Configuration Manual

MSc. Cybersecurity
Abstraction and automation of WordPress vulnerability scanning

Stephen Farrell
Student ID: x21132445

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Stephen Farrell		
Student ID:	x21132445		
Programme:	MSCCYB1_JAN22O	**Year:**	2022
Module:	MSc Cybersecurity		
Supervisor:	Stephen Parsons		
Submission Due Date:	15/12/22		
Project Title:	Abstraction and automation of WordPress vulnerability scanning		
Word Count:	2164	**PageCount**	13

Signature: Stephen Farrell

Date: 05/01/23

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Stephen Farrell
Student ID: x21132445

1 Building the WordPress vulnerability infrastructure

For the WordPress vulnerability scanning service, AWS was chosen as the (CSP) Cloud Service Provider. Any CSP could be used to achieve the same results. AWS offers many services within the free tier plan which have been used to build and secure the WordPress vulnerability scanning service.

The solution takes advantage of services such as EBS snapshots to take a backup of the EC2 Instance in its fully configured state to enable a service to restore in the event of a hardware failure. Leveraging Security Groups which act as an Instance level firewall, the service locks down access to administer the Instance over SSH on port 22 to restricted source I.P. range. This reduces the attack surface from malicious users. Monitoring the instance performance and availability was achieved using AWS CloudWatch.

This configuration guide will detail how to build the infrastructure, in what order and the configuration of the packages installed.

The following is a diagram of the infrastructure architecture on the left with the service workflow on the right detailing out the functional components and flow as presented in Figure 1.

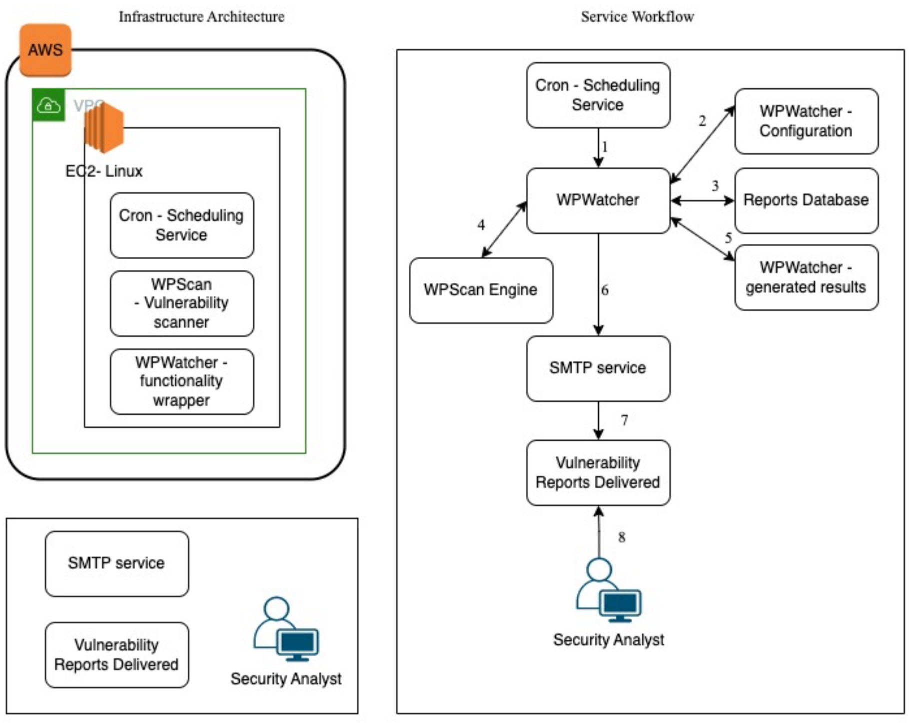

Infrastructure Architecture
AWS
VPC
EC2- Linux
Cron - Scheduling Service
WPScan - Vulnerability scanner
WPWatcher - functionality wrapper
SMTP service
Vulnerability Reports Delivered
Security Analyst
Service Workflow
Cron - Scheduling Service
1
WPWatcher
2
WPWatcher - Configuration
3
Reports Database
4
WPScan Engine
5
WPWatcher - generated results
6
SMTP service
7
Vulnerability Reports Delivered
8
Security Analyst

Figure 1: Infrastructure Architecture and Service Workflow

1.1 Create an AWS account

It is a security best practice to segment security tools, given this is a standalone service as a new AWS account is created, ensuring isolation from any other business applications. Each new AWS account comes with a free tier service for the first year. This will allow the WordPress scanner service to operate within the free tier service. [1]

1.2 Enable MFA

Given an AWS account can be accessed from anywhere in the world, it is important to add additional security by enabling MFA on the root account.[2]

1.3 Create an AWS EC2 server

The WordPress scanner service is installed on a T2.micro instance. This service is free for the first year under the free tier service. This instance type was chosen as it suits the computing profile of WordPress scanner. The T2.micro has a burstable CPU performance profile; this comes in the form of CPU credits. The WordPress scanner service is scheduled to run once a week with a total execution time of fewer than 5 minutes.

Using the AWS EC2 guide,[3] create a single t2.micro instance from debian-11-amd64-20221205-1220 AMI.[4] The WordPress scanner service can be created on any Linux distribution. This guide has been written for Debian 11.

During the instance creation, configuration restrictions to the inbound security group rule to the IP range the administrator will be connecting from. This reduces the attack surface of the instance.[5]

The instance type is chosen to suit the desired workload. In this solution, I have used a t2.micro instance. The WordPress scanner service execution impacts the CPU which needs to be monitored to ensure it can execute the required API calls within the instance limits. T2 instances have CPU credits which are monitored through CloudWatch and can trigger a notification when defined boundaries are met. Through deliberate monitoring, the infrastructure will only alert the service owner when it fails or breaches defined thresholds.

The following is the CloudWatch CPU credit usage for the WordPress scanner service execution of 10 WordPress websites in sequence over 5 minutes as presented in Figure 2.

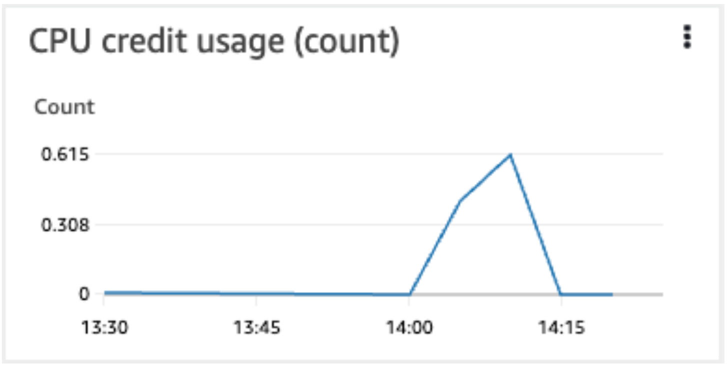


Figure 2: CPU credit usage of WordPress scanner service execution

[1] https://aws.amazon.com/resources/create-account/
[2] https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_virtual.html
[3] https://aws.amazon.com/ec2/instance-types/t2/
[4] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html
[5] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/security-group-rules.html

