Report Content

Wednesday, January 2, 2019 3:27 PM

Report in its definition is a statement of the results of an investigation or of any matter on which definite information is required

A report should detail the outcome of the test and, if you are making recommendations, document the recommendations to secure any high-risk systems

Repo of public pen testing reports - https://github.com/juliocesarfort/public-pentesting-reports

Report Writing Stages

Due to the comprehensive writing work involved, penetration report writing is classified into the following stages –

- Report Planning
- Information Collection
- Writing the First Draft
- Review and Finalization

From <https://www.tutorialspoint.com/penetration_testing/penetration_testing_report_writing.htm>

Content of Penetration Testing Report

Following is the typical content of a penetration testing report –

Executive Summary
• Scope of work • Project objectives • Assumption • Timeline

- Summary of findings
- Summary of recommendation

Methodology

- Planning
- Exploitation
- Reporting

Detail Findings

- Detailed systems information
- Windows server information

References

- Appendix

From <https://www.tutorialspoint.com/penetration_testing/penetration_testing_report_writing.htm>

1 - Executive Summary for Strategic Direction

The executive summary serves as a high-level view of both risk and business impact in plain English. The purpose is to be concise and clear. It should be something that non-technical readers can review and gain insight into the security concerns highlighted in the report.

While IT staffers may need all the technical details, executives don't need to understand *the technology*. They need to understand business risk, something a good executive summary will effectively communicate. It is imperative that business leaders grasp what's at stake to make informed decisions for their companies, and the executive summary is essential to delivering that understanding.

Visual communication can also be helpful in getting complex points across clearly. Look for graphs, charts, and similar visuals in communicating the summary data provided here.

2 - Walkthrough of Technical Risks

Most reports use some sort of rating system to measure risk, but seldom do they take the time to *explain* the risk. The client's IT department needs to make swift, impactful decisions on how best to resolve vulnerabilities. To do so, they require approval from the people upstairs. To simply state that something is dangerous does not properly convey risk.

For instance, if a critical vulnerability is found allowing file-uploads to a healthcare

portal, there are two ways to report this:

1 – Technically Accurate – Company X's web application does not limit user uploads by file type, creating a vulnerability that allows an attacker to execute arbitrary code remotely and elevate their privilege within the application.

2 – Both Accurate and Contextualized – Company X's web application does not limit user uploads by file type, creating a vulnerability that allows an attacker to execute arbitrary code remotely and elevate their privilege within the application. In this instance, the attacker would be able to view the medical records of any user and operate as an administrator on the application.

The second one has a more weight to it, indicating not only the technical aspects but the business impact as well. The most valuable reports are those that speak to all audience members in the language they understand – especially those in leadership positions.

For instance, if your team finds that a client's healthcare management web portal allows users to upload a profile picture, but does not prevent them from uploading arbitrary code instead, there are essentially two ways to report this:

3 - Potential Impact of Vulnerability

Risk can be broken down into two pieces: likelihood and potential impact.

Likelihood is standard in most assessment reports. Of course, the odds of an exploitation—while important—aren't enough to define risk. You wouldn't rank a deep-seated remote code execution lower than an email address of a developer obviously present in an HTML script. This is because the former would be far more impactful to the client.

If you think you're seeing a theme here, you're not wrong. An assessment report isn't *just* for the IT staff. Executives need to see a break-down of how a vulnerability that anyone could have would directly affect their organization specifically. Factoring both the likelihood and potential impact of an exploitation into the overall risk is a major component in an excellent report.

4 - Multiple Vulnerability Remediation Options

Most penetration test reports will include a generic, high-level description of how to handle these problems; however, these generic "catch-all' remediation guides often fall short when it comes to the unique context of the client's needs. If a client has a vulnerable service running on a webserver that they depend on, the remediation should

offer more than telling them to simply disable the service altogether.

Of course, it's important to let the client know that there's a straightforward method of filtering for SQL injections, or configuring their firewall to block certain attacks. That said, a quality pentest report will give you multiple remediation options that are detailed enough to prepare the client's IT team for a swift resolution. Assuming the internal staff already knows how to remediate all vulnerabilities greatly reduces the value of the penetration test.

Concluding Thoughts

The penetration test itself is not why clients seek out security assessments. The assessment report and client support are. That's exactly why we put so much of our attention and effort into reporting.

Details like verbose descriptions, proper methodology, vulnerability description, and other factors are important as well; implementing these four concepts, in addition, is a recipe for an excellent report.

If your organization's last engagement didn't have these items—or you're just unhappy with the documentation—contact us for a quote or check out our own example reports.

