

Then it will be asked to allocate that target webpage under my active license, and we need to allocate it by simply ticking the check box.



Updates

We release an update for Netsparker Standard every month. Updates include new security checks, new features and bug fixes. Here are some useful links:

[Netsparker Scanners Release Announcements](#)

[Netsparker Standard Change Log](#)

Web Application Security Blog

[Understanding the most common JavaScript vulnerabilities](#)

[Streamline application security testing with Netsparker's internal scan agents](#)

[Working with custom security checks in Netsparker](#)

Then it will start to scan my target domain. While it was scanning, we were able to separately examine each discovered vulnerability. This tool gave us the vulnerability details, impacts, actions to take, remedy and external references for each of specific identified vulnerabilities. Basically, this tool has different types of boxes such as sitemap where browsed pages are being added to as you browse them, issues – previous settings, activity, progress logs, knowledge base and etc.

As shown below figures, we were able to find the request and response which were send and received. All the identified vulnerabilities were listed done on a small column in the issues field. In right side of the scanner, we could find all the number of identified disclosure details such as emails.

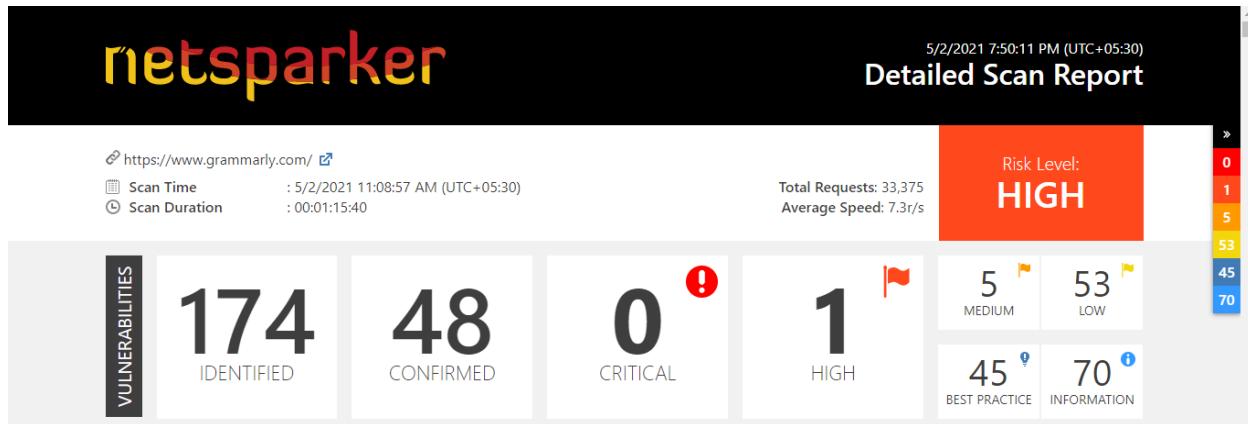
The screenshot shows the Netsparker interface during a scan of www.grammarly.com. The main window is divided into several sections:

- Sitemap - Previous Settings:** Shows a tree view of the website structure with various vulnerabilities detected, such as 'Security.txt Detected' and 'Missing X-XSS-Protection'.
- Issues - Previous Settings:** Lists 157 identified issues, including Out-of-date Version (Nginx), Weak Ciphers Enabled, and various Cross-site Scripting (XSS) and SQL injection vulnerabilities.
- HTTP Request / Response:** The central panel shows the request and response for a GET request to the root URL. The request headers include:

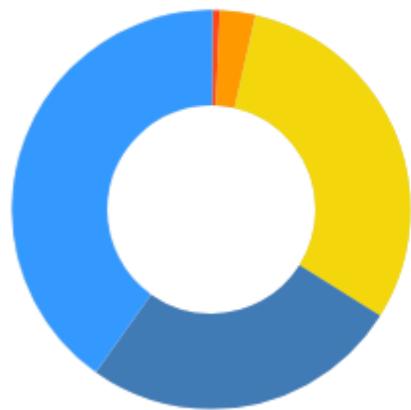

```
1 GET / HTTP/1.1
2 Host: www.grammarly.com
3 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
4 Accept-Encoding: gzip, deflate
5 Accept-Language: en-us,en;q=0.5
6 Cache-Control: no-cache
```
- Activity:** A table listing the attacking activities, showing five GET requests targeting specific parameters like requestId, isInitial, pageld, and userId, each with a unique identifier and status (Requesting or Analyzing).
- Knowledge Base:** A sidebar on the right containing a list of various security findings and their counts, such as 18 items in the Knowledge Base, 154 AJAX / XML HTTP Requests, and 117 Cookies.

As shown in the above figure, it took so much time to finish the scanning process and at last we were able to generate a html and pdf formatted report.

Finally, scan was end up with the identifying zero number of critical vulnerabilities, only one high vulnerability, 5 medium vulnerabilities, 53 number of low vulnerabilities, 45 number of best practices and 70 informational vulnerabilities within the total number of 33375 requests including 5995 head requests, 1485 number of 404 requests and only one failed request. And identified number of links were 1748 as shown below.



Identified Vulnerabilities



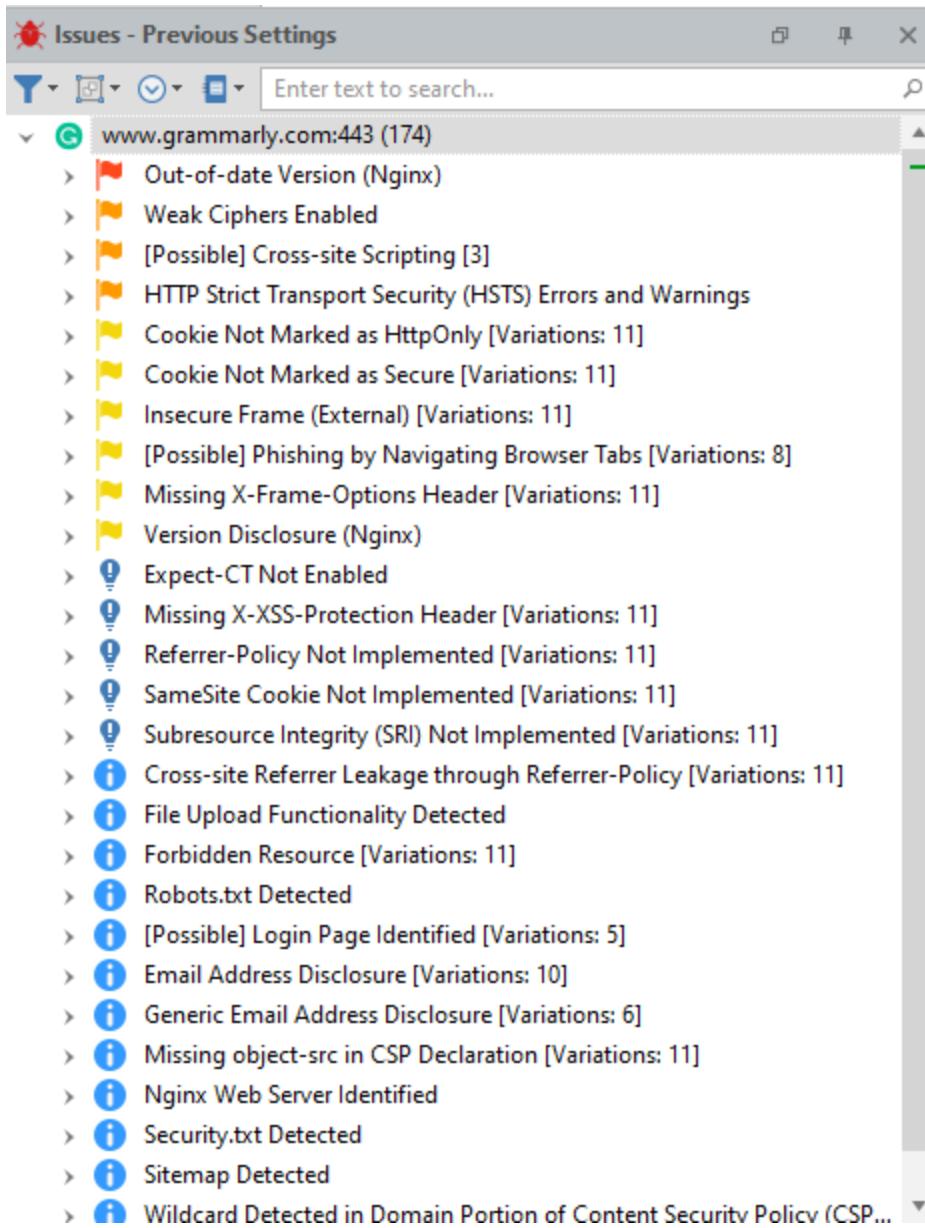
Critical	0
High	1
Medium	5
Low	53
Best Practice	45
Information	70
TOTAL	174

The below figure shows the log details regarding the scan.

The screenshot displays the Netsparker application window. On the left, there are two main sections: 'Sitemap - Previous Settings' and 'Issues - Previous Settings'. The 'Logs (26)' section is highlighted with a red border. It contains a table with columns 'Type', 'Log Time', and 'Description'. The table lists various log entries from May 2, 2021, at 11:09:50 AM to 11:12:03 AM. The descriptions include messages like 'Starting to find hidden files and folders.', 'Starting to find and analyze static resources.', and 'A URL Rewrite rule has been detected...'. To the right of the logs is a sidebar titled 'Knowledge Base (20)' containing links to various security topics such as 'AJAX / XML HTTP Requests [172]', 'Comments [111]', 'Cookies [140]', etc.

Type	Log Time	Description
Info	5/2/2021 11:09:50 AM	Starting to find hidden files and folders.
Info	5/2/2021 11:09:51 AM	Starting to find and analyze static resources.
Info	5/2/2021 11:09:52 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:52 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:09:53 AM	Finished analysing a static resource test.
Info	5/2/2021 11:10:00 AM	Finished analysing a static resource test.
Info	5/2/2021 11:10:21 AM	Finished analysing a static resource test.
Info	5/2/2021 11:10:32 AM	Finished analysing a static resource test.
Info	5/2/2021 11:10:38 AM	Custom 404 identified in https://www.grammarly.com/blog/category/product. If the crawling finishes earlier than expected go to "Custom 404 > Auto Custom 404 Detection" in policy settings and uncheck "Enabled" to ensure that Custom 404 detection doesn't cause any problem. This issue will not be reported again.
Info	5/2/2021 11:12:03 AM	A URL Rewrite rule has been detected... so fewer samples will be collected. You can enable and disable Heuristic URL...

The below figure shows the name of the identified vulnerabilities with the number of counts which were identified in our scope.



Even though total number of identified vulnerabilities were 174, confirmed number of vulnerabilities within above identified vulnerabilities were only 48. This Netsparker Professional version confirmed identified vulnerabilities by automatically exploiting them in a safe and read only manner.

Confirmed Vulnerabilities with critical level shown below.

Confirmed Vulnerabilities



Critical	0
High	0
Medium	1
Low	33
Best Practice	0
Information	14
TOTAL	48

i. Identified Vulnerabilities

1. Vulnerability: [Out-of-date Version \(Nginx\)](#) Critical Level: High

Vulnerability

Out-of-date Version (Nginx)

HIGH

Certainty : / 100

URL : <https://www.grammarly.com/blog/>

Identified Version : 1.14.2

Latest Version : 1.20.0 (in this branch)

Vulnerability Database : Result is based on 04/29/2021 20:30:00 vulnerability database content.