The WordPress scanner service is CPU heavy, In its current configuration each WordPress website is scanned in sequence. This can be changed to parallel if required. The CPU utilization for the current service execution is presented in Figure 3

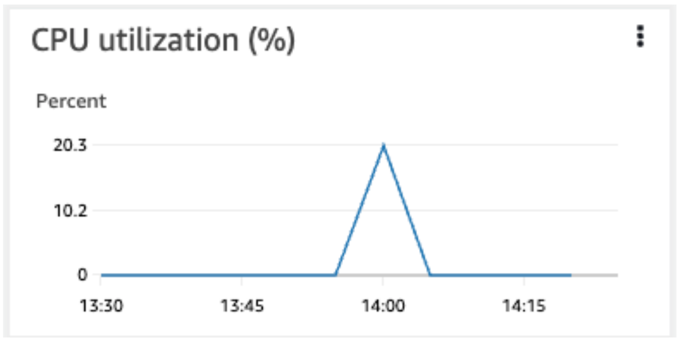
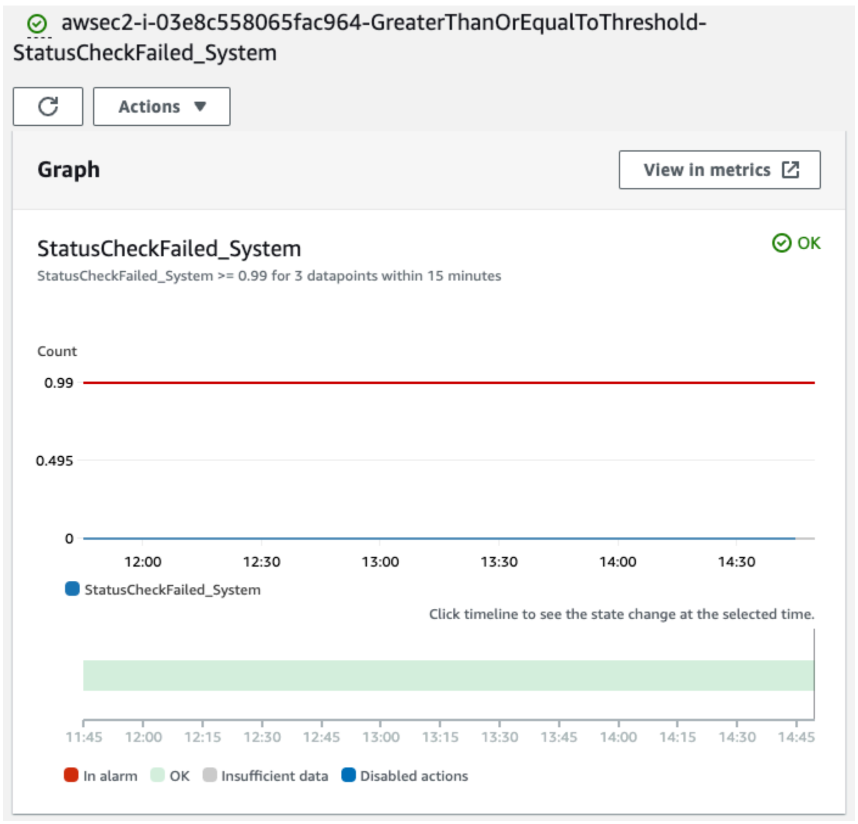


Figure 3: CPU utilization of WordPress scanner service execution

The WordPress scanner service is on a single EC2 instance. This is by design to reduce the cost of a non-critical security scanner. If the service owner required high availability and uptime a load-balanced solution could be built which would increase the cost of the service above the free tier range. The risk of failure is increased when operating a single instance. To address this risk a CloudWatch alert can be enabled to trigger an instance failure so the service operator can be notified immediately and manually restore the service from snapshots. The StatusCheckFailed_System alert is presented in Figure 4


awsec2-i-03e8c558065fac964-GreaterThanOrEqualToThreshold-StatusCheckFailed_System

StatusCheckFailed_System

StatusCheckFailed_System >= 0.99 for 3 datapoints within 15 minutes

Click timeline to see the state change at the selected time.

Figure 4: CloudWatch StatusCheckFailed_System alert.

Once the EC2 instance is fully configured and operating an EBS snapshot can be taken as a backup of the instance. In the event of an EC2 instance failure, a replacement EC2 instance can be built from the snapshot, restoring the service to its previous state and avoiding the repetition of the service setup. [6]

1.4 Create an email account

To enable Email/STMP functionality, the solution uses Gmail, but any SMTP service provider will work. Create a Gmail account using Google's guide.[7] Once the Gmail account is created, 2FA should be enabled as per Google's guide.[8]

1.5 Create WPScan API account

An account on WPScan is required to enable API access to the vulnerability database to enrich the WordPress scanner findings. The API key will be used when setting up WPWatcher. The account is free to set up and the API key gives the user 75 API requests per 24 hrs. [9]

1.6 Credential and Key storage

All sensitive credentials such as the SSH key for the EC2 instance and the Google, WPScan and API Key should be stored in a secure location. For the WordPress scanner service, LastPass was used.[10] This enabled the secure sharing of such credentials to the service owner.

1.7 Data classification and security controls consideration

It must be noted that the data gathered as part of this WordPress scanner service is publicly accessible to any internet user using the same or similar vulnerability scanning tools. The security controls chosen have taken this into account as the balance of cost and complexity of security controls vs the data it protects. With this in mind, multifactor authentication has been applied to the AWS and Gmail accounts. The WordPress scanner service is only accessible from defined IP ranges as per the EC2 instance-level security groups.

2 Configure WordPress scanner service

Once the infrastructure is built out WordPress scanner service can be installed and configured. Connect to the EC2 instance using the assigned ssh key.[11] Run updates to ensure all packages are on the latest releases and have current security patches.

On the AMI chosen as part of the EC2 instance build will have WPScan will be installed by default, if this is not the case it can be installed directly from Git.[12]

Python 3 is required and should be installed as the next step as it's a dependency package for the next packages. Once installed this will give you a warning:

---

[6] https://aws.amazon.com/ebs/snapshots/
[7] https://support.google.com/mail/answer/56256?hl=en
[8] https://support.google.com/accounts/answer/185839?hl=en&co=GENIE.Platform%3DDesktop
[9] https://wpscan.com/api
[10] https://www.lastpass.com/
[11]https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-connect-methods.html
[12] https://packaging.python.org/en/latest/guides/installing-using-linux-tools/

"WARNING: The script wheel is installed in '/home/admin/.local/bin' which is not on PATH"

This requires setting path variables.

