# OnlineKALI: Online Vulnerability Scanner

**Parthajit Dholey and Anup Kumar Shaw**

**Abstract** OnlineKALI is a Web framework for vulnerability assessment which allows you to quickly do a security audit of your own websites and network infrastructures from a remote location without having to set up external pen-testing operating system and with very high-speed network capability and processing power. It allows you to scan, enumerate the security loopholes, and vulnerability with full customization of the open-source tools. It uses a chroot or Docker environment to launch an attack without affecting the main system. It uses the features of Django, PostgreSQL, Jinja2, and python to be secure as far as possible. This paper is basically to take a maximum of open-source tools of Kali Linux and put into the cloud so that all can work without any hardware or network issues.

**Keywords** Vulnerability scanner · Open-source tools · Pen-testing framework Web application vulnerabilities · Security testing

## 1 Introduction

Penetration testing is a series of activities undertaken to identify and exploit security vulnerabilities. It helps to confirm the effectiveness or ineffectiveness of the security measures that have been implemented [1]. Today day by day, the threats are increasing in the digital world. Everyone is coming online to increase their business to the whole world, but many of them lack basic security which can lead to loss of money and business. So lots of manual human effort and infrastructure and network setup are required to scan the servers [2]. The attacks and breaches harm an organization reputation. To eradicate threats, a lot of open-source tools are available to analyze the servers if they lack some security hole which can be harmful [3]. We are trying

P. Dholey (✉) · A. K. Shaw
HackCieux, Cyber Space Security: Consultancy, Training, and Solutions, Kolkata, India
e-mail: p.dholey@hackcieux.com; parthajit1994@gmail.com

A. K. Shaw
e-mail: anup@hackcieux.com

M. Chakraborty et al. (eds.), *Proceedings of International Ethical Hacking Conference 2018*, Advances in Intelligent Systems and Computing 811,
https://doi.org/10.1007/978-981-13-1544-2_3

here to move such open-source tools to the clouds and provide users a usable Web framework which has high network speed and CPU performance that can let the users scan their servers and find a known vulnerability that exists [4]. We try to provide as much as customization for the tools and real-time output to the users. We are building a platform that helps users with no or intermediate knowledge in security to help find the vulnerabilities in their server [5].

## 2 Background

A research paper stated few questions and concerns about (a) network vulnerability scanning, (b) security vulnerabilities, (c) Is System Security a Concern? (d) application security [6]. So, we are trying to come up with one solution for fighting against cyberattacks. A description of "Main stages of a pen testing" [7], i.e., Step 1. Planning and preparation, Step 2. Reconnaissance, Step 3. Discovery, Step 4. Analyzing Information and Risks, Step 5. Active Intrusion Attempts, Step 6. Final Analysis, Step 7. Report Preparation. So our work focuses on mainly from steps 2–5 and step 7.

## 3 Related Work

Pen-testing your application server or network has become an important area, and several studies are being developed to improve security in data systems and networks. A live exercise was presented to help students with a live case study to understand the importance and complexity of security. E-voting is getting popular day by day and expects to assess and enhance critical thinking in the field of security [2]. A discussion was made with all the different network security tools that are available and developed in the market that are widely used to test a system network hardening [4]. They have developed one of the most used networks scanning and banner grabbing tools called as Nmap which is widely used and most advanced useful network sniffing tool called Wireshark. A person writes and shows what are the current rate and effectiveness of the current security and vulnerability scanning tools that are available in the market and how much of the prices are we trying to reduce [8]. It compares one of the best to most widely used tools for doing vulnerability assessment and penetration testing. Finding vulnerability has been the target of many of the tools and system available. But they start having a problem with configuration issues and network issues, and a very limited scanning and vulnerability assessment access are there until you have installed on the local system [9]. An implementation was widely used Kali Linux OS on Raspberry Pi [10]. Here we are focusing to bring all tools installed in Kali Linux to online instead of installing it locally or Raspberry Pi. This framework will provide users all features of open-source pen-testing tools (command line tools only) with highly improved performance of the integrated tools.