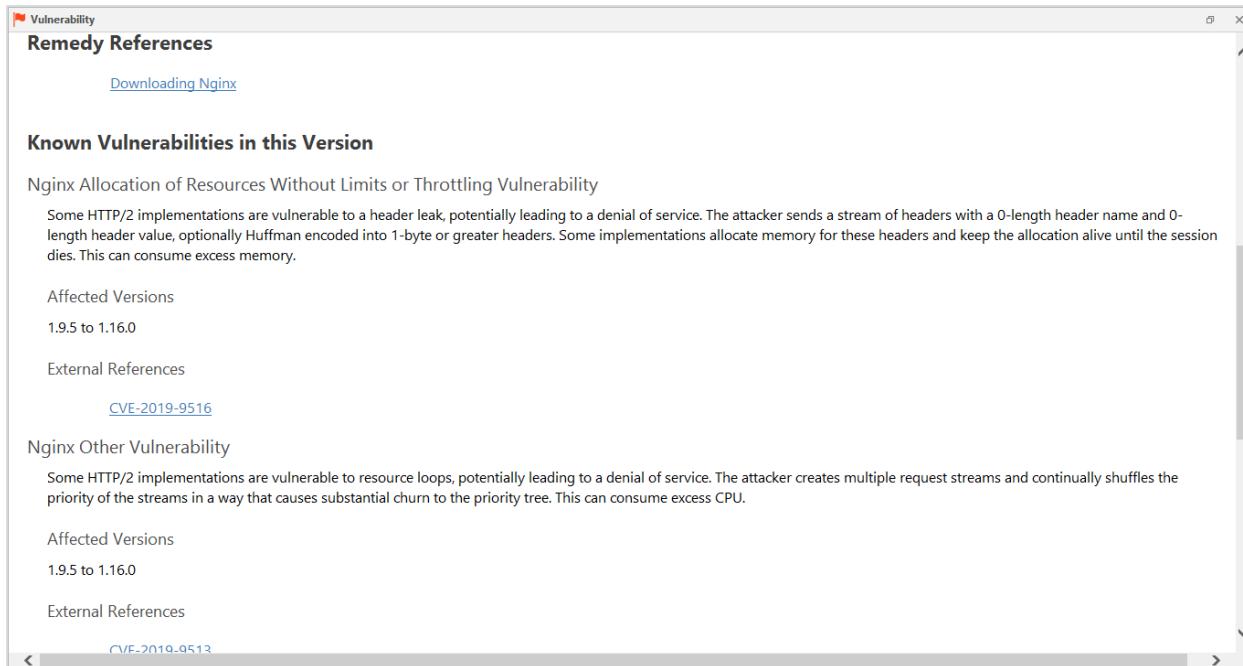
Vulnerability Details
Netsparker identified you are using an out-of-date version of Nginx.

Impact
Since this is an old version of the software, it may be vulnerable to attacks.

Remedy
Please upgrade your installation of Nginx to the latest stable version.

CLASSIFICATION

PCI DSS 3.2	6.2
OWASP 2013	A9
OWASP 2017	A9
CWE	829
CAPEC	310
WASC	13
HIPAA	164.308(A)(1)(i)
ISO27001	A.14.1.2



As above figures shown, using Netsparker scan I was able to identify that the webserver in blog sub directory was in outdated and this is vulnerable to attacks. As it shows, in order to prevent from those attacks, need to upgrade Nginx into latest stable version of version - 1.20.0. According to the Web Application Security Consortium (WASC) threat classification this can be consider as a high priority vulnerability and Nginx upgradable link also given through that report.

Upgradable link: <https://nginx.org/en/download.html>

Sub-directory of the identified webpage is shown below.

The screenshot shows a web browser window with multiple tabs open. The active tab is 'grammarly.com/blog/' which displays a cartoon illustration of a notepad with various writing tips and symbols like checkmarks and crosses. The browser's address bar also contains 'grammarly.com/blog/'. The sidebar on the right lists 'Staff Picks' with links to 'Strike the Right Tone on the Go with Our Tone Detector for Mobile', 'The Only Guide to Essay Writing You'll Ever Need', 'Welcome Rahul Roy-Chowdhury, Global Head of Product at Grammarly', and 'It's Time to Refresh These 7 Common Writing Habits'. The bottom of the page shows the URL 'https://www.grammarly.com/blog/editing/'.

2. Vulnerability: [Possible] Cross-site Scripting

Critical Level: Medium

Vulnerability Details

Netsparker detected Possible Cross-site Scripting, which allows an attacker to execute a dynamic script (JavaScript, VBScript) in the context of the application. This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by a user has been interpreted as HTML/JavaScript/VBScript by the browser. Cross-site scripting targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Although Netsparker believes there is a cross-site scripting in here, it could not confirm it. We strongly recommend investigating the issue manually to ensure it is cross-site scripting and needs to be addressed.

Impact

There are many different attacks that can be leveraged through the use of XSS, including:

- Hijacking user's active session.
- Changing the look of the page within the victim's browser.
- Mounting a successful phishing attack.
- Intercepting data and performing man-in-the-middle attacks.

Remedy

This issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, all input and output from the application should be filtered / encoded. Output should be filtered / encoded according to the output format and location.

CLASSIFICATION	SCORE
PCI DSS 3.2	6.57
OWASP 2013	A3
OWASP 2017	A7
CWE	79
CAPEC	19
WASC	8
HIPAA	164.308(A)
ISO27001	A14.2.5

CVSS 3.0 SCORE	Score
Base	7.4 (High)
Temporal	7.4 (High)
Environmental	7.4 (High)

CVSS Vector String	Score
CVSS3.0/AV:N/AC:L/PR:N/U:R:S:C/H:N/A:N	

CVSS 3.1 SCORE	Score
Base	7.4 (High)
Temporal	7.4 (High)

Vulnerability

There are many different attacks that can be leveraged through the use of XSS, including:

- Hijacking user's active session.
- Changing the look of the page within the victim's browser.
- Mounting a successful phishing attack.
- Intercepting data and performing man-in-the-middle attacks.

Remedy

This issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, all input and output from the application should be filtered / encoded. Output should be filtered / encoded according to the output format and location.

There are a number of pre-defined, well structured whitelist libraries available for many different environments. Good examples of these include [OWASP Reform](#) and [Microsoft Anti-Cross-site Scripting](#) libraries.

Additionally, you should implement a strong Content Security Policy (CSP) as a defense-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behavior in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross-site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

External References

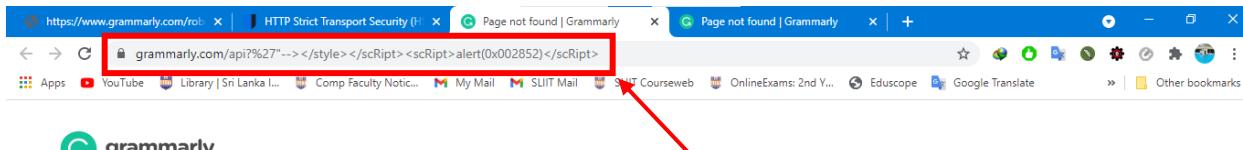
- [OWASP - Cross-site Scripting](#)
- [Cross-site Scripting Web Application Vulnerability](#)
- [XSS Shell](#)
- [XSS Tunnelling](#)

Remedy References

- [Content Security Policy \(CSP\) Explained](#)
- [Negative Impact of Incorrect CSP Implementations](#)
- [\[ASP.NET\] - Microsoft Anti-XSS Library](#)
- [OWASP XSS Prevention Cheat Sheet](#)

Cross-Site Scripting is a very serious vulnerability which allows the attacker to perform different types of attacks such as hijacking, phishing and man-in-the-middle (MIMT) attacks. This issue gains the 7.4 CVSS 3.0 score and implementing strong Content Security Policy (CSP) can be able to mitigate these types of attacks.

Displayed webpage of proof URL shown below.



The screenshot shows a browser window with multiple tabs open. The active tab is a search result for "grammarly.com/api?%27--></style><scRipt><scRipt>alert(0x002852)</scRipt>". A red box highlights this URL, and a red arrow points from it to the browser's address bar. The address bar also displays "grammarly.com".

The main content area of the browser shows an error message: "Sorry, we couldn't find that page". Below the message, it says: "We can't get you there from here, but here are some options that might help you get back on track:"

- Go to the Grammarly Homepage
- Return to Previous Page
- Contact Grammarly Support

3. Vulnerability: HTTP Strict Transport Security (HSTS) Errors and Warnings

Critical Level: Medium

The screenshot shows a Netsparker vulnerability report for the URL <https://www.grammarly.com/>. The report is titled "HTTP Strict Transport Security (HSTS) Errors and Warnings" and is marked as "MEDIUM".

Vulnerability Details: Netsparker detected errors during parsing of Strict-Transport-Security header.

Impact: The HSTS Warning and Error may allow attackers to bypass HSTS, effectively allowing them to read and modify your communication with the website.

Remedy: Ideally, after fixing the errors and warnings, you should consider adding your domain to the HSTS preload list. This will ensure that browsers automatically connect your website by using HTTPS, actively preventing users from visiting your site using HTTP. Since this list is hardcoded in users' browsers, it will enable HSTS even before they visit your page for the first time, eliminating the need for Trust On First Use (TOFU) with its associated risks and disadvantages. Unless you fix the errors and warnings your website won't meet the conditions required to enter the browser's preload list.

Browser vendors declared:

- Serve a valid certificate
- If you are listening on port 80, redirect all domains from HTTP to HTTPS on the same host. Serve all subdomains over HTTPS:
In particular, you must support HTTPS for the www subdomain if a DNS record for that subdomain exists

Impact: The HSTS Warning and Error may allow attackers to bypass HSTS, effectively allowing them to read and modify your communication with the website.

Remedy: Ideally, after fixing the errors and warnings, you should consider adding your domain to the HSTS preload list. This will ensure that browsers automatically connect your website by using HTTPS, actively preventing users from visiting your site using HTTP. Since this list is hardcoded in users' browsers, it will enable HSTS even before they visit your page for the first time, eliminating the need for Trust On First Use (TOFU) with its associated risks and disadvantages. Unless you fix the errors and warnings your website won't meet the conditions required to enter the browser's preload list.

Browser vendors declared:

- Serve a valid certificate
- If you are listening on port 80, redirect all domains from HTTP to HTTPS on the same host. Serve all subdomains over HTTPS:
In particular, you must support HTTPS for the www subdomain if a DNS record for that subdomain exists
- Serve an HSTS header on the base domain for HHTPS requests:
 - The max-age must be at least 31536000 seconds (1 year)
 - The includeSubDomains directive must be specified
 - The preload directive must be specified

If you are serving an additional redirect from your HTTPS site, that redirect must have the HSTS header (rather than the page it redirects to)

External References:

- [HTTP Strict Transport Security \(HSTS\) HTTP Header](#)
- [Wikipedia - HTTP Strict Transport Security Implementation](#)
- [Check HSTS Preload status and eligibility](#)

As errors were detected during the parsing of Strict-Transport-Security header, attackers are able to read and change your communication with the website simply by bypassing HSTS header. In order to remediate this vulnerability, domain should

add to the HSTS preloaded list and then the user's browsers can connect to the webpage using HTTPS request automatically.