The default Path is: "/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"

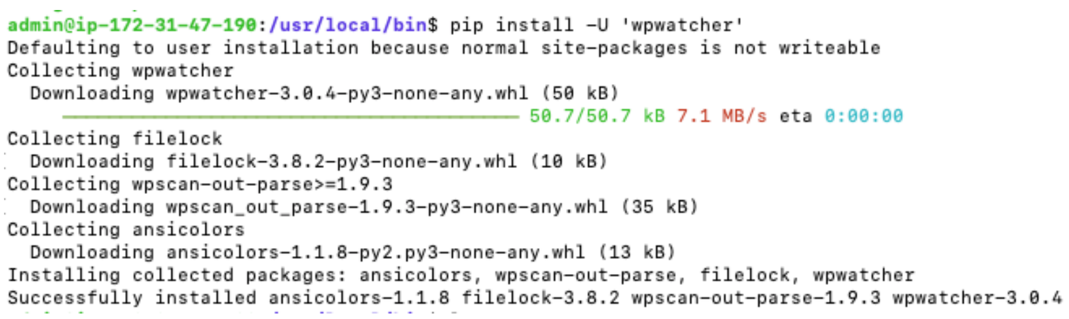
The following command will add a new Path: "export PATH="/home/admin/.local/bin:$PATH"

The new path will look like the following:
/home/admin/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

2.1 Install WPWatcher

WPWatcher can be installed directly from Git or through the Debian package manager PIP with the following command:[13][14]

"pip install -U 'wpwatcher'"



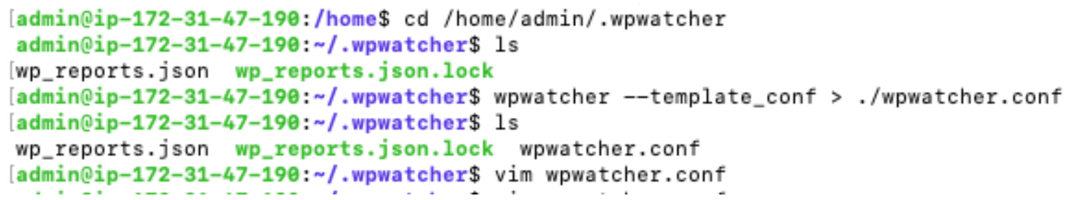
```
admin@ip-172-31-47-190:/usr/local/bin$ pip install -U 'wpwatcher'
Defaulting to user installation because normal site-packages is not writeable
Collecting wpwatcher
  Downloading wpwatcher-3.0.4-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.7/50.7 kB 7.1 MB/s eta 0:00:00
Collecting filelock
  Downloading filelock-3.8.2-py3-none-any.whl (10 kB)
Collecting wpscan-out-parse>=1.9.3
  Downloading wpscan_out_parse-1.9.3-py3-none-any.whl (35 kB)
Collecting ansicolors
  Downloading ansicolors-1.1.8-py2.py3-none-any.whl (13 kB)
Installing collected packages: ansicolors, wpscan-out-parse, filelock, wpwatcher
Successfully installed ansicolors-1.1.8 filelock-3.8.2 wpscan-out-parse-1.9.3 wpwatcher-3.0.4
```

Figure: 5 Console output for WPWatcher installation.

Once installed WPWatcher will be installed in the following path:
"/home/admin/.wpwatcher"

The next step is to create the default configuration files, this is done by moving into the .wpwatcher directory and running the following command:

"wpwatcher --template_conf > ./wpwatcher.conf" as demonstrated in the following.



```
admin@ip-172-31-47-190:/home$ cd /home/admin/.wpwatcher
admin@ip-172-31-47-190:~/.wpwatcher$ ls
wp_reports.json  wp_reports.json.lock
admin@ip-172-31-47-190:~/.wpwatcher$ wpwatcher --template_conf > ./wpwatcher.conf
admin@ip-172-31-47-190:~/.wpwatcher$ ls
wp_reports.json  wp_reports.json.lock  wpwatcher.conf
admin@ip-172-31-47-190:~/.wpwatcher$ vim wpwatcher.conf
```

Figure 6: creating wpwatcher.conf file from template.

[13] https://github.com/tristanlatr/WPWatcher
[14] https://wpwatcher.readthedocs.io/en/latest/

The wpwatcher.conf file will contain all the configuration and default service execution parameters. These parameters can be overridden by the service operator when executing a manual execution command. The wpwatcher.conf file is presented and discussed in three sections, Figure x, Figure x and Figure X.

The first section of the configuration file as presented in Figure 6 specifies the wpscan global parameters. The "--api-token" is derived from the WPScan developer account. A key configuration setting is the "--enumerate" parameter. This will define what WordPress options are scanned and to what intensity. There are 3 modes WPScan can use:

- Passive (A light scan that does not include full plugin enumeration)
- Aggressive (An intense scan that will execute a full plugin enumeration and take several hours to complete. This can have an impact the WordPress server.
- Mixed (Provides the most results, the scan can impact the performance of the WordPress site due to the intensity of the scan)

The WordPress scanner service has been configured to the missed mode with the following options provided as per line 13 of the wpwatcher.conf file "vp,vt,tt,cb,dbe,u,m". The full parameter options are as follows:

- vp (Vulnerable plugins)
- ap (All plugins)
- p (Popular plugins)
- vt (Vulnerable themes)
- at (All themes)
- t (Popular themes)
- tt (Timthumbs)
- cb (Config backups)
- dbe (Db exports)
- u (User IDs range. e.g: u1-5)
- m (Media IDs range. e.g m1-15)

The full configuration options can be found on the WPScan and WPWatcher pages. [15] [16]

[15] https://wpwatcher.readthedocs.io/en/latest/
[16] https://github.com/wpscanteam/wpscan/wiki/WPScan-User-Documentation