From <https://rhinosecuritylabs.com/penetration-testing/four-things-every-penetration-test-report/>

Information

Friday, January 4, 2019 11:16 PM

- https://www.blackhillsinfosec.com/how-to-not-suck-at-reporting-or-how-to-write-great-pentesting-reports/
- https://resources.infosecinstitute.com/writing-penetration-testing-reports/
- https://www.pluralsight.com/courses/writing-penetration-testing-reports
- https://www.peerlyst.com/posts/how-to-write-a-penetration-testing-report-magda-chelly-ph-d
- https://www.peerlyst.com/posts/how-to-write-a-penetration-testing-report-magda-chelly-ph-d
- https://resources.infosecinstitute.com/writing-penetration-testing-reports/
- https://www.pluralsight.com/courses/writing-penetration-testing-reports
- https://pentestmag.com/course/writing-an-effective-penetration-testing-report-w9-2/

Report Preparation

Report preparation must start with overall testing procedures, followed by an analysis of vulnerabilities and risks. The high risks and critical vulnerabilities must have priorities and then followed by the lower order.

However, while documenting the final report, the following points needs to be considered −

- Overall summary of penetration testing.
- Details of each step and the information gathered during the pen testing.
- Details of all the vulnerabilities and risks discovered.
- Details of cleaning and fixing the systems.
- Suggestions for future security.

Penetration Testing Vs. Vulnerability

Generally, these two terms, i.e., Penetration Testing and Vulnerability assessment are used interchangeably by many people, either because of misunderstanding or marketing hype. But, both the terms are different from each other in terms of their objectives and other means. However, before describing the differences, let us first understand both the terms one-by one.

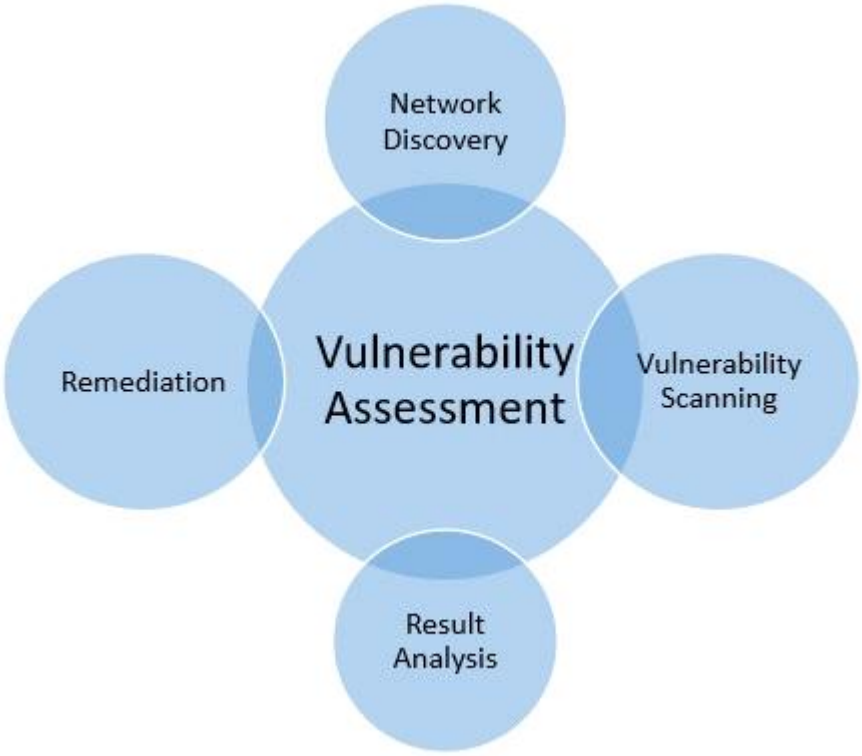
Penetration Testing

Penetration testing replicates the actions of an external or/and internal cyber attacker/s that is intended to break the information security and hack the valuable data or disrupt the normal functioning of the organization. So, with the help of advanced tools and techniques, a penetration tester (also known as ethical hacker) makes an effort to control critical systems and acquire access to sensitive data.

Vulnerability Assessment

On the other hand, a vulnerability assessment is the technique of identifying (discovery) and measuring security vulnerabilities (scanning) in a given environment. It is a comprehensive assessment of the information security position (result analysis). Further, it identifies the potential weaknesses and provides the proper mitigation measures (remediation) to either remove those weaknesses or reduce below the risk level.