4. Vulnerability: Version Disclosure (Nginx) Critical Level: Low

Vulnerability Details
Netsparker identified a version disclosure (Nginx) in the target web server's HTTP response.
This information might help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of Nginx.

Impact
An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.

Remedy
Add the following line to your nginx.conf file to prevent information leakage from the SERVER header of its HTTP response:

```
server_tokens off
```

CLASSIFICATION	
OWASP 2013	A5
OWASP 2017	A6
CWE	205
CAPEC	170
WASC	45
HIPAA	164.306(A), 164.308(A)
ISO27001	A.18.1.3

This vulnerability was regarding the previously mentioned critical vulnerability of outdated Nginx webserver. Allowing attackers to identifying the version in the target web server's HTTP response, attacker can get deep understand about the target system and able to perform specific exploits for the identified version. Simply adding “**server_tokens off**” in the webserver’s configuration file, able to remediate from those type of information disclosure.

ii. Confirmed Vulnerabilities

1. Vulnerability: Weak Ciphers Enabled Critical Level: Medium

Vulnerability

Weak Ciphers Enabled

CONFIRMED MEDIUM

URL : <https://www.grammarly.com/>

List of Supported Weak Ciphers :

- TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003C)
- TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003D)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC027)
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC028)

Vulnerability Details

Netsparker detected that weak ciphers are enabled during secure communication (SSL). You should allow only strong ciphers on your web server to protect secure communication with your visitors.

Impact

Attackers might decrypt SSL traffic between your server and your visitors.

Actions to Take

For Apache, you should modify the SSLCipherSuite directive in the `httpd.conf`.

```
SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4
```

Lighttpd:

```
ssl.honor-cipher-order = "enable"
ssl.cipher-list = "EECDH+AESGCM:EDH+AESGCM"
```

For Microsoft IIS, you should make some changes to the system registry. **Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on your computer.**

a. Click Start, click Run, type `regedit` or type `regedit`, and then click OK.
b. In Registry Editor, locate the following registry key: `HKEY\SYSTEM\CurrentControlSet\Control\SecurityProviders`
c. Set "Enabled" DWORD to "0x0" for the following registry keys:

```
SCHANNEL\Ciphers\DES 56/56
SCHANNEL\Ciphers\RC4 64/128
SCHANNEL\Ciphers\RC4 40/128
SCHANNEL\Ciphers\RC2 56/128
SCHANNEL\Ciphers\RC2 40/128
SCHANNEL\Ciphers\NULL
SCHANNEL\Hashes\MD5
```

Remedy

Configure your web server to disallow using weak ciphers.

External References

[OWASP - Insecure Configuration Management](#)
[OWASP Top 10-2017 A3-Sensitive Data Exposure](#)
[Zombie Poodle - Golden Doodle \(CBC\)](#)

<https://www.tripwire.com/state-of-security/vulnerability-management/zombie-poodle-goldendoodle/>

CLASSIFICATION

PCI DSS 3.2	6.5.4
OWASP 2013	A6
OWASP 2017	A3
CWE	327
CAPEC	217
WASC	4
ISO27001	A.14.1.3

CVSS 3.0 SCORE

Base	6.8 (Medium)
Temporal	6.8 (Medium)
Environmental	6.8 (Medium)

CVSS Vector String

CVSS:3.0:AV:A/AC:H/PR:N/Ui:N/S:U/C:H/I:H/A:N

CVSS 3.1 SCORE

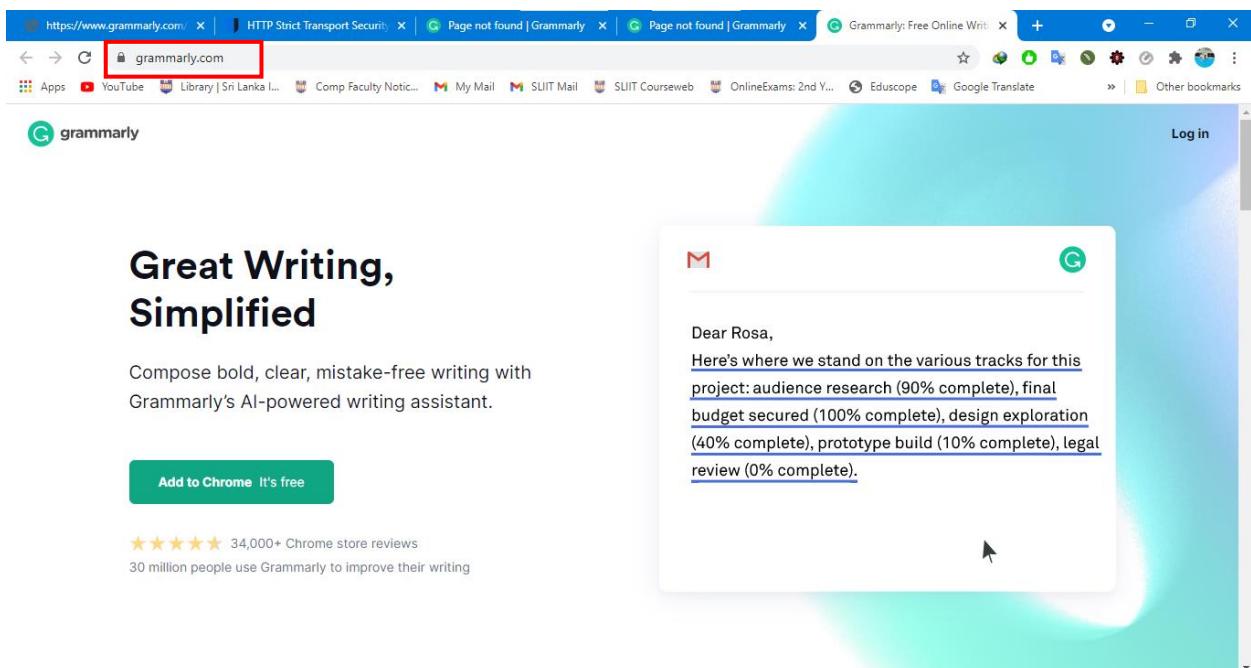
Base	6.8 (Medium)
Temporal	6.8 (Medium)
Environmental	6.8 (Medium)

CVSS Vector String

CVSS:3.1:AV:A/AC:H/PR:N/Ui:N/S:U/C:H/I:H/A:N

This vulnerability takes 6.8 base score from the CVSS 3.0 classification and this was a confirmed vulnerability by the Netsparker. This vulnerability allows attackers decrypt the traffic between the server and webpage visitors. In order to prevent from those attacks, need to specify only the strong ciphers in the web server.

Below figure show the vulnerable webpage with weak cipher enabled and that was the target domain of our scan.



2. Vulnerability: Cookie Not Marked as HttpOnly

Critical Level: Low

Vulnerability

Cookie Not Marked as HttpOnly

CONFIRMED **LOW**

URL : <https://www.grammarly.com/>

Identified Cookie(s) : `gnar_containerId`
`browser_info`
`funnelType`

Cookie Source : HTTP Header

Vulnerability Details

Netsparker identified a cookie not marked as HTTPOnly.

HTTPOnly cookies cannot be read by client-side scripts, therefore marking a cookie as HTTPOnly can provide an additional layer of protection against cross-site scripting attacks.

Impact

During a cross-site scripting attack, an attacker might easily access cookies and hijack the victim's session.

Actions to Take

See the remedy for solution.
Consider marking all of the cookies used by the application as HTTPOnly. (After these changes javascript code will not be able to read cookies.)

Remedy

Mark the cookie as HTTPOnly. This will be an extra layer of defense against XSS. However this is not a silver bullet and will not protect the system against cross-site scripting attacks. An attacker can use a tool such as [XSS Tunnel](#) to bypass HTTPOnly protection.

External References

[Netsparker - Security Cookies - HTTPOnly Flag](#)
[OWASP HTTPOnly Cookies](#)
[MSDN - ASP.NET HTTPOnly Cookies](#)



3. Vulnerability: Cookie Not Marked as Secure

Critical Level: Low

Vulnerability

Cookie Not Marked as Secure

CONFIRMED | LOW

URL : <https://www.grammarly.com/>

Identified Cookie(s) : browser_info
funnelType

Cookie Source : HTTP Header

Vulnerability Details

Netsparker identified a cookie not marked as secure, and transmitted over HTTPS. This means the cookie could potentially be stolen by an attacker who can successfully intercept and decrypt the traffic, or following a successful man-in-the-middle attack.

Impact

This cookie will be transmitted over a HTTP connection, therefore if this cookie is important (such as a session cookie), an attacker might intercept it and hijack a victim's session. If the attacker can carry out a man-in-the-middle attack, he/she can force the victim to make an HTTP request to steal the cookie.

Actions to Take

See the remedy for solution.
Mark all cookies used within the application as secure. (If the cookie is not related to authentication or does not carry any personal information, you do not have to mark it as secure.)

Classification

PCI DSS 3.2	6.5.10
OWASP 2013	A6
OWASP 2017	A3
CWE	614
CAPEC	102
WASC	15
ISO27001	A.14.1.2

CVSS 3.0 Score

Base	2 (Low)
Temporal	2 (Low)
Environmental	2 (Low)

CVSS Vector String

CVSS:3.0/AV:P/AC:H/PR:N/UF:N/S:U/C:L/I:N/A:N

Vulnerability

Impact

This cookie will be transmitted over a HTTP connection, therefore if this cookie is important (such as a session cookie), an attacker might intercept it and hijack a victim's session. If the attacker can carry out a man-in-the-middle attack, he/she can force the victim to make an HTTP request to steal the cookie.

Actions to Take

See the remedy for solution.
Mark all cookies used within the application as secure. (If the cookie is not related to authentication or does not carry any personal information, you do not have to mark it as secure.)

Remedy

Mark all cookies used within the application as secure.

Required Skills for Successful Exploitation

To exploit this issue, the attacker needs to be able to intercept traffic. This generally requires local access to the web server or to the victim's network. Attackers need to be understand layer 2, have physical access to systems either as waypoints for the traffic, or have locally gained access to a system between the victim and the web server.

External References

[Netsparker - Security Cookies - Secure Flag](#)
[.NET CookieSecure Property](#)
[How to Create Totally Secure Cookies](#)

Classification

WASC	15
ISO27001	A.14.1.2

CVSS 3.0 Score

Base	2 (Low)
Temporal	2 (Low)
Environmental	2 (Low)

CVSS Vector String

CVSS:3.0/AV:P/AC:H/PR:N/UF:N/S:U/C:L/I:N/A:N

CVSS 3.1 Score

Base	2 (Low)
Temporal	2 (Low)
Environmental	2 (Low)

CVSS Vector String

CVSS:3.1/AV:P/AC:H/PR:N/UF:N/S:U/C:L/I:N/A:N

4. Vulnerability: Insecure Frame (External)

Critical Level: Low

Vulnerability

Insecure Frame (External)

CONFIRMED LOW

URL : <https://www.grammarly.com/>
Frame Name(s) : a-ud3qy7m3imp
Sandbox Value(s) : allow-forms allow-popups allow-same-origin allow-scripts allow-top-navigation allow-modals allow-popups-to-escape-sandbox
Parsing Source : DOM Parser

Vulnerability Details
Netsparker identified an external insecure or misconfigured iframe.