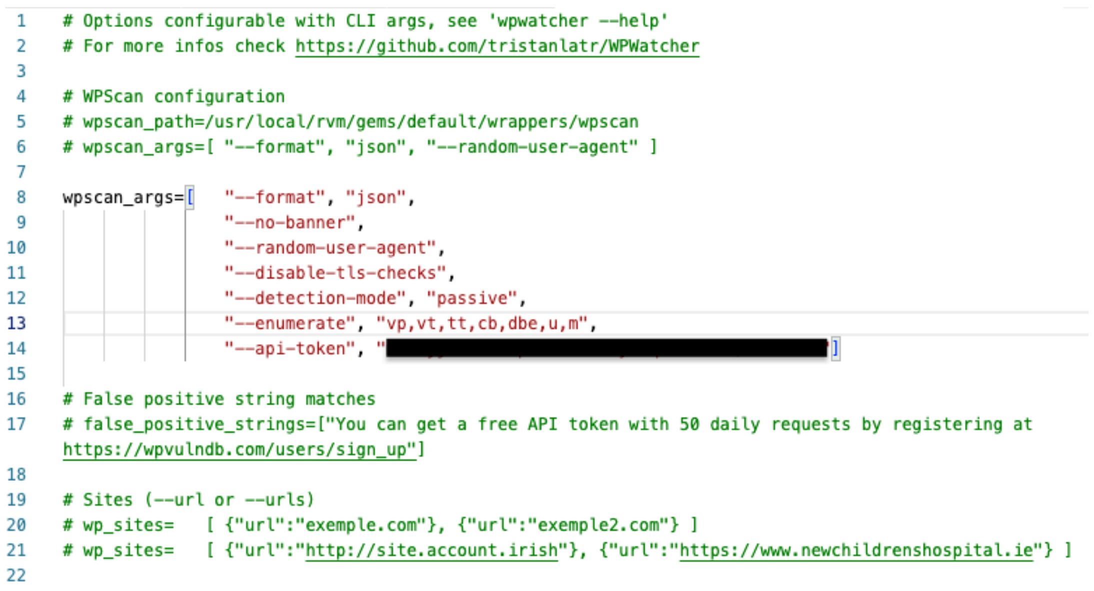

```
1   # Options configurable with CLI args, see 'wpwatcher --help'
2   # For more infos check https://github.com/tristanlatr/WPWatcher
3
4   # WPScan configuration
5   # wpscan_path=/usr/local/rvm/gems/default/wrappers/wpscan
6   # wpscan_args=[ "--format", "json", "--random-user-agent" ]
7
8   wpscan_args=[   "--format", "json",
9                   "--no-banner",
10                  "--random-user-agent",
11                  "--disable-tls-checks",
12                  "--detection-mode", "passive",
13                  "--enumerate", "vp,vt,tt,cb,dbe,u,m",
14                  "--api-token", "[redacted]]
15
16  # False positive string matches
17  # false_positive_strings=["You can get a free API token with 50 daily requests by registering at
    https://wpvulndb.com/users/sign_up"]
18
19  # Sites (--url or --urls)
20  # wp_sites=   [ {"url":"exemple.com"}, {"url":"exemple2.com"} ]
21  # wp_sites=   [ {"url":"http://site.account.irish"}, {"url":"https://www.newchildrenshospital.ie"} ]
22
```

Figure 6: The first 22 lines of the wpwatcher.conf file.

The second section of the configuration file as presented in Figure 7 specifies the WordPress sites to be scanned. These sites are added in JSON format and each entry has a commented line out for custom parameters. Any parameters specified here will override the default configuration on line 13 in Figure 6. This option is helpful to tune a scan to the customer's requirements or add customer-specific email notifications.

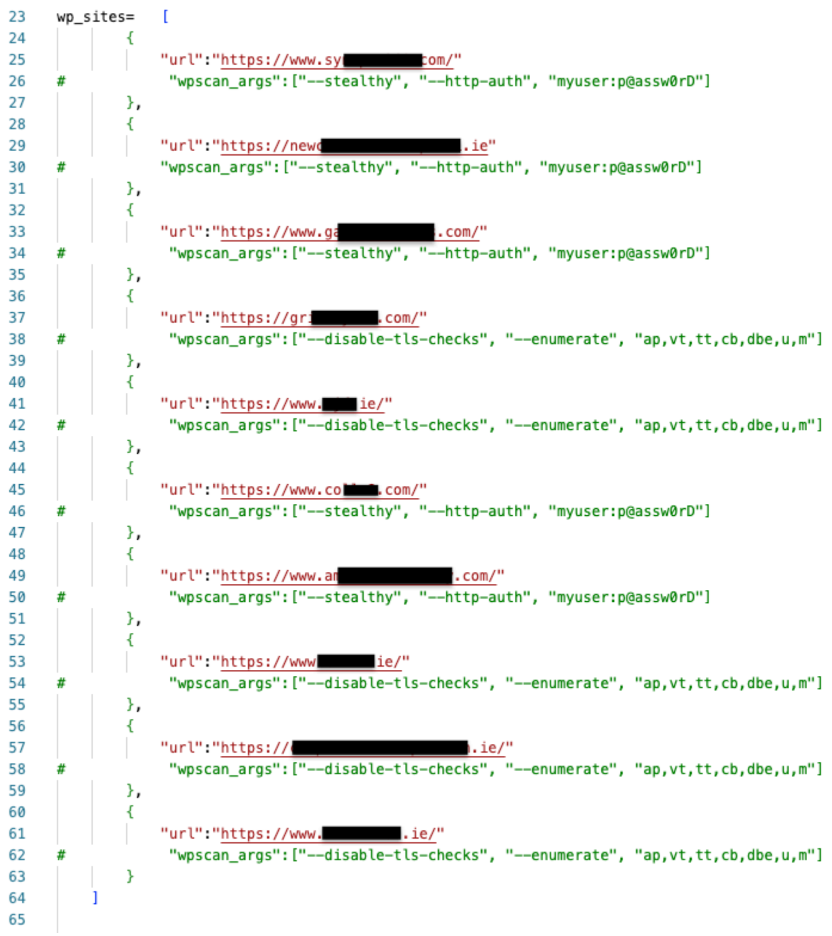
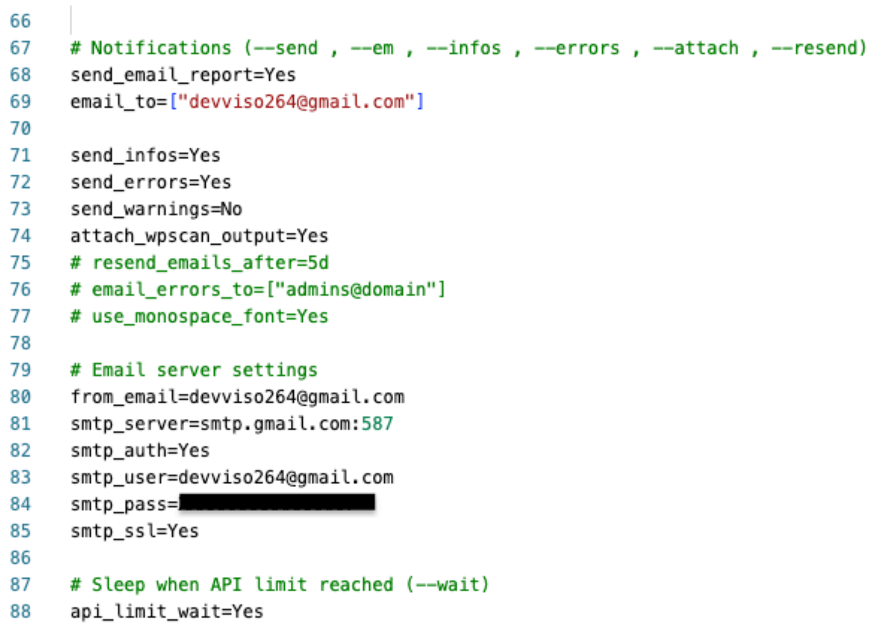

```
23  wp_sites=   [
24          {
25              "url":"https://www.sy[redacted]com/"
26  #            "wpscan_args":["--stealthy", "--http-auth", "myuser:p@assw0rD"]
27          },
28          {
29              "url":"https://new[redacted].ie"
30  #           "wpscan_args":["--stealthy", "--http-auth", "myuser:p@assw0rD"]
31          },
32          {
33              "url":"https://www.ga[redacted].com/"
34  #            "wpscan_args":["--stealthy", "--http-auth", "myuser:p@assw0rD"]
35          },
36          {
37              "url":"https://gr[redacted].com/"
38  #            "wpscan_args":["--disable-tls-checks", "--enumerate", "ap,vt,tt,cb,dbe,u,m"]
39          },
40          {
41              "url":"https://www.[redacted]ie/"
42  #            "wpscan_args":["--disable-tls-checks", "--enumerate", "ap,vt,tt,cb,dbe,u,m"]
43          },
44          {
45              "url":"https://www.co[redacted].com/"
46  #            "wpscan_args":["--stealthy", "--http-auth", "myuser:p@assw0rD"]
47          },
48          {
49              "url":"https://www.a[redacted].com/"
50  #            "wpscan_args":["--stealthy", "--http-auth", "myuser:p@assw0rD"]
51          },
52          {
53              "url":"https://www[redacted]ie/"
54  #            "wpscan_args":["--disable-tls-checks", "--enumerate", "ap,vt,tt,cb,dbe,u,m"]
55          },
56          {
57              "url":"https://[redacted].ie/"
58  #            "wpscan_args":["--disable-tls-checks", "--enumerate", "ap,vt,tt,cb,dbe,u,m"]
59          },
60          {
61              "url":"https://www.[redacted].ie/"
62  #            "wpscan_args":["--disable-tls-checks", "--enumerate", "ap,vt,tt,cb,dbe,u,m"]
63          }
64      ]
65
```