The following diagram summarizes the vulnerability assessment −



The following table illustrates the fundamental differences between penetration testing and vulnerability assessments −

Penetration Testing	Vulnerability Assessments
Determines the scope of an attack.	Makes a directory of assets and resources in a given system.
Tests sensitive data collection.	Discovers the potential threats to each resource.
Gathers targeted information and/or inspect the system.	Allocates quantifiable value and significance to the available resources.
Cleans up the system and gives final report.	Attempts to mitigate or eliminate the potential vulnerabilities of valuable resources.
It is non-intrusive, documentation and environmental review and analysis.	Comprehensive analysis and through review of the target system and its environment.
It is ideal for physical environments and network architecture.	It is ideal for lab environments.
It is meant for critical real-time systems.	It is meant for non-critical systems.

Which Option is Ideal to Practice?

Both the methods have different functionality and approach, so it depends upon the security position of the respective system. However, because of the basic difference between penetration testing and vulnerability assessment, the second technique is more beneficial over the first one.

Vulnerability assessment identifies the weaknesses and gives solution to fix them. On the other hand, penetration testing only answers the question that "can anyone break-in the system security and if so, then what harm he can do?"

Further, a vulnerability assessment attempts to improve security system and develops a more mature, integrated security program. On the other hand, a penetration testing only gives a picture of your security program's effectiveness.

As we have seen here, the vulnerability assessment is more beneficial and gives better result in comparison to penetration testing. But, experts suggest that, as a part of security management

system, both techniques should be performed routinely to ensure a perfect secured environment.

Areas of Penetration Testing

Penetration testing is normally done in the following three areas −

- Network Penetration Testing − In this testing, the physical structure of a system needs to be tested to identify the vulnerability and risk which ensures the security in a network. In the networking environment, a tester identities security flaws in design, implementation, or operation of the respective company/organization's network. The devices, which are tested by a tester can be computers, modems, or even remote access devices, etc
- Application Penetration Testing − In this testing, the logical structure of the system needs to be tested. It is an attack simulation designed to expose the efficiency of an application's security controls by identifying vulnerability and risk. The firewall and other monitoring systems are used to protect the security system, but sometime, it needs focused testing especially when traffic is allowed to pass through the firewall.
- The response or workflow of the system − This is the third area that needs to be tested. Social engineering gathers information on human interaction to obtain information about an organization and its computers. It is beneficial to test the ability of the respective organization to prevent unauthorized access to its information systems. Likewise, this test is exclusively designed for the workflow of the organization/company.

In penetration testing, report writing is a comprehensive task that includes methodology, procedures, proper explanation of report content and design, detailed example of testing report, and tester's personal experience. Once the report is prepared, it is shared among the senior management staff and technical team of target organizations. If any such kind of need arises in future, this report is used as the reference.

Report Writing Stages

Due to the comprehensive writing work involved, penetration report writing is classified into the following stages −

- Report Planning
- Information Collection
- Writing the First Draft
- Review and Finalization

Report Planning

Report planning starts with the objectives, which help readers to understand the main points of the penetration testing. This part describes why the testing is conducted, what are the benefits of pen testing, etc. Secondly, report planning also includes the time taken for the testing.

Major elements of report writing are –

- Objectives – It describes the overall purpose and benefits of pen testing.
- Time – Inclusion of time is very important, as it gives the accurate status of the system. Suppose, if anything wrong happens later, this report will save the tester, as the report will illustrate the risks and vulnerabilities in the penetration testing scope during the specific period of time.
- Target Audience – Pen testing report also needs to include target audience, such as information security manager, information technology manager, chief information security officer, and technical team.
- Report Classification – Since, it is highly confidential which carry server IP addresses, application information, vulnerability, threats, it needs to be classified properly. However, this classification needs to be done on the basis of target organization which has an information classification policy.
- Report Distribution – Number of copies and report distribution should be mentioned in the scope of work. It also needs to mention that the hardcopies can be controlled by printing a limited number of copies attached with its number and the receiver's name.

Information Collection

Because of the complicated and lengthy processes, pen tester is required to mention every step to make sure that he collected all the information in all the stages of testing. Along with the methods, he also needs to mention about the systems and tools, scanning results, vulnerability assessments, details of his findings, etc.

Writing the First Draft

Once, the tester is ready with all tools and information, now he needs to start the first draft. Primarily, he needs to write the first draft in the details – mentioning everything i.e. all activities, processes, and experiences.

Review and Finalization

Once the report is drafted, it has to be reviewed first by the drafter himself and then by his seniors or colleagues who may have assisted him. While reviewing, reviewer is expected to check every detail of the report and find any flaw that needs to be corrected.