Impact
Iframe sandboxing enables a set of additional restrictions for the content within a frame in order to restrict its potentially malicious code from causing harm to the web page that embeds it.
The Same Origin Policy (SOP) will prevent JavaScript code from one origin from accessing properties and functions - as well as HTTP responses - of different origins. The access is only allowed if the protocol, port and also the domain match exactly.

Here is an example, the URLs below all belong to the same origin as <http://site.com> :
<http://site.com>
<http://site.com/>
<http://site.com/my/page.html>

Whereas the URLs mentioned below aren't from the same origin as <http://site.com> :
<http://www.site.com> (a sub domain)
<http://site.org> (different top level domain)
<https://site.com> (different protocol)
<http://site.com:8080> (different port)

When the `sandbox` attribute is set, the iframe content is treated as being from a unique origin, even if its hostname, port and protocol match exactly. Additionally, sandboxed content is re-hosted in the browser with the following restrictions:

Any kind of plugin, such as ActiveX, Flash, or Silverlight will be disabled for the iframe.
Forms are disabled. The hosted content is not allowed to make forms post back to any target.
Scripts are disabled. JavaScript is disabled and will not execute.
Links to other browsing contexts are disabled. An anchor tag targeting different browser levels will not execute.
Unique origin treatment. All content is treated under a unique origin. The content is not able to traverse the DOM or read cookie information.

When the `sandbox` attribute is not set or not configured correctly, your application might be at risk.

A compromised website that is loaded in such an insecure iframe might affect the parent web application. These are just a few examples of how such an insecure frame might affect its parent:

It might trick the user into supplying a username and password to the site loaded inside the iframe.
It might navigate the parent window to a phishing page.
It might execute untrusted code.
It could show a popup, appearing to come from the parent site.

Sandbox containing a value of :

`allow-same-origin` will not treat it as a unique origin.
`allow-top-navigation` will allow code in the iframe to navigate the parent somewhere else, e.g. by changing `parent.location`.
`allow-forms` will allow form submissions from inside the iframe.
`allow-popups` will allow popups.
`allow-scripts` will allow malicious script execution however it won't allow to create popups.

Remedy
Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

Vulnerability

Insecure Frame (External)

CONFIRMED LOW

When the `sandbox` attribute is set, the iframe content is treated as being from a unique origin, even if its hostname, port and protocol match exactly. Additionally, sandboxed content is re-hosted in the browser with the following restrictions:

Any kind of plugin, such as ActiveX, Flash, or Silverlight will be disabled for the iframe.
Forms are disabled. The hosted content is not allowed to make forms post back to any target.
Scripts are disabled. JavaScript is disabled and will not execute.
Links to other browsing contexts are disabled. An anchor tag targeting different browser levels will not execute.
Unique origin treatment. All content is treated under a unique origin. The content is not able to traverse the DOM or read cookie information.

When the `sandbox` attribute is not set or not configured correctly, your application might be at risk.

A compromised website that is loaded in such an insecure iframe might affect the parent web application. These are just a few examples of how such an insecure frame might affect its parent:

It might trick the user into supplying a username and password to the site loaded inside the iframe.
It might navigate the parent window to a phishing page.
It might execute untrusted code.
It could show a popup, appearing to come from the parent site.

Sandbox containing a value of :

`allow-same-origin` will not treat it as a unique origin.
`allow-top-navigation` will allow code in the iframe to navigate the parent somewhere else, e.g. by changing `parent.location`.
`allow-forms` will allow form submissions from inside the iframe.
`allow-popups` will allow popups.
`allow-scripts` will allow malicious script execution however it won't allow to create popups.

Remedy
Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

Remedy

Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

For untrusted content, avoid the usage of `seamless` attribute and `allow-top-navigation`, `allow-popups` and `allow-scripts` in `sandbox` attribute.

External References

[HTML5 Security Cheat Sheet](#)

Remedy References

[How to Safeguard your Site with HTML5 Sandbox](#)

[Play safely in sandboxed Iframes](#)

5. Vulnerability: [Missing X-Frame-Options Header](#) Critical Level: Low

Vulnerability

Missing X-Frame-Options Header

LOW

Certainty :

URL : <https://www.grammervlv.com/>

Vulnerability Details

Netsparker detected a missing X-Frame-Options header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP header field indicates a policy that specifies whether the browser should render the transmitted resource within a frame or an iframe. Servers can declare this policy in the header of their HTTP responses to prevent clickjacking attacks, which ensures that their content is not embedded into other pages or frames.

CLASSIFICATION

OWASP 2013	A5
OWASP 2017	A6
CWE	693
CAPEC	103
ISO27001	A.14.2.5

Impact

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on a framed page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

Remedy

Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.

X-Frame-Options: DENY It completely denies to be loaded in frame/iframe.

X-Frame-Options: SAMEORIGIN It allows only if the site which wants to load has a same origin.

X-Frame-Options: ALLOW-FROM URL It grants a specific URL to load itself in a iframe. However please pay attention to that, not all browsers support this.

Employing defensive code in the UI to ensure that the current frame is the most top level window.

6. Vulnerability: Robots.txt Detected

Critical Level: Information



Vulnerability

Robots.txt Detected

CONFIRMED **INFORMATION**

URL : <https://www.grammarly.com/robots.txt>

Interesting Robots.txt Entries :

```
Disallow: /voucher/
Disallow: /social/
Disallow: /api/
Disallow: /enterprise/
Sitemap: https://www.grammarly.com/sitemap.xml
Sitemap: https://www.grammarly.com/blog/sitemap_index.xml
```

Vulnerability Details

Netsparker detected a Robots.txt file with potentially sensitive content.

Impact

Depending on the content of the file, an attacker might discover hidden directories and files.

Remedy

Please note that when you use the instructions above, search engines will not index your website except for the specified directories.

If you want to hide certain section of the website from the search engines X-Robots-Tag can be set in the response header to tell crawlers whether the file should be indexed or not:

```
User-Agent: *
Allow: /web/
Disallow: /
```

X-Robots-Tag: googlebot: nofollow
X-Robots-Tag: otherbot: noindex, nofollow

By using X-Robots-Tag you don't have to list the these files in your Robots.txt.

It is also not possible to prevent media files from being indexed by putting using Robots Meta Tags. X-Robots-Tag resolves this issue as well.

For Apache, the following snippet can be put into httpd.conf or an .htaccess file to restrict crawlers to index multimedia files without exposing them in Robots.txt

```
<Files ~ "\.(pdf)$">
# Don't index PDF files.
Header set X-Robots-Tag "noindex, nofollow"
</Files>

<Files ~ "\.(png|jpe?g|gif)$">
#Don't index image files.
Header set X-Robots-Tag "noindex"
</Files>
```

External References

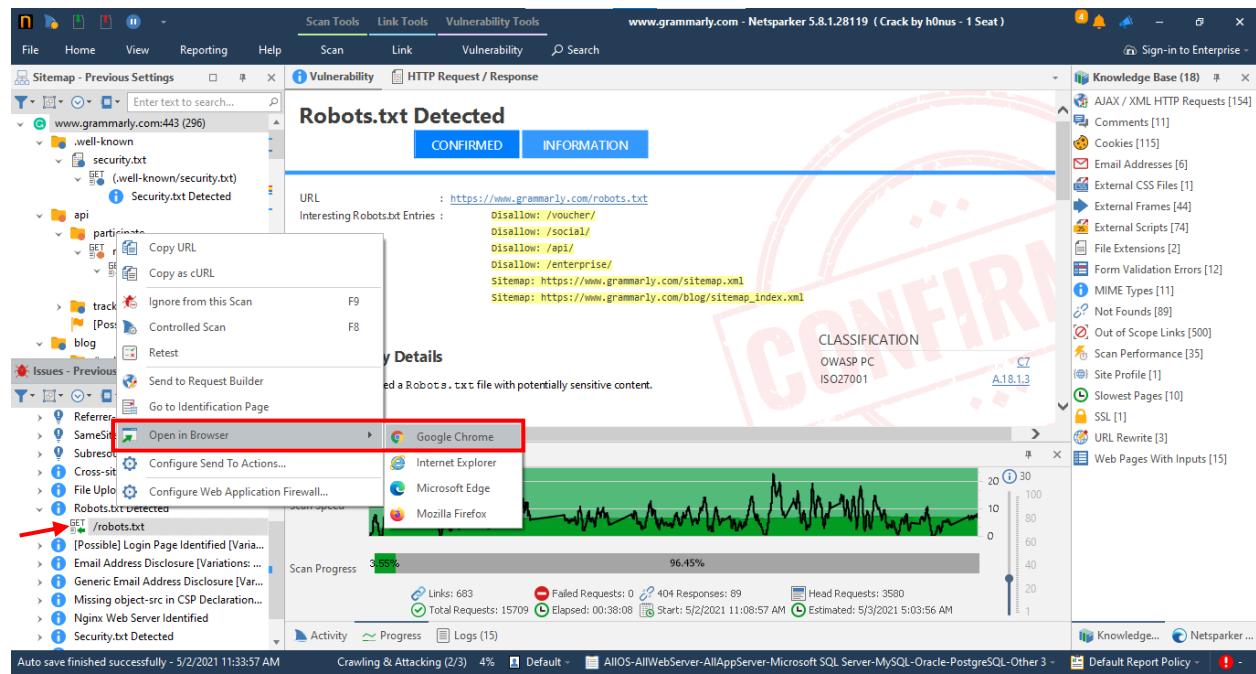
[What Content Is Not Crawled? - Google](#)
[How Search organizes information](#)
[X-Robots-Tag: A Simple Alternate For Robots.txt and Meta Tag](#)

The file robots.txt is used to inform web robots, such as search engine crawlers, where they are able to crawl and index, and where they are not allowed to crawl and index. The robots.txt file is not a security risk in and of itself, and its proper use can be considered good practice for non-security purposes. It is, however,

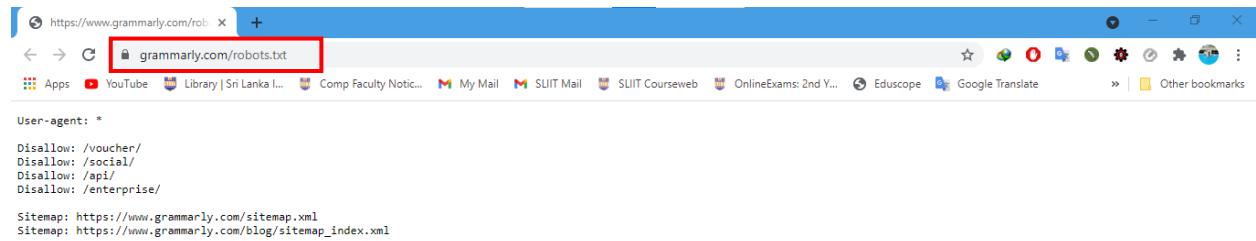
sometimes used to designate limited or private areas of a website's content. As a result, the information in the file might aid an attacker in mapping out the contents of the site, particularly if any of the locations found are not connected from somewhere else on the site. This is a major vulnerability if the program depends on robots.txt to secure access to these areas and does not apply sufficient access control. In order to prevent from those vulnerabilities, no need to include any important paths in this file, such as an administration panel's route. Those disallowed paths can be shield from unwanted entry by not including in Robots.txt file and properly securing with the authentication.

Revealed information from the Robots.txt are shown below.