Figure 7: Lines 22 to 65 of the wpwatcher.conf file.

The third section of the configuration file as presented in Figure 8 specifies the SMTP configuration settings. Gmail has been used as the SMTP provider in this configuration but any SMTP provider can be used. A dedicated Gmail account was created to handle scan reports. Gmail rules can be created to forward specific customer reports to specific individuals such as the customer account representative or other interested parties.

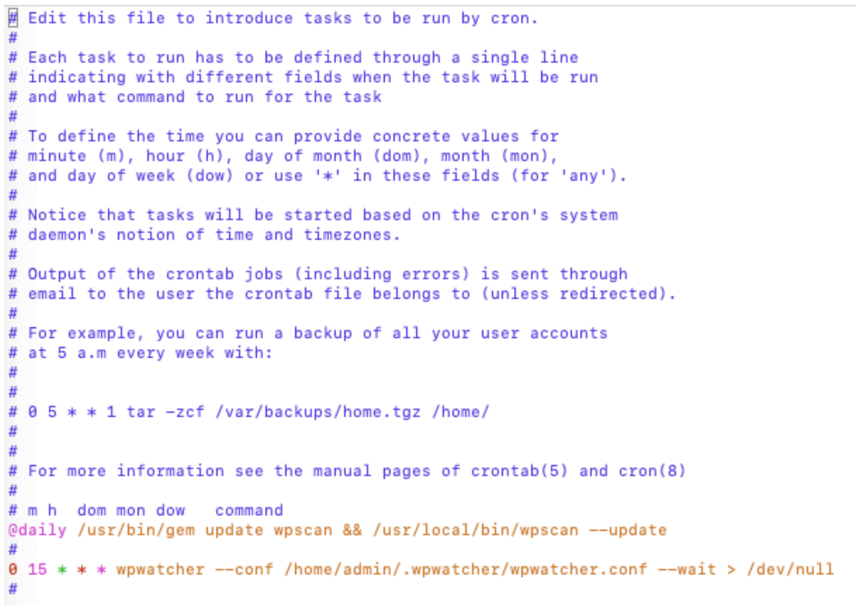


```

# Notifications (--send , --em , --infos , --errors , --attach , --resend)
send_email_report=Yes
email_to=["devviso264@gmail.com"]

send_infos=Yes
send_errors=Yes
send_warnings=No
attach_wpscan_output=Yes
# resend_emails_after=5d
# email_errors_to=["admins@domain"]
# use_monospace_font=Yes

# Email server settings
from_email=devviso264@gmail.com
smtp_server=smtp.gmail.com:587
smtp_auth=Yes
smtp_user=devviso264@gmail.com
smtp_pass=[redacted]
smtp_ssl=Yes

# Sleep when API limit reached (--wait)
api_limit_wait=Yes
```

Figure 8: Lines 66 to 88 of the wpwatcher.conf file.

2.2 Automation of WordPress scan execution

The wpwatcher.conf will handle the WordPress scanner input and execution parameters. Using the built-in Linux Cron service a simple schedule is configured to execute the on a set schedule. The following Cron tab entry sets the scan to be executed each day at 3 pm as defined in Figure 8.



```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
#
#
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
@daily /usr/bin/gem update wpscan && /usr/local/bin/wpscan --update
#
0 15 * * * wpwatcher --conf /home/admin/.wpwatcher/wpwatcher.conf --wait > /dev/null
#
```

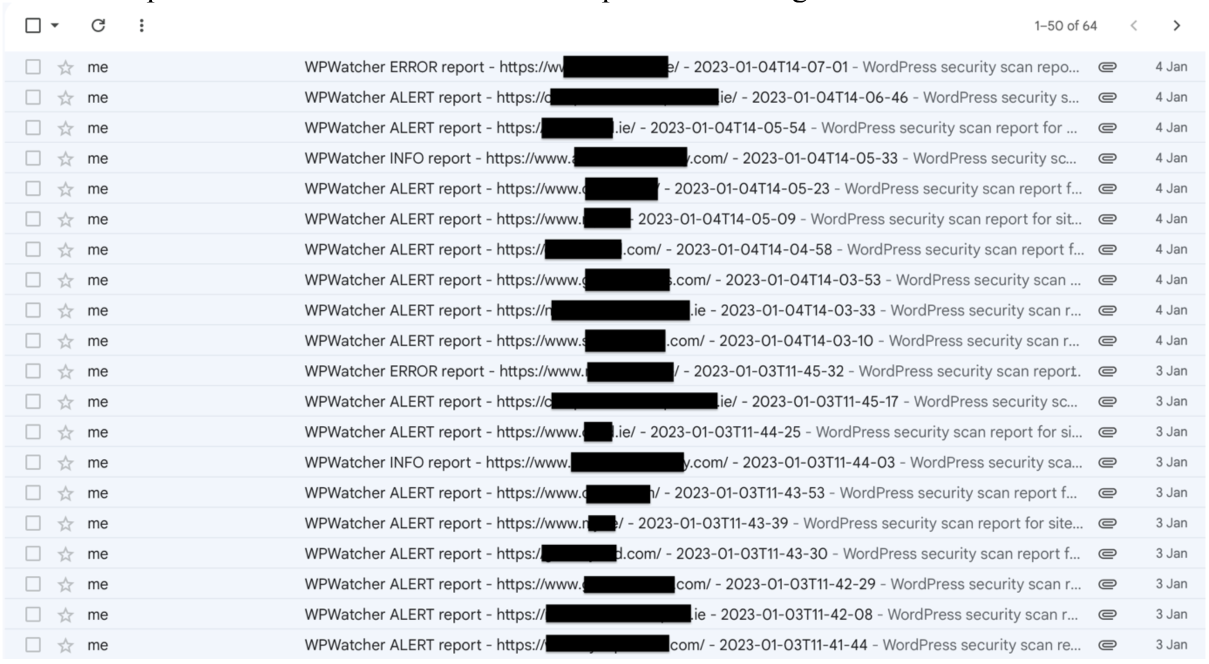
Figure 8: Crontab configuration file

2.3 WordPress scanner output

As per the WPWatcher command execution defined in Cron. A daily scan will be executed, and an email report sent on the completion of each site scan to the defined Gmail account. Each email will contain an

Individual report per site with an executive summary and status of the findings. A raw JSON file is also attached. A copy of this scan will also be stored on the server under the following directory: "/home/admin/.wpwatcher/wp_reports.json"

Each report is stored in the email server as presented in Figure 9.