Content of Penetration Testing Report

Following is the typical content of a penetration testing report –

Executive Summary

- Scope of work
- Project objectives
- Assumption
- Timeline
- Summary of findings
- Summary of recommendation

Methodology

- Planning
- Exploitation
- Reporting

Detail Findings

- Detailed systems information
- Windows server information

References

- Appendix

From <https://www.tutorialspoint.com/penetration_testing/penetration_testing_quick_guide.htm>

Common Problems in Reports

Friday, January 4, 2019 11:13 PM

Problems with penetration testing reports

There are several common complaints from clients related to the penetration testing reports solution providers present at the end of an engagement.

1. The reports are merely copies of the scan results.
Copying vulnerability scan results verbatim from the scanning tool into the report adds little value for the client. It reduces confidence in the report, and in the organization performing the test.

2. The reports do not tell the client how to fix problems.
Some solution providers produce reports that detail the issues discovered during the test, without going into detail about how developers, administrators and security teams can fix the problems. This has much less value to clients, as anyone can tell an organization they have problems! A valuable penetration testing report contains remediation details.

3. The reports do not help clients replicate the problem.
Many clients want to recreate the compromise scenarios themselves. Detailed information about tools used, techniques employed, scripts written and other information can help them to glean more value from the test report.

4. The reports contain false positives.
False positives can fill a penetration testing report with unnecessary data and lead to wasted time. A skilled solution provider can often eliminate or at least reduce false positives, producing a concise and valuable report.

Suggestions for valuable penetration testing reports

Given these issues, there is much a solution provider can do to deliver greater value in pen test reports. The following suggestions should help improve the quality of reports in most cases:

- Translate results from network and vulnerability scanners into customized language that is tailored specifically to the client being tested. For example, a client's business environment, risk concerns and priorities, and any specific testing parameters (user profiles and system types) should be included.
- For any issues discovered during the test, provide concrete evidence of the compromise via screenshots or "flags" planted on compromised systems. Many organizations will prohibit the planting of flags (often text files or images), so screenshots will be the most effective evidence of successful attack completion.

- Manually check all discovered vulnerabilities for false positives to ensure reports are as accurate as possible.
- Include extensive advice on how to address and remediate discovered vulnerabilities. By categorizing vulnerabilities as patching and configuration issues, coding errors, weak authentication scenarios, etc., solution providers can tailor advice to client's IT teams best suited to perform the remediation steps.
- Describe the tools and tactics employed at each phase of the test, and whether the test was successful in compromising systems or applications. For example, listing the output of reconnaissance tools like Google search queries and Paterva's Maltego, specific scanning commands with open source tools such as NMAP, Hping and Scapy, and exact sequences of variables chosen with exploitation tools like Metasploit (a free tool now owned by Rapid 7), will be invaluable for recreating and validating the issues themselves internally. In most cases, this data should be included as appendices to the main report so as not to clutter the report with extensive technical detail.
- Focus the report on specific risks or concerns important to the client. In some cases, clients will be looking for a general overview of network and application vulnerabilities, but tailoring the report to compliance initiatives, particular security controls, and sensitive data specific to the organization, may improve the report's overall impact.

  Penetration tests represent a client's security posture at a point in time, and should ideally be used to demonstrate exactly what vulnerabilities are present and the inherent risks based on these vulnerabilities. By customizing and tailoring the results to the client, as well as providing more remediation guidance and testing details, solution providers can substantially increase the value of the test and final report for any client.

From <https://searchitchannel.techtarget.com/tip/Wow-your-client-with-a-winning-penetration-testing-report>

Note Organization

Wednesday, January 2, 2019 5:59 PM

- https://lifehacker.com/back-to-basics-perfect-your-note-taking-techniques-484879924
- https://www.oxfordlearning.com/5-effective-note-taking-methods/
- https://lifehacker.com/back-to-basics-perfect-your-note-taking-techniques-484879924
- https://medium.goodnotes.com/the-best-note-taking-methods-for-college-students-451f412e264e
- https://www.educationcorner.com/note-taking.html

https://www.google.com/search?rlz=1C1GCEB_enUS786US786&ei=OVAtXMP2BM26ggfWhZPACg&q=OSCP+note+taking&oq=OSCP+note+taking&gs_l=psy-ab.3..0i71l8.7836.10193..10333...0.0..0.0.0.....0....1..gws-wiz.q6OW-SpYZR4

Homepage
Luke's Ultimate OSCP Guide: Part 2 — Workflow and documentation tips
Go to the profile of Luke Stephens (@hakluke)
Luke Stephens (@hakluke)
Feb 15, 2018

Man walks through door with large shadow. OFFENSIVE security logo dramatically appears in a red abyss.
At the start of my labs, I wasn't really prepared for how much documentation I would be writing. I opted to document ALL of the exercises and 10 lab machines (while this is not compulsory, it earns you an extra 5 points in the exam). On top of that, I also documented the exam boxes. In total, this came to around 280 pages. Within the first couple of weeks, my reporting process changed dramatically, and I wasted a lot of time getting a good workflow happening.