Robot.txt file URL: <https://www.grammarly.com/robots.txt>



I was able to find some interesting information from that Robot.txt file.



```
User-agent: *
Disallow: /voucher/
Disallow: /social/
Disallow: /api/
Disallow: /enterprise/
Sitemap: https://www.grammarly.com/sitemap.xml
Sitemap: https://www.grammarly.com/blog/sitemap_index.xml
```

a. Sitemaps:

Sitemaps are a simple way for webmasters to let search engines know which pages on their site are crawlable. Simply a Sitemap is an XML file that lists URLs for a site along with additional information regarding each URL (when it was last changed, how much it normally varies, and how relevant it is in relation to other URLs in the site) so that search engines can crawl the site more intelligently. Crawlers find pages by following connections both inside the web and to other sites. Crawlers that help Sitemaps can pick up all URLs in the Sitemap and learn about them using the related metadata, thanks to Sitemaps. The Sitemap protocol does not ensure that web pages will be used in search engines, but it does include information for web crawlers to help them crawl your content more efficiently.

Site map information are shown below.

```

<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://www.grammarly.com/</loc>
    <priority>1.0</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/grammar-check</loc>
    <priority>0.9</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/plagiarism-checker</loc>
    <priority>0.9</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/business</loc>
    <priority>0.9</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/proofreading</loc>
    <priority>0.9</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/premium</loc>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/office-addin</loc>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/keyboard</loc>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/blog/</loc>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>https://www.grammarly.com/blog/post-sitemap1.xml</loc>
    <priority>0.8</priority>
  </url>

```

XML Sitemap

Generated by [YoastSEO](#), this is an XML Sitemap, meant for consumption by search engines.

You can find more information about XML sitemaps on [sitemaps.org](#).

This XML Sitemap contains 968 URLs.

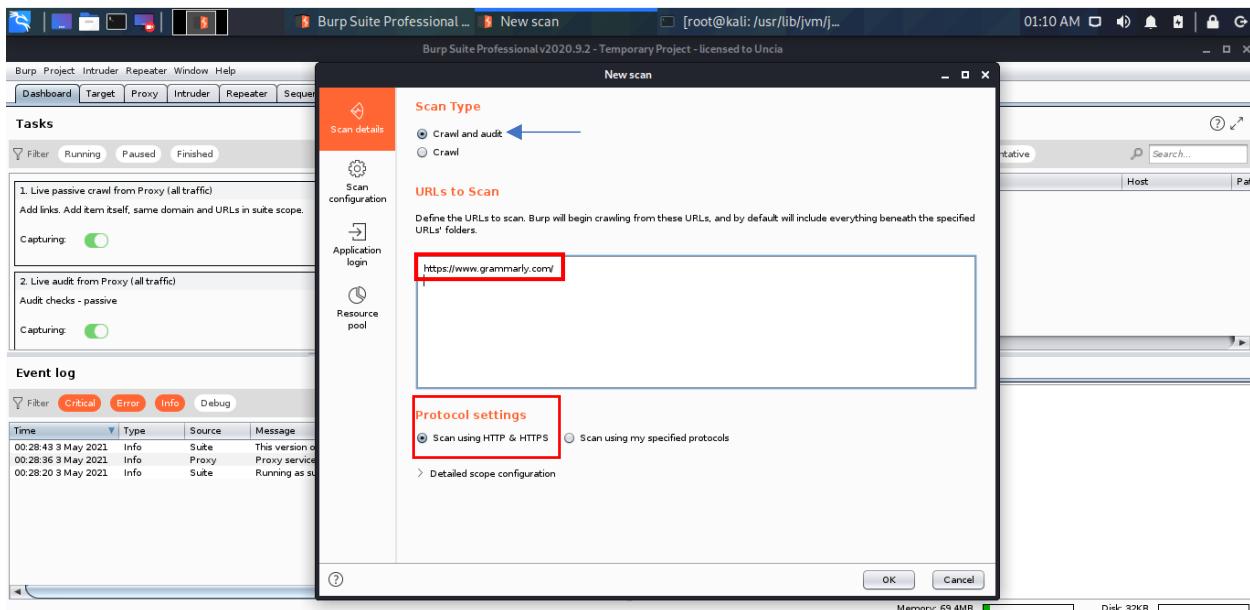
URL	Images	Last Mod.
https://www.grammarly.com/blog/	0	2021-04-29 17:52 -07:00
https://www.grammarly.com/blog/celebrities-on-tv-who-makes-more-mistakes/	1	2014-05-16 11:21 -07:00
https://www.grammarly.com/blog/celebrate-poetry-month-with-this-fun-quiz/	1	2015-04-14 01:53 -07:00
https://www.grammarly.com/blog/inf-fan-grammar-rankings-how-does-your-team-stack-up/	3	2015-05-22 06:58 -07:00
https://www.grammarly.com/blog/is-your-grammar-ready-for-fathers-day-take-the-quiz/	2	2015-06-17 05:48 -07:00
https://www.grammarly.com/blog/are-you-using-the-correct-idiot-terms-you-might-be-surprised/	4	2015-07-27 07:08 -07:00
https://www.grammarly.com/blog/5-book-to-movie-adaptations-worth-your-time/	2	2015-07-27 07:09 -07:00
https://www.grammarly.com/blog/10-ways-keeping-a-journal-will-genuinely-improve-your-life/	1	2015-08-03 04:48 -07:00
https://www.grammarly.com/blog/10-smart-things-to-do-when-waiting/	1	2015-08-05 07:53 -07:00
https://www.grammarly.com/blog/english-words-from-around-the-world/	1	2015-10-23 03:03 -07:00
https://www.grammarly.com/blog/words-peace-around-world/	1	2015-12-16 06:39 -06:00
https://www.grammarly.com/blog/the-best-word-games-to-play-with-the-family/	1	2015-12-25 05:58 -06:00
https://www.grammarly.com/blog/5-amazing-winter-idioms-and-their-etymologies/	1	2015-12-25 12:51 -06:00
https://www.grammarly.com/blog/words-inspired-by-animals/	1	2016-01-21 12:05 -06:00
https://www.grammarly.com/blog/how-to-customize-your-writing-in-job-applications/	1	2016-01-31 13:37 -06:00
https://www.grammarly.com/blog/10-jargon-phrases-to-avoid-in-business-writing/	1	2016-02-14 16:04 -06:00
https://www.grammarly.com/blog/5-tips-for-writing-an-amazing-thank-you-card/	1	2016-02-21 05:51 -06:00
https://www.grammarly.com/blog/5-best-tools-improve-writing-brain/	1	2016-02-22 04:19 -06:00
https://www.grammarly.com/blog/grammatical-errors-from-academy-awards-past/	1	2016-02-29 08:24 -06:00
https://www.grammarly.com/blog/learn-the-basics-of-grammar-one-day-at-a-time/	1	2016-03-04 08:21 -06:00
https://www.grammarly.com/blog/grammar-basics-what-are-transitive-and-intransitive-verbs/	1	2016-03-10 08:18 -06:00
https://www.grammarly.com/blog/write-job-want-good-grammar-promotions/	2	2016-03-15 09:55 -07:00
https://www.grammarly.com/blog/looking-get-lucky-st-patricks-day-idioms-may-help/	2	2016-03-17 02:12 -07:00
https://www.grammarly.com/blog/introducing-grammarly-for-dogs-april-fools/	7	2016-04-04 03:38 -07:00
https://www.grammarly.com/blog/7-weird-rare-words-illustrated/	8	2016-04-08 06:42 -07:00

Finally, as I mentioned above, using this Netsparker tool I was able to find more vulnerabilities than the other tools and this is a very valuable tool for pen testing. In order to ensure the security of the domain, need to do those remediations and other solutions regarding those vulnerabilities securely.

VI. Burp Suite Professional

Burp Suite is an advanced framework for conducting web application security monitoring. Its different tools work in unison to help the whole testing process, from plotting and analyzing an application's attack surface to identifying and manipulating security vulnerabilities. It gives complete control to the user, allowing user to mix sophisticated manual processes with cutting-edge automation to speed up and improve the task.

I used Burp Suite Professional Version to crawl and audit my target domain and I was able to get detailed report and some vulnerabilities. I installed it on my Kali environment. After the tool opens, first we need to select the option of “new scan” and then we need to select the scan type as crawl and audit. Then we need to enter the URL to scan with the protocol of HTTP & HTTPS which is under the default settings of Burp as shown below.



i. Crawl and Audit:

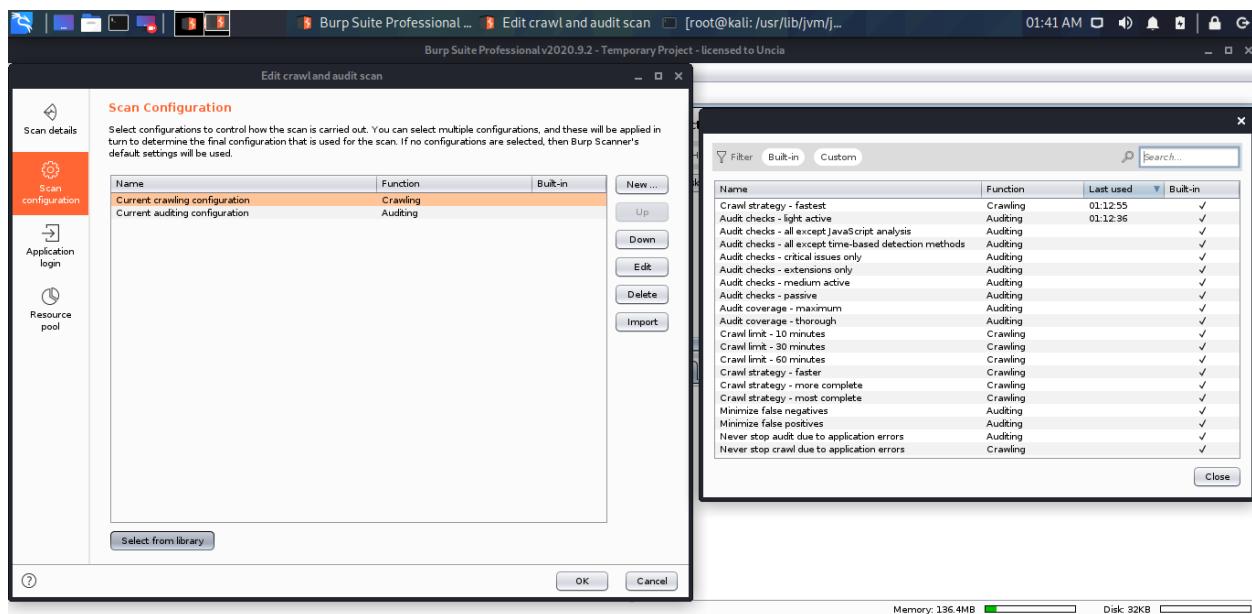
a. Crawl:

The crawl process of a scan includes traveling around the application, following connections, uploading forms, and logging in where appropriate, to index the content of the application and the navigational pathways. This seemingly easy task poses a number of obstacles that Burp's crawler is up to the task of overcoming in order to produce a detailed map of the application.