## 4 Technical Description

### *4.1 Overview*

OS: Ubuntu 16.04, Nginx version: Nginx/1.10.3 (Ubuntu), Django 1.8.7, PostgreSQL Version: (PostgreSQL) 9.5.11, Pgbouncer: pgbouncer version 1.7 (Debian), Chroot Jail: Ubuntu Artful High-Level Design.

We have developed a framework which is responsible for handling the tools and the request that the user is making to make them available in a browser and sending a request to scan the target and synchronously getting output. The Nginx web server is accepting connection request and sending it over to Django web server, where the highly secured front end is responsible for validating user request [11]. Depending upon validation, it passes the request to a chroot jailed environment, where it has very limited environment access and it executes the scan requests and the commands that the user gives and sends the results back to users synchronously [12]. The diagrammatic representation of the framework can be seen in Fig. 1, where the arrow represents the flow of data. If ever an RCE is done or the environment is compromised, we have built a jailed environment where the basic functionality is removed. It cannot revert to main Django server because of the environment access commands, such as chmod, ls, cd, whoamiand muchmore, will not be available [13]. The configuration we have tested are Memory: 1 GB, CPU = 1 vCPU, SSD Disk = 25 GB, Network Speed (Download: 1053.43 Mbit/s, Upload: 1118.61 Mbit/s) Memory: 1 GB, CPU = 1 vCPU, SSD Disk = 25 GB.

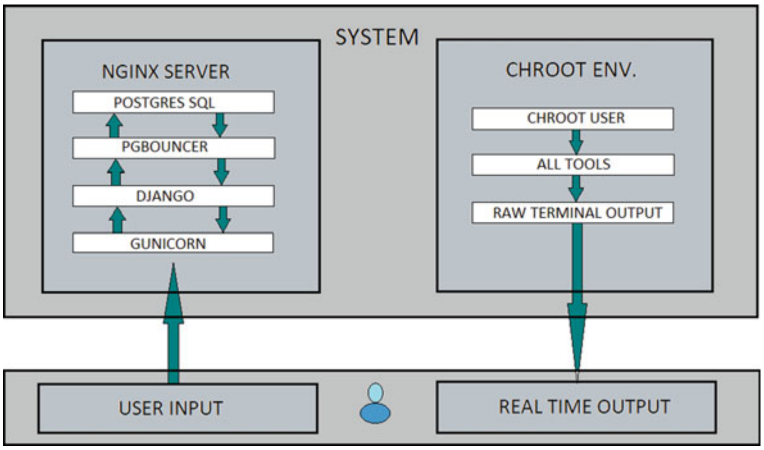


**Fig. 1** Overview of the design

### 4.2 Technology Used

Django is a high-level web development framework built in python that encourages rapid and clean development, pragmatic design keeping security in mind. Built by experienced developers, it takes care of much of the hassle of Web development, so one can focus on writing our application without needing to reinvent the wheel for the development of its security protocols [11]. Its free and open source do same for PostgreSQL and pgbouncer and Nginx [14].