Hopefully, you can make the most of your lab time by learning from my mistakes. Read on, fellow hackers!

Environment
The Virtual Machine
Offsec provides you with a modified Kali Linux virtual machine. I'm not going to run through setting it up here, but I will say this:

The virtual machine provided by Offsec is meant to be run with VMWare, not VirtualBox.

Generally, I'm more of a VirtualBox guy. It's free, and I like Vagrant. Unfortunately, converting a VMWare VM to VirtualBox, and vice versa, is dodgy. If you're using Linux or Windows as your host machine — you're in luck! Download VMWare workstation for free and install away. Unfortunately for Apple users — you will need to buy VMWare Fusion, or convert the Kali PWK VM to VirtualBox and use that. I used VirtualBox for the entire time including the exam. It worked pretty well but I ran into issues copy/pasting between the the host and guest machines. After being converted, the Kali PWK VM also lagged a lot more than the regular Kali in VirtualBox — perhaps that is just an isolated incident though.

To update? Or not to update?
If you've spent much time trawling the #offsec channel on IRC, or any OSCP related chats, you will know that some of the most common problems arise from people updating their Kali VM. Be careful about what you update because it will cause problems with the exercises, running apt upgrade will break stuff.

One exception to the rule is searchsploit which you can update by doing a searchsploit -u.

If you really want to check out the latest upgrades to Kali, I recommend having a seperate Kali VM which is fully updated, not the PWK Kali version.

Sharing Files
Sharing files between your host machine and your Kali PWK VM is important for two reasons. Firstly, you'll find yourself wanting to share your documentation between the two regularly. Secondly, it makes backups easier.

Protip: Create a shared folder between your host and guest machine. Put the shared folder on the host machine inside a folder which syncs to the cloud (such as Dropbox or Google Drive). That way, your files are automatically being backed up to the cloud as you create them.

Another protip: Got some tools or files that you use regularly? Pop them in a private git repo. If you bork your VM, it won't take long to spin up a new one and pull down your custom tools.

Applications that will make your life easier
Shutter: For taking screenshots. When you take the screenshot, it's automatically added to the clipboard for an easy paste into your docs.
CherryTree: CherryTree took over when KeepNote stopped being maintained. It's much better for a number of reasons. Ditch KeepNote and use CherryTree instead. CherryTree is also included in the latest Kali, while KeepNote is not.
Terminator: Mainly for the lovely split-screen terminals.
Burp: From memory, it's not covered in much detail in the PWK course. It's my favourite tool for hacking webapps and it comes pre-installed. There's plenty of tutorials out there for learning how to use it. Try YouTube.
Documentation
The Workflow
My main machine is a humble 13" Macbook Air from 2015. It's powerful enough for me, but it's no super-computer. Once my documentation reached about 80 pages of screenshot heavy material in "Pages" (the apple equivalent of Microsoft Word), my computer became so slow that it was virtually unresponsive. From then on my documentation process changed, I used CherryTree within Kali to create the initial documentation and screenshots without any formatting. When it came time to format it nicely, I would export the entire CherryTree file to HTML, share it with my host machine, copy the text over to Google docs, and add some nice formatting. Voila!

CherryTree Setup
CherryTree allows for hierachical notetaking. I found it useful to set up the notes the same way as the network — with each subnet as a parent node, then machines as subnodes. If you want to go even more hierachical, you could create another layer under this for each stage of the hack. Perhaps "Enumeration", "Exploitation", "Privilege Escalation" and "Flags" would work well for you.

I also made a few top level nodes. One was a "cheatsheet" which kept a lot of command syntaxes that were hard to remember, one was "credentials" which stored credentials I had found throughout the network that I might want to try later, and one was called "flyover", which contained my full network enumeration scans, DNS enumeration, a log of which boxes I had already owned, and which ones were yet to fall.