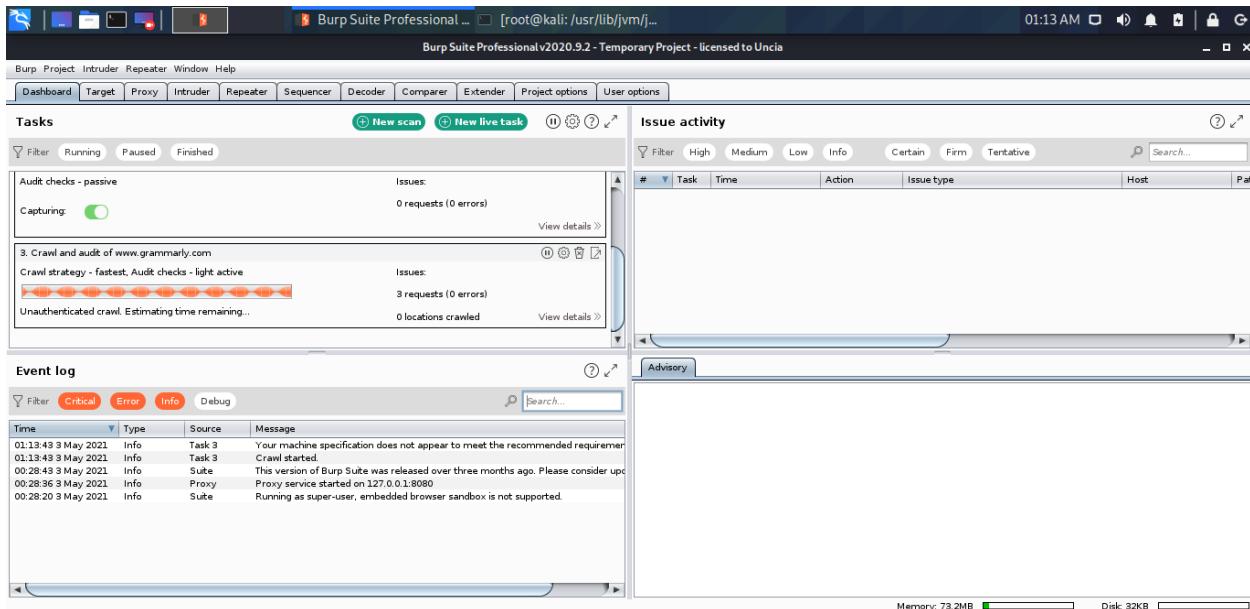
b. Audit:

During the audit process of a scan, the application's traffic and actions are analyzed for security flaws and other problems. Burp Scanner uses a variety of tools to provide a comprehensive, dead-on audit of the program being scanned.

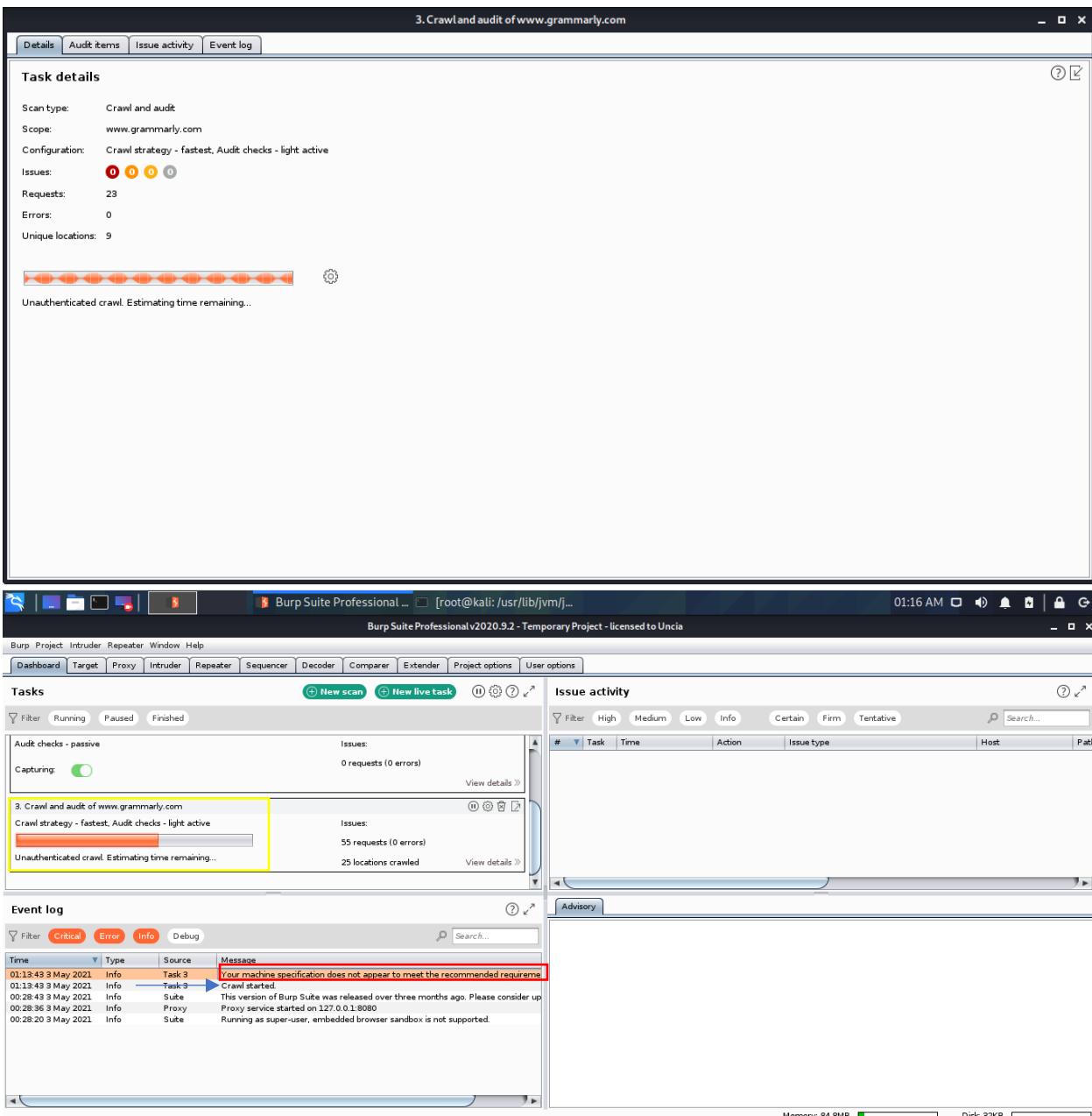
Furthermore, we can set the configurations of scan as we want, and I first scan my target using default configurations, but it takes so much time and getting my computer freeze. All the scanning configuration libraries are shown below.



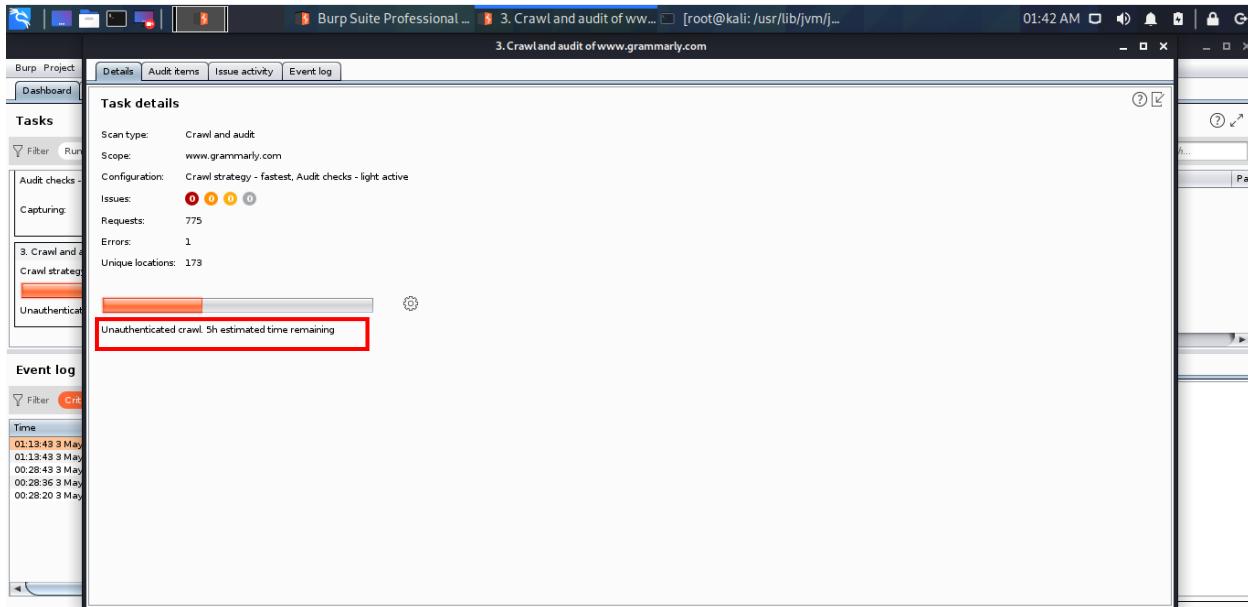
Then I change some configurations in crawl and audit feature in order to reduce the scan time and prevent my computer from freezing throughout the process. Then it will start the scan automatically.



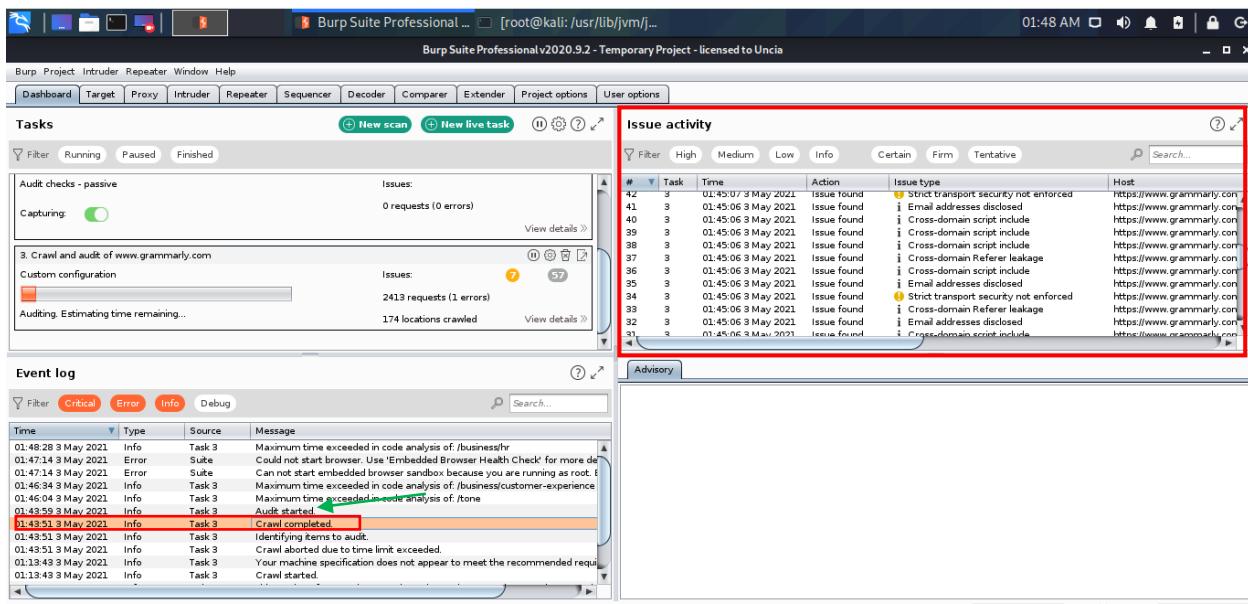
We can see the number of issues, requests, error, and some relevant information of the scan while it is continuing the web audit by maximizing the relevant tab and the following figure shows the output.