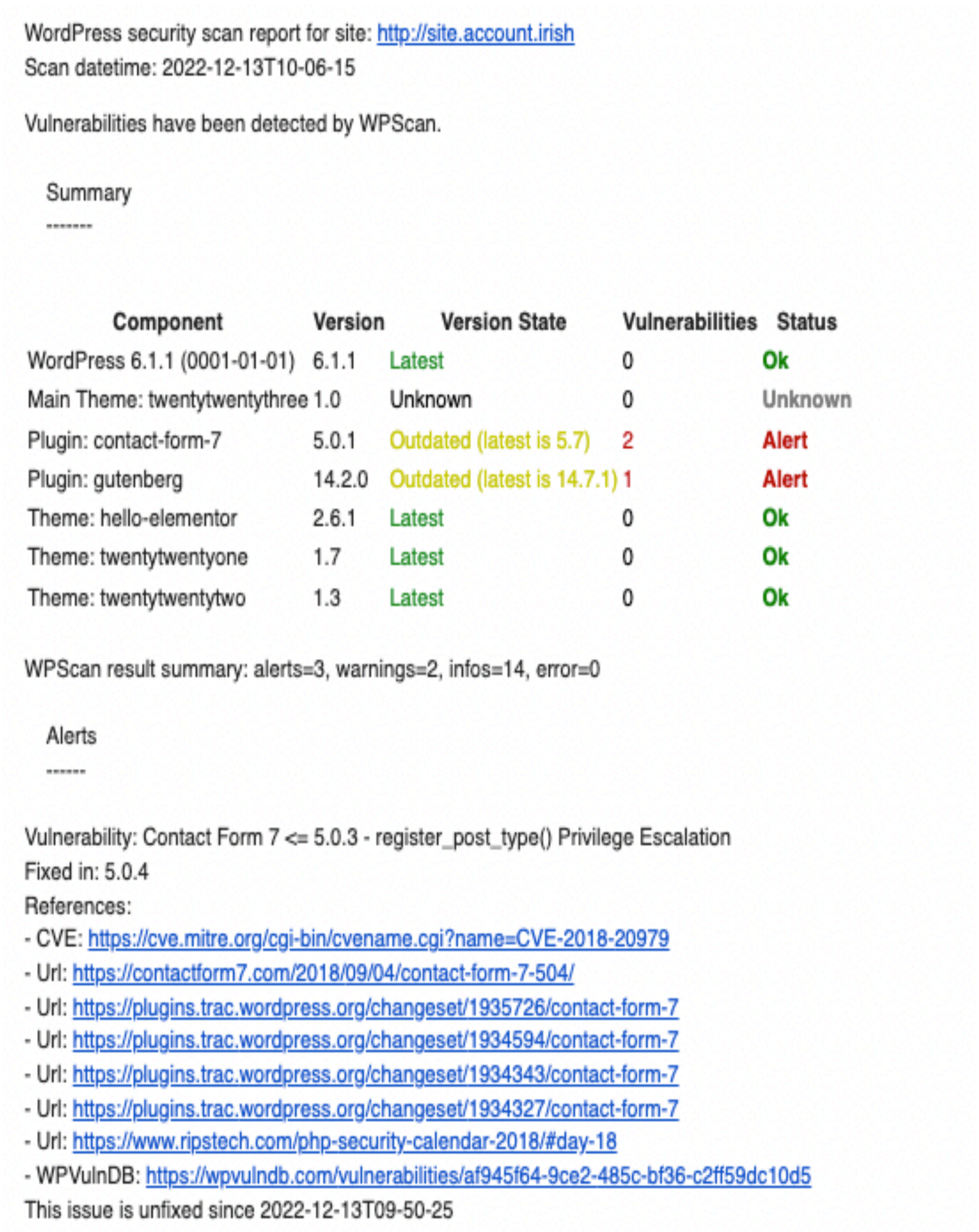
1–50 of 64

me	WPWatcher ERROR report - https://w[redacted]e/ - 2023-01-04T14-07-01 - WordPress security scan repo...	4 Jan
me	WPWatcher ALERT report - https://c[redacted]ie/ - 2023-01-04T14-06-46 - WordPress security s...	4 Jan
me	WPWatcher ALERT report - https:/[redacted].ie/ - 2023-01-04T14-05-54 - WordPress security scan report for ...	4 Jan
me	WPWatcher INFO report - https://www.a[redacted].com/ - 2023-01-04T14-05-33 - WordPress security sc...	4 Jan
me	WPWatcher ALERT report - https://www.c[redacted] - 2023-01-04T14-05-23 - WordPress security scan report f...	4 Jan
me	WPWatcher ALERT report - https://www.[redacted] 2023-01-04T14-05-09 - WordPress security scan report for sit...	4 Jan
me	WPWatcher ALERT report - https://[redacted].com/ - 2023-01-04T14-04-58 - WordPress security scan report f...	4 Jan
me	WPWatcher ALERT report - https://www.g[redacted].com/ - 2023-01-04T14-03-53 - WordPress security scan ...	4 Jan
me	WPWatcher ALERT report - https://n[redacted].ie - 2023-01-04T14-03-33 - WordPress security scan r...	4 Jan
me	WPWatcher ALERT report - https://www.s[redacted].com/ - 2023-01-04T14-03-10 - WordPress security scan r...	4 Jan
me	WPWatcher ERROR report - https://www.[redacted]/ - 2023-01-03T11-45-32 - WordPress security scan report.	3 Jan
me	WPWatcher ALERT report - https://c[redacted]ie/ - 2023-01-03T11-45-17 - WordPress security sc...	3 Jan
me	WPWatcher ALERT report - https://www.[redacted].ie/ - 2023-01-03T11-44-25 - WordPress security scan report for si...	3 Jan
me	WPWatcher INFO report - https://www.[redacted].com/ - 2023-01-03T11-44-03 - WordPress security sca...	3 Jan
me	WPWatcher ALERT report - https://www.c[redacted]/ - 2023-01-03T11-43-53 - WordPress security scan report f...	3 Jan
me	WPWatcher ALERT report - https://www.n[redacted]/ - 2023-01-03T11-43-39 - WordPress security scan report for site...	3 Jan
me	WPWatcher ALERT report - https:/[redacted]d.com/ - 2023-01-03T11-43-30 - WordPress security scan report f...	3 Jan
me	WPWatcher ALERT report - https://www.[redacted]com/ - 2023-01-03T11-42-29 - WordPress security scan r...	3 Jan
me	WPWatcher ALERT report - https://[redacted].ie - 2023-01-03T11-42-08 - WordPress security scan r...	3 Jan
me	WPWatcher ALERT report - https://[redacted]com/ - 2023-01-03T11-41-44 - WordPress security scan re...	3 Jan

Figure 9: Automated email reports in Gmail.