What should be included in the documentation?
Basically, we can split the documentation in to three parts:

The lab exercises (Not compulsory, but will earn an extra 5 points in the exam if you submit these

alongside a write-up of 10 lab machines)
10 lab machines (Not compulsory, but will earn an extra 5 points in the exam if you submit these alongside the lab exercises write-up)
The exam machines (Compulsory!)
When you're documenting the lab exercises, you need to show just enough to document the steps you took to achieve the task. No more, no less. If in doubt, leave it in — because it's better to have too much documentation than miss an important step!

When documenting machines (both in the labs AND in the exam), you need to include every step taken to exploit the machine including screenshots if necessary. There is no need to include things you tried that didn't work. You must also include one single screenshot which displays the output of the proof.txt file, the output of ifconfig/ipconfig, and the output of whoami, id, or a similar command to demonstrate your current user.

Official Support
If you don't trust me, or if you want more info — the official support for the course documentation requirements can be found here. You can also contact the admins to clarify anything you're not 100% on.

You can also download a sample lab report here, although I didn't use it in the end, it didn't really fit with my reporting style.

RTFM
The most important part of all this is to read the official documentation on how to report, the correct formats, etc. If the report is amazing but you send it in the wrong format, you will automatically receive a fail. Read it and read it again!

If you read these two pages 10 times over, you should be safe:

Exam requirements (also contains reporting requirements): https://support.offensive-security.com/#!oscp-exam-guide.md
PWK support page: https://support.offensive-security.com/#!pwk-support.md
Where are the other parts to this guide?
If you haven't already read part 1 of my "Ultimate Guide to OSCP" series, it's here. Part 3 includes my approach to hacking new machines in the labs, a cheatsheet, and some other useful hacking tricks. You can find that here.

Get in touch
If you have any suggestions or would like to stay in touch — the best way is to follow me on Twitter.

OscpDocumentationWorkflowInfosecPenetration Testing
Go to the profile of Luke Stephens (@hakluke)
Luke Stephens (@hakluke)
Pentester | Hubby | Musician | On a mission to free my thoughts and actions from the limits which are imposed on them by society.

More from Luke Stephens (@hakluke)
How Do Hacking Challenges and Certifications Differ From An Actual Hacking Job?
Go to the profile of Luke Stephens (@hakluke)
Luke Stephens (@hakluke)
Related reads
Waldo Write-up (HTB)
Go to the profile of George O

George O
Related reads
Bounty Write-up (HTB)
Go to the profile of George O
George O
Responses

Templates

Friday, January 4, 2019 11:49 PM

Penetration Testing Report Templates

Leave a reply

Sample Penetration Test Report by Offensive Security— An excellent report by an excellent team.
www.offensive-security.com/offsec-sample-report.pdf

Writing a Penetration Testing Report — Probably one of the best papers on this subject. It was written by Mansour A. Alharbi for his GIAC certification. The author starts with *report development stages*, then describes the *report format* and ends it with a *sample report*.
http://www.sans.org/reading_room/whitepapers/bestprac/writing-penetration-testing-report_33343

Report Template— A report template by vulnerabilityassessment.co.uk
www.vulnerabilityassessment.co.uk/report%20template.html
PDF version:
http://www.okdhs.org/NR/rdonlyres/EDC02492-637C-4C45-B305-35856EBEE8DF/0/SecurityPenetrationTestResultsRprttemp_EPMO_05052009.pd

Penetration Testing Report— Sample report by niiconsulting.com
http://www.niiconsulting.com/services/security_assessment/NII_Sample_PT_Report.pdf

Penetration Test Report— Another good sample report
www.besnard.org/biometrics/2BIO706_business_report.pdf

Penetration Test Report— Sample OSSAR report
www.digitalencode.net/ossar/ossar_v0.5.pdf

penetration testing report template— Template by logicallysecure.com
http://www.logicallysecure.com/resources/downloads/penetration%20testing%20report%20template.doc

Cross-posted from EthicalHacker.net

From <https://darrylmacleod.wordpress.com/2012/03/26/penetration-testing-report-templates/>

https://github.com/Salesforce/Vulnreport - Tool released by Salesforce

Vulnreport
Pentesting management and automation platform
Vulnreport is a platform for managing penetration tests and generating well-formatted, actionable findings reports without the normal overhead that takes up security engineer's time. The platform is built to support automation at every stage of the process and allow customization for whatever other systems you use as part of your pentesting process.

Vulnreport was built by the Salesforce Product Security team as a way to get rid of the time we spent writing, formatting, and proofing reports for penetration tests. Our goal was and continues to be to build great security tools that let pentesters and security engineers focus on finding and fixing vulns.