As shown in the above figure, even I change configurations to low and speedy scan, it prompts as my machine has not enough requirements and performance to continue scan and will take lot of time to finish.



Then it finished the crawling phase and starts to audit my target domain. Then we were able to examine the identified vulnerabilities under the issue activity panel.



Issue activity

#	Task	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence	Comment
50	3	01:45:07 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/blog/the-singular-theyf		Low	Certain	
42	3	01:45:07 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/terms		Low	Certain	
34	3	01:45:06 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/culture		Low	Certain	
28	3	01:45:05 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/plansin		Low	Certain	
20	3	01:45:04 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/press-room		Low	Certain	
16	3	01:45:03 3 May 2021	Issue found	Strict transport security not enforced	https://www.grammarly.com	/trust/		Low	Certain	
3	3	01:44:24 3 May 2021	Issue found	Unencrypted communications	http://www.grammarly.com	/		Low	Certain	

We can view the site map under the target tab and can view the request and response for specific vulnerabilities as shown in the below figure.

Burp Suite Professional v2020.9.2 - Temporary Project - licensed to Uncia

Site map | Scope | Issue definitions

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Contents

- about
- affiliates
- business
- contact
- contact
- cookie-policy
- culture
- edu
- enterprise
- grammar-check
- jobs
- keyboard
- media-assets
- native
- nonprofits-ngos
- office-admin
- plagiarism-checker
- plans
- premium
- press
- press-room
- privacy-policy
- privacy-policy-Sept-18-2020
- robots.txt
- security

Issues

Request

Response

```

1. GET / HTTP/1.1
2. Host: www.grammarly.com
3. Accept-Encoding: gzip, deflate
4. Accept: */*
5. Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
6. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.122 Safari/537.36
7. Connection: close
8. Cache-Control: max-age=0
9. Referer: https://www.grammarly.com/
10. Cookie: grauth=
11. Set-Cookie: funnelType=free; Max-Age=108000; Domain=gr
    
```

I was able to find the robot.txt file using this Burp tool as the previous tool of Netsparker.

As shown in the below figure, we can see the definitions of some issues under the target tab.

Name	Typical severity	Type index
OS command injection	High	0x0100100
SQL injection	High	0x0100200
SQL injection (second order)	High	0x0100210
ASP.NET tracing enabled	High	0x0100280
File path traversal	High	0x0100300
XML external entity injection	High	0x0100400
LDAP injection	High	0x0100500
XPath injection	High	0x0100600
XML injection	Medium	0x0100700
ASP.NET debugging enabled	Medium	0x0100800
HTTP PUT method is enabled	High	0x0100900
Out-of-band resource load (HTTP)	High	0x0100A00
File path manipulation	High	0x0100B00
PHP code injection	High	0x0100C00
Server-side JavaScript code injection	High	0x0100D00
Perl code injection	High	0x0100E00
Ruby code injection	High	0x0100F00
Python code injection	High	0x0101000
Executive language injection	High	0x0101100
Unidentified code injection	High	0x0101200
Server-side template injection	High	0x0101800
SSI injection	High	0x0101100
Cross-site scripting (stored)	High	0x0200100
HTTP request smuggling	High	0x0200140
Web cache poisoning	High	0x0200180
HTTP response header injection	High	0x0200200
Cross-site scripting (reflected)	High	0x0200300
Client-side template injection	High	0x0200308
Cross-site scripting (DOM-based)	High	0x0200310
Cross-site scripting (reflected DOM-based)	High	0x0200311

OS command injection

Description

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

OS command injection vulnerabilities are usually very serious and may lead to compromise of the server hosting the application, or of the application's own data and functionality. It may also be possible to use the server as a platform for attacks against other systems. The exact potential for exploitation depends upon the security context in which the command is executed, and the privileges that this context has regarding sensitive resources on the server.

Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be used to prevent attacks:

- The user data should be strictly validated. Ideally, a whitelist of specific accepted values should be used. Otherwise, only short alphanumeric strings should be accepted. Input containing any other data, including any conceivable shell metacharacter or whitespace, should be rejected.
- The application should use command APIs that launch a specific process via its name and command-line parameters, rather than passing a command string to a shell interpreter that supports command chaining and redirection. For example, the Java API Runtime.exec and the ASP.NET API Process.Start do not support shell metacharacters. This defense can mitigate the impact of an attack even in the event that an attacker circumvents the input validation defenses.

References

- OS command injection

We can view the audited items with status and some of other relevant information in a diagrammatic view.

Burp Suite Professional ... 3. Crawl and audit of www.grammarly.com [root@kali: /usr/lib/jvm/j...]

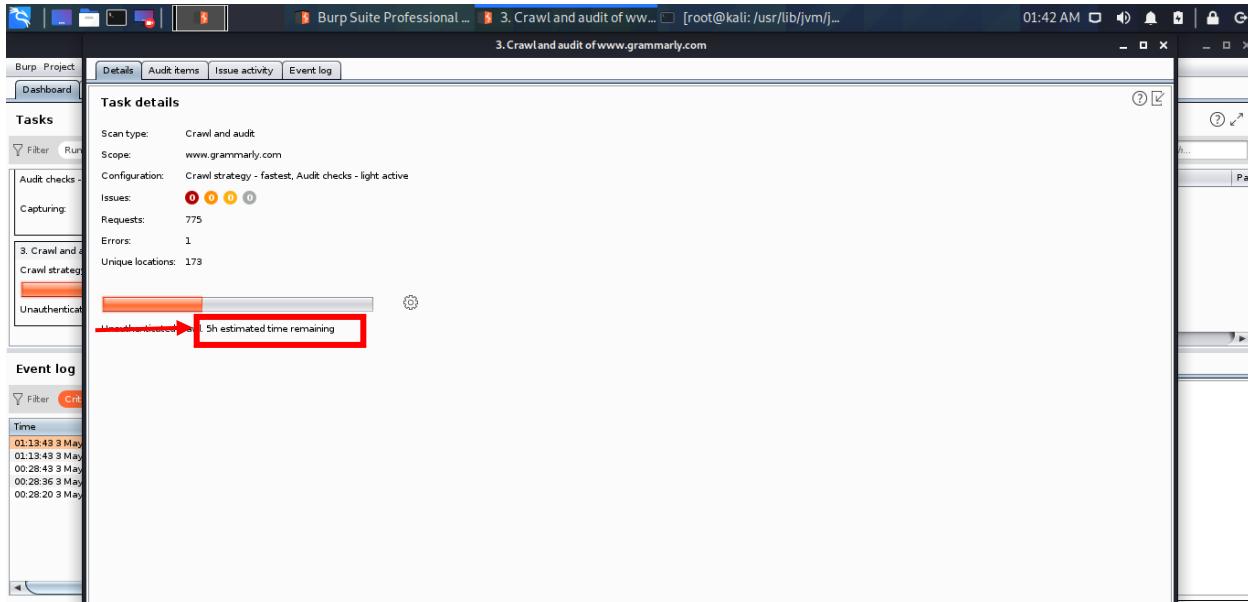
01:53 AM

3. Crawland audit of www.grammarly.com

#	A	Host	URL	Status	Passiv...	Active phases	JavaScrip...	Issues	Requests	Errors	Insertion points	Start time	End time
1		https://www.grammarly.com	/blog/category/how-to/	Scanning	1 2	1 2 3 4 5	1	61	16	01:48:59 3 May 2021			
2		https://www.grammarly.com	/business/customer-experience	Scanning	1 2	1 2 3 4 5	1	64	14	01:48:59 3 May 2021			
3		https://www.grammarly.com	/tone	Scanning	1 2	1 2 3 4 5	1	68	13	01:48:59 3 May 2021			
4		https://www.grammarly.com	/blog/what-is-audio/	Scanning	1 2	1 2 3 4 5	1	61	13	01:48:59 3 May 2021			
5		https://www.grammarly.com	/blog/grammarly-uses-ai/	Scanning	1 2	1 2 3 4 5	1	55	14	01:48:59 3 May 2021			
6		https://www.grammarly.com	/blog/bartaining/	Scanning	1 2	1 2 3 4 5	1	61	15	01:48:59 3 May 2021			
7		https://www.grammarly.com	/blog/what-is-a-dialect/	Scanning	1 2	1 2 3 4 5	1	60	15	01:49:00 3 May 2021			
8		https://www.grammarly.com	/blog/engineering/product-mindset-for-engineer...	Scanning	1 2	1 2 3 4 5	1	63	16	01:49:00 3 May 2021			
9		https://www.grammarly.com	/blog/grammarly-12-year-history/	Scanning	1 2	1 2 3 4 5	1	61	15	01:49:00 3 May 2021			
10		https://www.grammarly.com	/blog/articles/	Scanning	1 2	1 2 3 4 5	1	61	15	01:49:00 3 May 2021			
11		https://www.grammarly.com	/business/learning/unlocking-team-efficiency/	Scanning	1 2	1 2 3 4 5	1	64	16	01:49:03 3 May 2021			
12		https://www.grammarly.com	/blog/engineering/get-tag-not-revert/	Scanning	1 2	1 2 3 4 5	1	63	16	01:49:04 3 May 2021			
13		https://www.grammarly.com	/blog/essay-writing/	Scanning	1 2	1 2 3 4 5	1	117	15	01:49:15 3 May 2021			
14		https://www.grammarly.com	/blog/drone-bug-bounty-security/	Scanning	1 2	1 2 3 4 5	1	94	14	01:49:18 3 May 2021			
15		https://www.grammarly.com	/blog/drone-bug-bounty-security/	Scanning	1 2	1 2 3 4 5	1	89	14	01:49:18 3 May 2021			
16		https://www.grammarly.com	/blog/distance-learning-college/	Scanning	1 2	1 2 3 4 5	1	125	15	01:49:36 3 May 2021			
17		https://www.grammarly.com	/blog/better-writing-with-grammarly/	Scanning	1 2	1 2 3 4 5	1	101	14	01:49:36 3 May 2021			
18		https://www.grammarly.com	/blog/similes/	Scanning	1 2	1 2 3 4 5	1	132	15	01:49:47 3 May 2021			
19		https://www.grammarly.com	/business/hr	Scanning	1 2	1 2 3 4 5	1	109	14	01:49:47 3 May 2021			
20		https://www.grammarly.com	/blog/most-innovative-2019/	Scanning	1 2	1 2 3 4 5	1	118	14	01:49:47 3 May 2021			
21		https://www.grammarly.com	/blog/most-innovative-2019	Scanning	1 2	1 2 3 4 5	1	110	14	01:49:48 3 May 2021			
22		https://www.grammarly.com	/blog/resume-writing-grammarly-suggestions/	Scanning	1 2	1 2 3 4 5	1	145	15	01:49:48 3 May 2021			
23		https://www.grammarly.com	/blog/what-type-of-writer-are-you/	Scanning	1 2	1 2 3 4 5	1	146	15	01:49:48 3 May 2021			
24		https://www.grammarly.com	/robots.txt	Scanning	1 2	1 2 3 4 5	1	15	4	01:49:49 3 May 2021			
25		https://www.grammarly.com	/about/grammarly_PCI_Compliance_Report...	Scanning	1 2	1 2 3 4 5	1	18	14	01:49:50 3 May 2021			
26		https://www.grammarly.com	/blog/contributions/	Scanning	1 2	1 2 3 4 5	1	139	15	01:49:50 3 May 2021			
27		https://www.grammarly.com	/trust	Scanning	1 2	1 2 3 4 5	1	96	13	01:49:50 3 May 2021			
28		https://www.grammarly.com	/trust	Scanning	1 2	1 2 3 4 5	1	124	13	01:49:50 3 May 2021			
29		https://www.grammarly.com	/upgrade	Scanning	1 2	1 2 3 4 5	1	118	13	01:49:50 3 May 2021			
30		https://www.grammarly.com	/blog/welcome-rahal-roy-chowdhury/	Scanning	1 2	1 2 3 4 5	1	138	15	01:49:51 3 May 2021			
31		https://www.grammarly.com	/plagiarism-checker	Scanning	1 2	1 2 3 4 5	1	132	13	01:49:51 3 May 2021			
32		https://www.grammarly.com	/blog/engineering/security-infrastructure-aws/	Scanning	1 2	1 2 3 4 5	1	136	16	01:49:52 3 May 2021			
33		https://www.grammarly.com	/blog/welcome-erica-galos-alicot/	Scanning	1 2	1 2 3 4 5	1	139	15	01:49:52 3 May 2021			
34		https://www.grammarly.com	/blog/category/trends/	Scanning	1 2	1 2 3 4 5	1	96	16	01:49:52 3 May 2021			
35		https://www.grammarly.com	/blog/apostrophe/	Scanning	1 2	1 2 3 4 5	1	121	15	01:49:54 3 May 2021			
36		https://www.grammarly.com	/robots.txt	Scanning	1 2	1 2 3 4 5	1	41	4	01:49:54 3 May 2021			
37		https://www.grammarly.com	/blog/spelling/	Scanning	1 2	1 2 3 4 5	1	87	15	01:49:54 3 May 2021			

Running 9 requests in progress. 0 requests queued.