### 4.3 Algorithm Overview

The framework provides the power to customize the scans as far as possible. Every scan request coming from the user is handled by Django, and it creates a shell file of the scan command and starts the shell file to run in a jailed environment and feed with the output to the users at the same time so that they can see the raw results. As soon as the request is sent by the user, the web server checks if the user has a valid session and also check if any other scan is running. The program code is

```
def invokenmap(request):
 if 'username' in request.COOKIES:
  try:
          value = request.COOKIES['username']
          ses_email = request.session['member_id']
  except:
          return HttpResponseRedirect('/ohmygod/?message=Not Allowed')
 else:
  return HttpResponseRedirect('/')
 if authenticate(value ,ses_email) == True:
  if captcha_validation( request.POST.get('g-recaptcha-response')) ==
False:
          return HttpResponseRedirect('/ohmygod/?message=Cannot Validate
Captcha&tags=hidden')
  address=request.POST['ip']
  if address == "":
          return HttpResponseRedirect('/ohmygod/?message=address blank')
  try:
          data=request.POST['flags']
          not_acceptable_strings = ['&&','all',';', '-p-','|','*','-iL',
'-iR','-e','>','>>','--exclude','--excludefile','-sL','--system-dns','-
sI','-p-','--version','--script-updatedb','--script-all','--osscan-
guess','--max-retries','--source-port','-g','--badsum','-oN','-oX','-
oS','-oG','-oA','--resume','--stylesheet','--webxml','--no-
stylesheet','--datadir','--interactive','!sh','127.0.0.1']
          if any(x in address for x in not_acceptable_strings):
                    return HttpResponseRedi-
rect('/ohmygod/?message=Invalid Strings at the Flag parameter')
          if any(x in data for x in not_acceptable_strings):
                    return HttpResponseRedi-
rect('/ohmygod/?message=Invalid Strings at the Flag parameter')
  except Exception:
          data = ""
  value = request.COOKIES['username']
```

```
   if one_at_a_time (value) == False:
           muluser = value + ".txt"
           open(muluser, 'w').close()
           scanner_input = 'nmap '+data+' '+address+'
           make_file ( value , scanner_input)
           myprocess = subprocess.Popen(['schroot -c artful -u django --
directory=/home/django/'+value+'/ -- "./magic.sh"'],shell=True,
stdout=subprocess.PIPE, bufsize=1)
           scan_log (get_ip(request) , address , value , scanner_input)
           t = threading.Thread(target=process_output,
args=(myprocess,muluser,value))
           t.daemon = True
           t.setName(value)
           t.start()
           dbupdate(value , True , myprocess.pid)
           return HttpResponseRedirect("/result")
  else:
         ip = get_ip(request)
         objects = UserDatabase.objects.get(mail=value )
           return ren-
der(request,'ltscan.html',{'name':objects.first_name,'user_ip':ip})
 else:
           return HttpResponseRedirect('/')
```

Another thread starts running and synchronously reads the shell output and starts sending to the user. Restarting process or blocking the user to run multiple scans at the same time, as this can lead to the exhaustive use of resources. The output is formatted to support the HTML front end so the user gets the output synchronously and hassle-free. The below code is responsible for taking the execution output in real time and sending it to the user by sending the output to a file.

```
def process_output(myprocess,muluser,value): #output-consuming thread
    nextline = None
    buf = ''
    while True:
        out = myprocess.stdout.read(1)
        if out == '' and myprocess.poll() != None: break
        if out != '':
            buf += out
            if out == '\n':
                nextline = buf
                buf = ''
        if not nextline: continue
        line = nextline
        nextline = None
        with open(muluser,"a") as test_file:
                    line = line.encode("utf-8")
                line = line + ' <br>'
                    test_file.write(line)
          test_file.close()
myprocess.stdout.close()
```

The tools that we have integrated with the framework has been modified to make it compatible such as not to ask any further input, run independently, remove vulnerable functionality inside the tools source code, and tried to make it as much optimized as possible. We can see the below graph Fig. 2 which shows the performance and