For full documentation, see http://vulnreport.io/documentation

Deployment
Vulnreport is a Ruby web application (Sinatra/Rack stack) backed by a PostgreSQL database with a Redis cache layer.

Vulnreport can be installed on a local VM or server behind something like nginx, or can be deployed to Heroku.

Local Deploy / Your own server
To deploy locally, you'll need to make sure you have installed the dependencies:

Ruby >= 2.1
PostgreSQL
Redis
Rollbar
Bundler
Clone the repo and open up the .env file, updating it as necessary. The run bundle install. You'll probably want to modify start.sh to make it work for your environment - the one included in the repo is intended to be used for local use during debugging/development.

You should also create a .env file based on .env.example, or set the same ENV variables defined in .env in your environment.

Heroku Deploy
Automatic Deployment
Deploy

You can automatically deploy to Heroku. After doing so, follow the instructions below to login to Vulnreport and finish configuration.

Manual Deployment
To deploy to Heroku (assuming you have created a Heroku app and have the toolbelt installed)

git clone [Vulnreport repo url]

heroku git:remote -a [Heroku app name]

heroku addons:create heroku-postgresql:hobby-dev
heroku addons:create heroku-redis:hobby-dev
heroku addons:create rollbar:free
heroku addons:create sendgrid:starter
You'll then want to open up the .env file and copy the keys/values (updating values where necessary) to the Heroku settings for your app. This can also be done via the toolbelt CLI commands. Note that the default ENV variables after running the addons should be fine, but you can double check. You'll definitely want to update VR_SESSION_SECRET. If this isn't your production install, you should change RACK_ENV to development.

heroku config:set VR_SESSION_SECRET=abc123456
heroku config:set RACK_ENV=production

git push heroku master
You can now follow the instructions for installation as you would if you were running Vulnreport locally.

Installation
To handle the initial configuration for Vulnreport, run the SEED.rb script. If you are deploying on Heroku, run this via heroku run ./SEED.rb.

If you used the automated 'Deploy to Heroku' feature, this step should have been handled for you automatically.

Running ./SEED.rb on ⬢ vulnreport-test... up, run.8035

Vulnreport 3.0.0.alpha seed script
WARNING: This script should be run ONCE immediately after deploying and then DELETED

Setting up Vulnreport now...

Setting up the PostgreSQL database...
Done

Seeding the database...
Done

User ID 1 created for you

ALL DONE! :)
Login to Vulnreport now and go through the rest of the settings!

Now, delete the SEED.rb file.

The default admin user has been created for you with username admin and password admin. This should be immediately rotated and/or SSO should be configured.

At this point you should go to your Vulnreport URL (e.g. https://my-vr-test.herokuapp.com above) and login with the user created. Go through the Vulnreport and user settings to configure your instance of Vulnreport.

Pentest!
You're ready to go - for documentation about how to use your newly-installed Vulnreport instance, see the full docs at http://vulnreport.io/documentation

Custom Interfaces and Integrations
Vulnreport is designed and intended to be used with external systems. For more information about how to implement the interfaces that allow for integration/synchronization with external systems please see the custom interfaces documentation at http://vulnreport.io/documentation#interfaces.

Code Documentation
To generate the documentation for the code, simply run Yard:

yard doc
yard server
A Note on XML Import/Export
Currently, Vulnreport supports an XML format to import Vulns to a specific Test. This is useful if you want Vulnreport to be on a different network than you do your pentests on and thus are using a different client to record findings while you actively pentest, but relies on being configured for your specific Vulnreport instance and Vulntypes configuration.

We're working on supporting a few other types of XML import (ZAP and Burp, for instance) as well as allowing arbitrary XML export/import between Vulnreport instances. Stay tuned as we hope to push these features soon.