Since it took so much time, I stopped the audit phase after waiting more than three hours and generate the report based on the identified vulnerabilities. On that period, Burp Suite able to identify 6 number of low vulnerabilities and 62 number of information vulnerabilities which are under the certain confidence. Also, there were 36 number of information vulnerabilities under the firm confidence level.



We can generate the report as shown below.

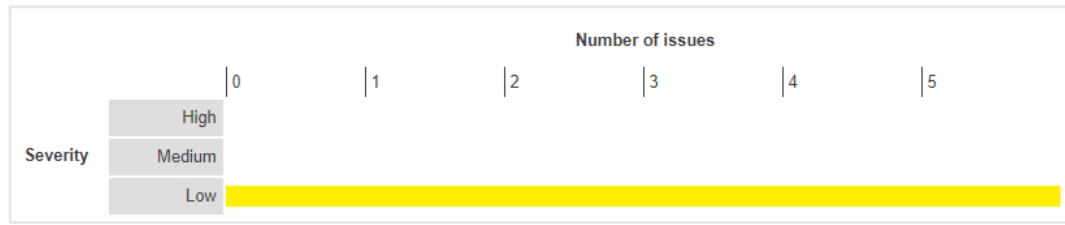
The screenshot shows the Burp Suite Professional interface. On the left, a site map for <https://www.grammarly.com> is displayed, showing various sections like About, Affiliates, Blog, Business, Contact, Culture, etc. In the center, a table lists network requests with columns for Method, URL, Params, Status, Length, MIME type, Title, and Comment. On the right, the 'Issues' panel is open, showing a list of identified vulnerabilities. One specific issue is highlighted: 'Strict transport security not enforced [6]'. The 'Advisory' panel provides details about this issue, including its severity (Low), confidence (Certain), and host (https://www.grammarly.com). The advisory text states: 'Strict transport security not enforce'.

The generated report based on the identified vulnerabilities is shown below. The report consists with all the details of specified vulnerability and remediation instructions as same as the previous Netsparker tool.

The screenshot shows the 'Burp Scanner Report' interface. At the top, there's a header with the Burp Suite logo and the text 'BURPSUITE PROFESSIONAL'. Below it, a 'Summary' section contains a table showing the number of issues identified by severity (High, Medium, Low, Information) and confidence (Certain, Firm, Tentative). The table also includes a 'Total' column. A second table, titled 'Confidence', shows the distribution of issues across different severity levels and confidence levels.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	0	0	0	0
	Medium	0	0	0	0
	Low	6	0	0	6
	Information	62	36	0	98

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.



Identified Vulnerabilities

1. Vulnerability: Strict transport security not enforced

Severity Level: Low

Vulnerable URLs:

<https://www.grammarly.com/blog/the-singular-they/>
<https://www.grammarly.com/culture>
<https://www.grammarly.com/plans/n>
<https://www.grammarly.com/press-room>
<https://www.grammarly.com/terms/>
<https://www.grammarly.com/trust/>

1. Strict transport security not enforced

[Next](#)

There are 6 instances of this issue:

- </blog/the-singular-they/>
- </culture>
- </plans/n>
- </press-room>
- </terms/>
- </trust/>

Issue description

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The ssstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=<expireTime>', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

References

- [HTTP Strict Transport Security](#)
- [sslstrip](#)
- [HSTS Preload Form](#)

Vulnerability classifications

- [CWE-523: Unprotected Transport of Credentials](#)

1.1. <https://www.grammarly.com/blog/the-singular-they/>

[Next](#)

Summary

	Severity:	Low
	Confidence:	Certain
	Host:	https://www.grammarly.com
	Path:	/blog/the-singular-they/

Request

```
GET /blog/use-the-singular-they/ HTTP/1.1
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36
Connection: close
Host: www.grammarly.com
```

Response

```
HTTP/1.1 301 Moved Permanently
Server: awselb/2.0
Date: Mon, 03 May 2021 05:34:19 GMT
Content-Type: text/html
Content-Length: 134
Connection: close
Location: https://www.grammarly.com:443/blog/use-the-singular-they/

<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
```

This vulnerability happens due to the Strict-Transport-Security header and attackers are able to read and change your communication with the website simply by bypassing HSTS header. In order to remediate this vulnerability, domain should add to the HSTS preloaded list and then the user's browsers can connect to the webpage using HTTPS request automatically.

2. Vulnerability: [TLS cookie without secure flag set](#)

Severity Level: Information

Vulnerable URLs:

<https://www.grammarly.com/>

<https://www.grammarly.com/plagiarism-checker>

4. TLS cookie without secure flag set

[Previous](#)

[Next](#)

There are 2 instances of this issue:

- /
- /plagiarism-checker

Issue background

If the secure flag is set on a cookie, then browsers will not submit the cookie in any requests that use an unencrypted HTTP connection, thereby preventing the cookie from being trivially intercepted by an attacker monitoring network traffic. If the secure flag is not set, then the cookie will be transmitted in clear-text if the user visits any HTTP URLs within the cookie's scope. An attacker may be able to induce this event by feeding a user suitable links, either directly or via another web site. Even if the domain that issued the cookie does not host any content that is accessed over HTTP, an attacker may be able to use links of the form <http://example.com:443/> to perform the same attack.

To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The secure flag should be set on all cookies that are used for transmitting sensitive data when accessing content over HTTPS. If cookies are used to transmit session tokens, then areas of the application that are accessed over HTTPS should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications.

Vulnerability classifications

- [CWE-614: Sensitive Cookie in HTTPS Session Without 'Secure' Attribute](#)

Attacker can use this issue by providing some user suitable links since the secure flag is not set and then the cookie will be transmitted in plain-text format when the user visits any HTTP URLs within the cookie's scope. In order to remediate this vulnerability, secure flag should be set on all cookies that are used for transmitting sensitive data.

3. Vulnerability: [Cookie without HttpOnly flag set](#)

Severity Level: Information

Vulnerable URLs:

<https://www.grammarly.com/>
<https://www.grammarly.com/>
<https://www.grammarly.com/plagiarism-checker>

8. Cookie without HttpOnly flag set

[Previous](#) [Next](#)

There are 3 instances of this issue:

- /
- /
- /plagiarism-checker

Issue background

If the HttpOnly attribute is set on a cookie, then the cookie's value cannot be read or set by client-side JavaScript. This measure makes certain client-side attacks, such as cross-site scripting, slightly harder to exploit by preventing them from trivially capturing the cookie's value via an injected script.

Issue remediation

There is usually no good reason not to set the HttpOnly flag on all cookies. Unless you specifically require legitimate client-side scripts within your application to read or set a cookie's value, you should set the HttpOnly flag by including this attribute within the relevant Set-cookie directive.

You should be aware that the restrictions imposed by the HttpOnly flag can potentially be circumvented in some circumstances, and that numerous other serious attacks can be delivered by client-side script injection, aside from simple cookie stealing.

References

- [Configuring HttpOnly](#)

Vulnerability classifications

- [CWE-16: Configuration](#)

I was able to find Cookie without HTTP Only flag set vulnerability found through the scan which takes the CWE-16 classification. A Set-Cookie HTTP response header includes an additional flag called HttpOnly. When creating a cookie, use the HttpOnly flag to reduce the possibility of a client side script accessing the secure cookie. In order to remediate this issue, need to set the HTTPOnly flag by including attribute within the relevant Set-cookie directive.

Finally, as I mentioned above, using this Burp Suite tool I was able to only find few vulnerabilities since it took more time than the other tool. However, I was able to find 6 low severity vulnerabilities and 62 informational vulnerabilities on that time period. In order to ensure the security of the domain, need to do those remediations and other solutions regarding those vulnerabilities securely.

Conclusion

Overall, the web application that was audited was well-designed and had a number of security features in order to protect it from cyber-attacks. During the research, only medium to low-risk vulnerabilities were found, as well as best practices, with no high-risk vulnerabilities, which is a good sign that a website is safe and reliable. The website, on the other hand, was discovered to be vulnerable to a variety of today's most common cyber-attacks, such as Cross Site Scripting and SQL injections. Grammarly is a well-known spelling and grammar checker on the internet, and the fact that such a well-known company cannot easily protect itself from these forms of cyber-attacks shows how difficult network and cyber defense has become in recent years.

References

1. <https://www.getastracom/blog/security-audit/website-security-audit/>
2. <https://webscoot.io/blog/website-security-audit/#how>
3. <https://www.bugcrowd.com/bug-bounty-list/>
4. <https://hackerone.com/grammarly?type=team>
5. <https://www.grammarly.com/>
6. <https://oscp.medium.com/owasp-amass-installation-b5ba0c54f94a>
7. <https://hydrasky.com/network-security/kali-tools/amass-subdomain-enumeration-tool/>
8. <https://github.com/projectdiscovery/subfinder>
9. https://www.youtube.com/watch?v=V910F5abjc0&ab_channel=MrMudhalai
10. https://www.youtube.com/watch?v=vV1wIK9CwI&ab_channel=5H4D0WR00
7
11. <https://tools.kali.org/information-gathering/nikto>
12. https://www.youtube.com/watch?v=bLLPxCE-x50&ab_channel=SemiYulianto
13. [https://www.cmscritic.com/reviews/netsparker-web-application-scanner-review/#](https://www/cmscritic.com/reviews/netsparker-web-application-scanner-review/#)
14. <https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/http-strict-transport-security-hsts-errors-and-warnings/>
15. https://www.youtube.com/watch?v=VP9eQhUASYQ&ab_channel=PortSwigger

Explanation Video URL:

https://mysliit-my.sharepoint.com/:v/g/personal/it19976686_my_sliit_llk/EewPtRFPRoBMrIYnc3WEffABkVmYhyrCiMG0O0ytmvfttg?e=o94ZhR