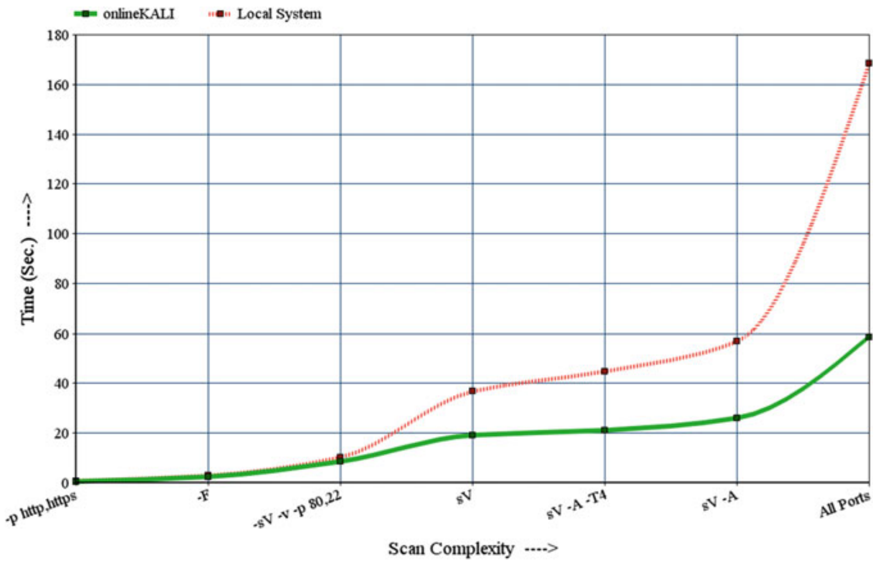


**Fig. 2** Comparison between locally running Nmap against web framework

timings of Nmap scans running from a local machine compared with the framework. We drastically see a performance improvement when running by the framework.

We are also working on building the customized layout of the raw output and also a personalized report for a particular IP with all the necessary vulnerability assessment done with a single click.

## 5 Experimental Results and Findings

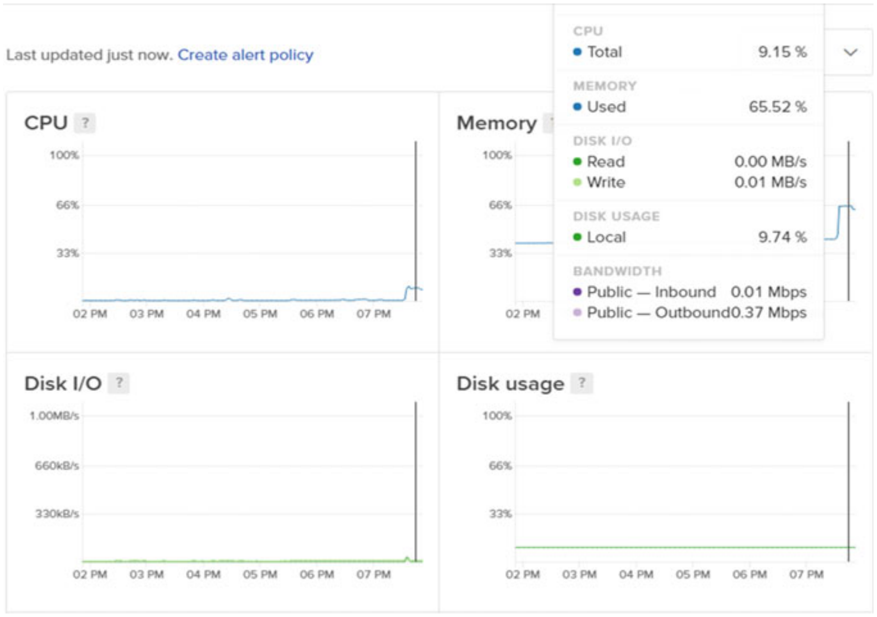
### *5.1 Performance Graph*

Performance of the framework is quite impressive. See the below performance graph on specific machine configuration. Performance level will increase with machine configuration. Currently, we are using the lowest configuration of server available for our development purpose, as soon as we move into our production we hope to have a high-end CPU and RAM with load balancer which may work ever faster and reliable than the current results. We are using servers to test from DigitalOcean [15]

Scan: nmap -A -p- -v scanme.nmap.org

We will show three graphs to explain this section.