3 Operating the WordPress vulnerability scanning service

The consumer of the vulnerability scanning service results will be the security analysts. They will review each of the report findings per customer against the findings in the risk management database. This database is used to track the customer's risks, assign severity and track resolution. Any new findings identified in the WordPress vulnerability results will be added to the risk management database to be addressed as part of the monthly customer status meeting. An example of this report email is presented in Figure 10.



WordPress security scan report for site: http://site.account.irish
Scan datetime: 2022-12-13T10-06-15

Vulnerabilities have been detected by WPScan.

Summary

Component	Version	Version State	Vulnerabilities	Status
WordPress 6.1.1 (0001-01-01)	6.1.1	Latest	0	Ok
Main Theme: twentytwentythree	1.0	Unknown	0	Unknown
Plugin: contact-form-7	5.0.1	Outdated (latest is 5.7)	2	Alert
Plugin: gutenberg	14.2.0	Outdated (latest is 14.7.1)	1	Alert
Theme: hello-elementor	2.6.1	Latest	0	Ok
Theme: twentytwentyone	1.7	Latest	0	Ok
Theme: twentytwentytwo	1.3	Latest	0	Ok

WPScan result summary: alerts=3, warnings=2, infos=14, error=0

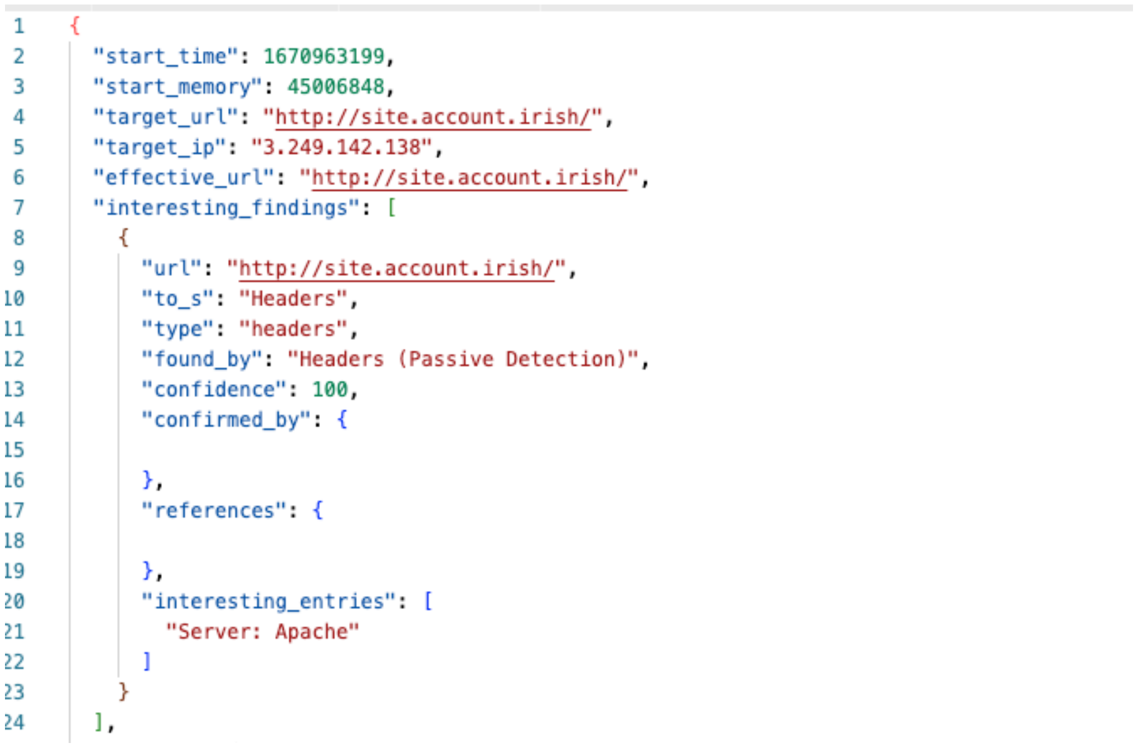
Alerts

Vulnerability: Contact Form 7 <= 5.0.3 - register_post_type() Privilege Escalation
Fixed in: 5.0.4
References:
- CVE: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-20979
- Url: https://contactform7.com/2018/09/04/contact-form-7-504/
- Url: https://plugins.trac.wordpress.org/changeset/1935726/contact-form-7
- Url: https://plugins.trac.wordpress.org/changeset/1934594/contact-form-7
- Url: https://plugins.trac.wordpress.org/changeset/1934343/contact-form-7
- Url: https://plugins.trac.wordpress.org/changeset/1934327/contact-form-7
- Url: https://www.ripstech.com/php-security-calendar-2018/#day-18
- WPVulnDB: https://wpvulndb.com/vulnerabilities/af945f64-9ce2-485c-bf36-c2ff59dc10d5

This issue is unfixed since 2022-12-13T09-50-25

Figure 10: Sample email from the WordPress vulnerability scanning service.