As in Figs. 1, 2, 3, we can see the CPU, memory, and disk usage when we are running 10 simultaneous Nmap full port services scan at the same time. The curve in Fig. 3 describes us about the load that the CPU and memory are having when very



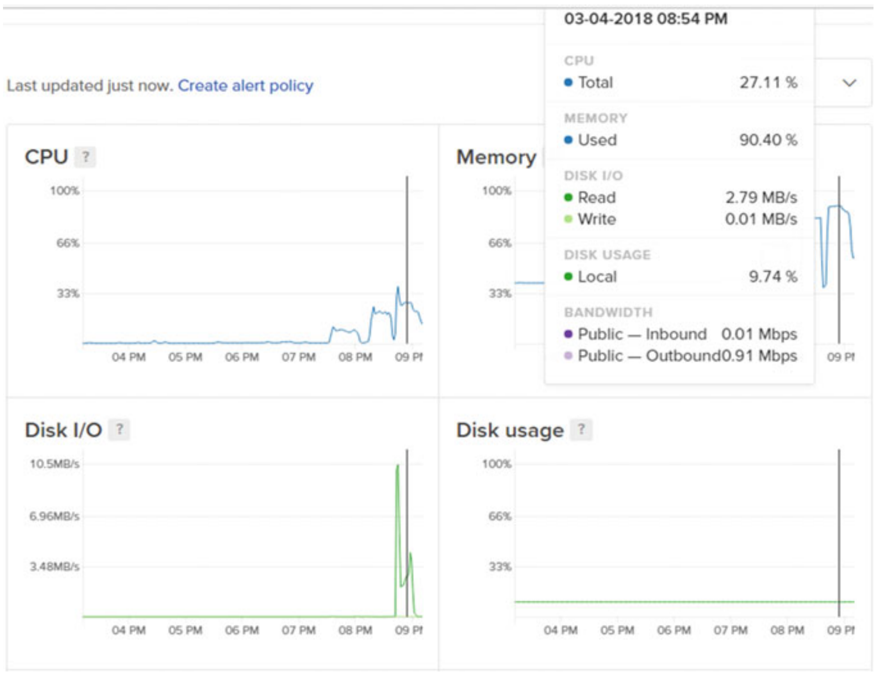
CPU
● Total 9.15 %
MEMORY
● Used 65.52 %
DISK I/O
● Read 0.00 MB/s
● Write 0.01 MB/s
DISK USAGE
● Local 9.74 %
BANDWIDTH
● Public — Inbound 0.01 Mbps
● Public — Outbound0.37 Mbps

**Fig. 3** Performance load graph for 10 simultaneous Nmap scan while scanning on scanme.nmap.org

high complexity scan is running 10 times at same time. Figures 4 and 5 show the load on the system when high complexity scan is running 30 and maximum scans at same time. The scan we choose is of very high complexity that uses the maximum functionality of the Nmap framework, which shows how well our front end is capable of handling load of the request that will be provided.

## *5.2 Security Level*

Security level of this web framework has been kept at very high priority, so we chose the Django framework to develop it. The attacks are launched with a jailed chroot environment, which has all the environment connection ability and user movability and commands are disabled so that if any user by loophole or command injection gets any access, he is jailed in a non-movable environment [12]. The front is based on Django framework which is known as the most secure web development framework currently in the market [11]. We also use the encryption or hashing of some input and output flags and working on currently to put our own custom encryption algorithm to all the messages flowing throughout the system. We have also made some changes to the legacy tools to remove some flags which can impact on the performance and



03-04-2018 08:54 PM

CPU
● Total 27.11 %

MEMORY
● Used 90.40 %

DISK I/O
● Read 2.79 MB/s
● Write 0.01 MB/s

DISK USAGE
● Local 9.74 %

BANDWIDTH
● Public — Inbound 0.01 Mbps
● Public — Outbound0.91 Mbps

**Fig. 4** Performance load graph for 30 simultaneous Nmap scan while scanning on scanme.nmap.org

system performance. This implementation improves the performance of the scanning process of all the tools implemented.

## *5.3 User Experience*

This framework will help cybersecurity professionals (beginner and intermediate) to do a security testing and vulnerability assessment on the go without any hardware configuring issues or network hassle. Figure 6 shows the UI that the framework provides to the user. We recommend the users to have a proper knowledge of what they are doing, and they have the permissions of the things doing through our framework. We do not encourage data loss or hacking of any computer or IOT devices. Continuous research is going on for the advancement of this framework so that it can be used as a single platform for cybersecurity professionals. Also, this framework includes flag field for all the tools with as much as customization that we can provide to the user for input for the doing customized scan. We are also continuously working on to put new tools in our system and compatibility with the framework [16].

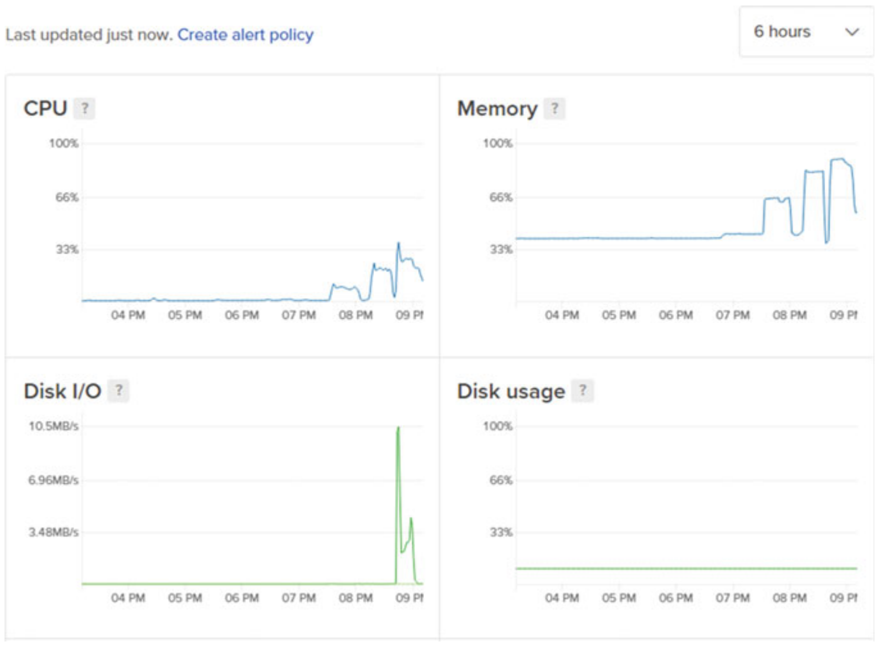
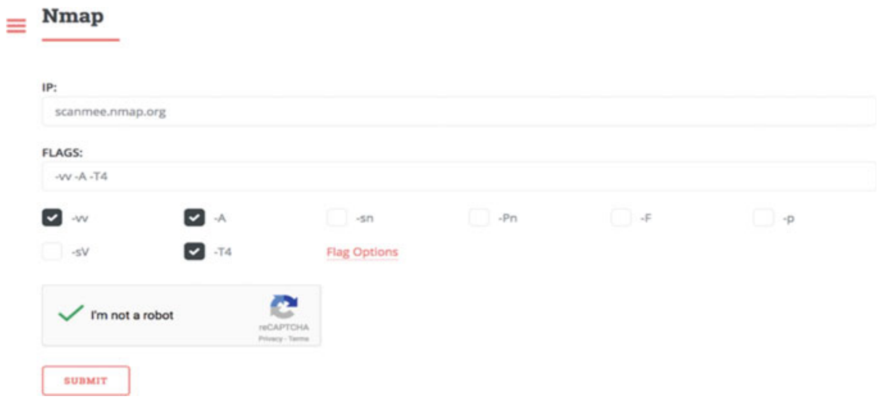


**Fig. 5** Performance load graph for 10–35 simultaneous Nmap scan while scanning on scanme.nmap.org



IP: scanmee.nmap.org

FLAGS: -vv -A -T4

[✓ -vv] [✓ -A] -sn -Pn -F -p
-sV [✓ -T4]

**Fig. 6** Nmap scan page from the framework (flags filed for user input)

## *5.4 Time Consumption*

Time consumption for a particular scan depends on the flags of scan and complexity level. As simple scan would take less time, while full port scan will take maximum time [17]. We also have some time limit for some scans as it can through too much of network hits on a web server which can affect it users or may harm its business functionality. We aim not to cause any disturbance to any organization or business and hope to thrive to help business professions to help to improve the security of their Web site [13].