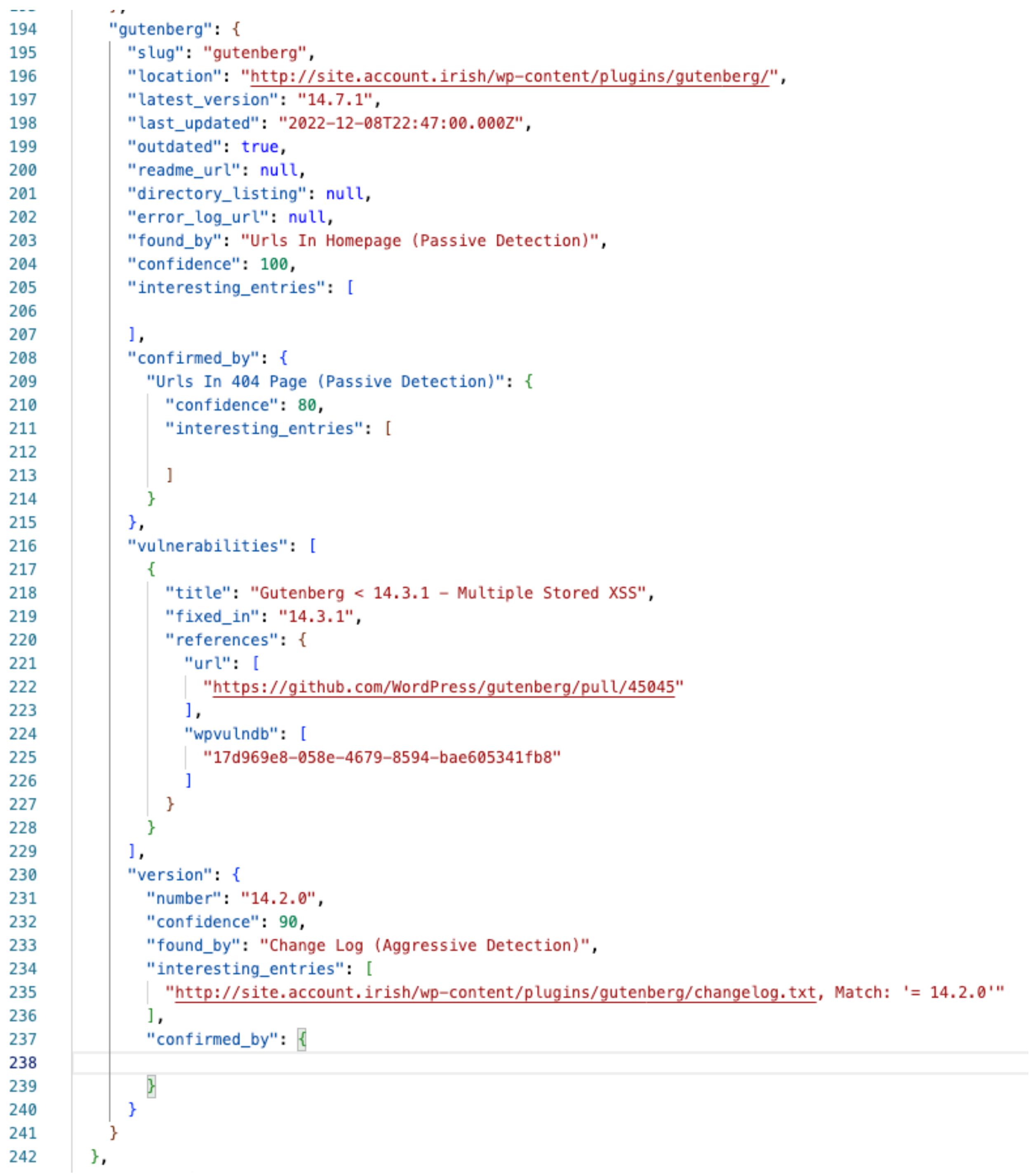
Each vulnerability report email will come with the full report findings in JSON format. The file comes with additional detail such as the dates, times and links to the relevant CVE or wpvulndb findings. The following are some extracts of the JSON file to highlight its contents. In Figure 11 the file contains date timestamps, the target of the scan and the scan type. In this case it was “Passive Detection”.



```
{
  "start_time": 1670963199,
  "start_memory": 45006848,
  "target_url": "http://site.account.irish/",
  "target_ip": "3.249.142.138",
  "effective_url": "http://site.account.irish/",
  "interesting_findings": [
    {
      "url": "http://site.account.irish/",
      "to_s": "Headers",
      "type": "headers",
      "found_by": "Headers (Passive Detection)",
      "confidence": 100,
      "confirmed_by": {

      },
      "references": {

      },
      "interesting_entries": [
        "Server: Apache"
      ]
    }
  ],
```

Figure 11: Vulnerability report JSON file extract.

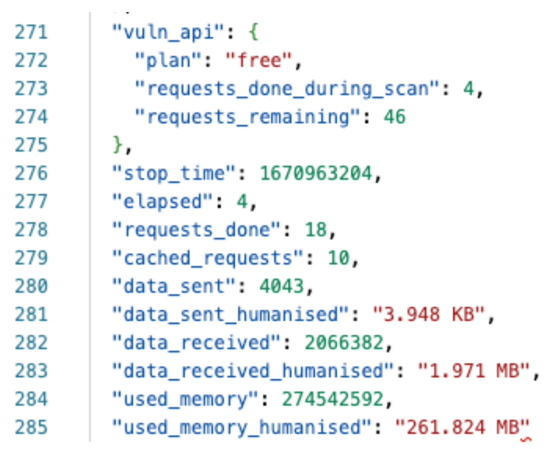
In the following JSON extract in Figure 12, there is an example of a vulnerability finding for the “gutenberg” plugin. The location of the file and version are identified. The wpvulndb link is also included which provides a detailed report on the vulnerability.



```
    "gutenberg": {
      "slug": "gutenberg",
      "location": "http://site.account.irish/wp-content/plugins/gutenberg/",
      "latest_version": "14.7.1",
      "last_updated": "2022-12-08T22:47:00.000Z",
      "outdated": true,
      "readme_url": null,
      "directory_listing": null,
      "error_log_url": null,
      "found_by": "Urls In Homepage (Passive Detection)",
      "confidence": 100,
      "interesting_entries": [

      ],
      "confirmed_by": {
        "Urls In 404 Page (Passive Detection)": {
          "confidence": 80,
          "interesting_entries": [

          ]
        }
      },
      "vulnerabilities": [
        {
          "title": "Gutenberg < 14.3.1 - Multiple Stored XSS",
          "fixed_in": "14.3.1",
          "references": {
            "url": [
              "https://github.com/WordPress/gutenberg/pull/45045"
            ],
            "wpvulndb": [
              "17d969e8-058e-4679-8594-bae605341fb8"
            ]
          }
        }
      ],
      "version": {
        "number": "14.2.0",
        "confidence": 90,
        "found_by": "Change Log (Aggressive Detection)",
        "interesting_entries": [
          "http://site.account.irish/wp-content/plugins/gutenberg/changelog.txt, Match: '= 14.2.0'"
        ],
        "confirmed_by": {

        }
      }
    }
  },
```

Figure 12: An example of a Vulnerability finding

At the end of each JSON report will be a summary resources used such as data transfers, number of requests and the api request balance. This example scan was within the 75 api requests per 24 hr cap as presented in Figure 13.



```
  "vuln_api": {
    "plan": "free",
    "requests_done_during_scan": 4,
    "requests_remaining": 46
  },
  "stop_time": 1670963204,
  "elapsed": 4,
  "requests_done": 18,
  "cached_requests": 10,
  "data_sent": 4043,
  "data_sent_humanised": "3.948 KB",
  "data_received": 2066382,
  "data_received_humanised": "1.971 MB",
  "used_memory": 274542592,
  "used_memory_humanised": "261.824 MB"
```

Figure 13: Summary of scan resources used.

References

WPScan. 2022. WPScan WordPress Security Scanner. [Online]. Available at: https://wpscan.com/wordpress-security-scanner [Accessed on: 11 November 2022].

WPWatcher. 2022. Automating WPScan to scan and report vulnerable Wordpress sites. [Online]. Available at: https://github.com/tristanlatr/WPWatcher [Accessed: 05 November 2022].

Amazon Web Services (AWS). 2022. Amazon Web Services, Inc. [Online]. Available at: https://aws.amazon.com/ [Accessed: 05 November 2022].

WPScan - Github. (2023). [Online]. Available at: https://github.com/wpscanteam/wpscan [Accessed: 02 November 2022].