### *5.5 Export Vulnerability Report*

Further in development.

## 6 Discussion

### *6.1 Limitations and Boundaries*

This framework cannot be implemented on the local network. Only command line tools included in this framework. We are currently developing and making changes to the tools so that the tools can run faster and remove the flags from its basic functionality

And user interaction code so that it can perform smoothly with the framework.

### *6.2 Future Scope*

**Automation**: For future development and research, this framework can be used as an automate vulnerability and penetration testing.
**Agent**: This framework can come up with an agent so that internal network pen test can be performed (still under research process).

## 7 Conclusion

The automated penetration test plays an important role in the security professional's toolkit. As part of a comprehensive security program, these tools can quickly evaluate the security of systems, networks, and applications against a wide variety of threats. But security pros should view them as a supplement, rather than a replacement, for traditional manual testing techniques.

So, this framework is all about bringing all open-source security tools into a single online framework. So only IP/Domain name required in IP field and already basic flags options are there in form of the checkbox to perform a simple quick scan.

As stated before this framework will help cybersecurity professionals (beginner and intermediate) to do a security testing and vulnerability assessment on the go without any hardware configuring issues or network hassle.

**Acknowledgements** This research is supported by HackCieux. We are thankful to our colleagues who provided expertise that greatly assisted the research, although they may not agree with all of the interpretations provided in this paper.

## References

1. Bacudio, A.G., Yuan, X., Bill Chu, B.-T., Jones, M.: An overview of penetration testing. Int. J. Netw. Secur. Appl. (IJNSA) **3**(6) (2011). http://airccse.org/journal/nsa/1111nsa02.pdf
2. Bishop, M., Frincke, D.A.: Achieving Learning Objectives Through E-Voting Case Studies. http://ieeexplore.ieee.org/document/4085594/
3. Web Application Vulnerability Scanner Evaluation Project (Vulnerability Scanner Evaluation Project) (2012). http://code.google.com/p/wavsep/
4. Gordon Lyon. Top 125 Network Security Tools [EB] (2011). http://sectools.org/
5. Chroot Concept. https://en.wikipedia.org/wiki/Chroot
6. Ethical hacking and network defense: choose your best network vulnerability scanning tool. In: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA). http://ieeexplore.ieee.org/document/7929663/
7. de Jimenez, R.E.L.: Pentesting on web applications. In: 2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI). http://ieeexplore.ieee.org/document/7942364/
8. Shay-Chen The Web Application Vulnerability Scanner Evaluation Project [EB] (2012). http://www.sectoolmarket.com/
9. W3af. http://w3af.sourceforge.net/
10. Yevdokymenko, M., Mohamed, E., Arinze, P.O.: Ethical hacking and penetration testing using Rasberry PI. In: 2017 4th International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T). http://ieeexplore.ieee.org/document/8246375/
11. https://www.digitalocean.com/
12. PgBouncer. https://pgbouncer.github.io/
13. DDOS. https://www.arbornetworks.com/research/what-is-ddos
14. Django Server hosted in DigitalOcean. https://www.digitalocean.com/
15. Nginx Web Server Security and Hardening Guide. https://geekflare.com/nginx-webserver-security-hardening-guide/
16. Kali Linux Operating System (Specially designed for Pentesting). https://www.kali.org/
17. Web Application Vulnerability Scanner Evaluation Project (Vulnerability Scanner Evaluation Project) (2012). http://code.google.com/p/wavsep/
18. Offensive Security. https://www.offensive-security.com/
19. Open Source tools list which is inbuilt in Kali Linux. https://tools.kali.org/tools-listing
20. Ngnix. https://nginx.org/en/
21. Django. https://www.djangoproject.com/
22. PostgresSQL. https://www.postgresql.org/