The XML format Vulnreport currently supports is:

```
<?xml version="1.0" encoding="UTF-8"?>

<Test xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <Vuln>
  <Type>[Vulntype ID]</Type>
  <File>[File Vuln Data]</File>
  <Code>
   [Code Vuln Data]
  </Code>
  <File>clsSyncLog.cls</File>
  <Code>
   hello world
  </Code>
  ...etc...
 </Vuln>

 <Vuln>
  <Type>6</Type>
  <File>clsSyncLog.cls</File>
  <File>CommonFunction.cls</File>
  <Code>
   12 Public Class CommonFunction{
  </Code>
 </Vuln>
</Test>
<?xml version="1.0" encoding="UTF-8"?>

<Test xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"">
 <Vuln>
  <Type>REQUIRED - EXACTLY 1 - INTEGER - ID of VulnType. 0 = Custom</Type>
  <CustomTypeName>OPTIONAL - EXACTLY 1 - STRING if TYPE == 0</CustomTypeName>
  <BurpData>OPTIONAL - UNLIMITED - STRING - Burp req/resp data encoded in our
protocol</BurpData>
  <URL>OPTIONAL - UNLIMITED - STRING - URL for finding</URL>
  <FileName>OPTIONAL - UNLIMITED - STRING - Name/path of file for finding</FileName>
  <Output>OPTIONAL - UNLIMITED - STRING - Output details</Output>
  <Code>OPTIONAL - UNLIMITED - STRING - Code details</Code>
  <Notes>OPTIONAL - UNLIMITED - STRING - Notes for vuln</Notes>
  <Screenshot>
   OPTIONAL - UNLIMITED - Screenshots of vuln
   <Filename>REQUIRED - EXACTLY 1 - STRING - Filename with extension</Filename>
   <ImageData>
    REQUIRED - EXACTLY 1 - BASE64 - Screenshot data
   </ImageData>
  </Screenshot>
 </Vuln>

 ....unlimited vulns....

 <Vuln>
 </Vuln>
</Test>
```

Report Template

- Introduction
- Date carried out
- Testing Team details
- Name
- Contact Nos.
- Relevant Experience if required.
- Network Details
- Peer to Peer, Client-Server, Domain Model, Active Directory, integrated
- Number of Servers and workstations
- Operating System Details
- Major Software Applications
- Hardware configuration and setup
- Interconnectivity and by what means i.e. T1, Satellite, Wide Area Network, Lease Line Dial up etc.
- Encryption/ VPN's utilized etc.
- Role of the network or system
- Scope of test
- Constraints and limitations imposed on the team i.e. Out of scope items, hardware, IP addresses.
- Constraints, limitations or problems encountered by the team during the actual test
- Purpose of Test
- Deployment of new software release etc.
- Security assurance for the Code of Connection
- Interconnectivity issues.
- Type of Test
- Compliance Test
- Vulnerability Assessment
- Penetration Test
- Test Type
- White-Box
- The testing team has complete carte blanche access to the testing network and has been supplied with network diagrams, hardware, operating system and application details etc, prior to a test being carried out. This does not equate to a truly blind test but can speed up the process a great deal and leads to a more accurate results being obtained. The amount of prior knowledge leads to a test targeting specific operating systems, applications and network devices that reside on the network rather than spending time enumerating what could possibly be on the network. This type of test equates to a situation whereby an attacker may have complete knowledge of the internal network.
- Black-Box
- No prior knowledge of a company network is known. In essence an example of this is when an external web based test is to be carried out and only the details of a website URL or IP address is supplied to the testing team. It would be their role to attempt to break into the company website/ network. This would equate to an external attack carried out by a malicious hacker.
- Grey-Box
- The testing team would simulate an attack that could be carried out by a disgruntled, disaffected staff member. The testing team would be supplied with appropriate user level privileges and a user account and access permitted to the internal network by relaxation of specific security policies present on the network i.e. port level security.
- Executive Summary (Brief and Non-technical)
- OS Security issues discovered with appropriate criticality level specified
- Exploited
- Causes
- Hardware failing
- Software failing
- Human error

- Unable to exploit – problem area
- Causes
- Hardware failing
- Software failing
- Human error
- Application Security issues discovered with appropriate criticality level specified
- Exploited
- Unable to exploit – problem area
- Physical Security issues discovered with appropriate criticality level specified
- Exploited
- Unable to exploit – problem area
- Personnel Security issues discovered with appropriate criticality level specified
- Exploited
- Unable to exploit – problem area
- General Security issues discovered with appropriate criticality level specified
- Exploited
- Unable to exploit – problem area
- Technical Summary
- OS Security issues discovered
- File System Security
- Details of finding
- Example: A FAT partition was found. FAT by default does not give the ability to set appropriate access control permissions to files. In addition moving files to this area removes the protection of the current ACLs applied to the file.
- Recommendation and fix
- Example: Format the file system to NTFS.
- Password Policy
- Details of finding
- Example: LM Hashes found still being utilized on the network.
- Recommendation and fix
- Example: Ensure NTLM2 is enforced by means of the correct setting in Group Policy.
- Auditing Policy
- Details of finding
- Example: Logon success and failure was not enabled
- Recommendation and fix
- Example: Amend appropriate Group Policy Objects and ensure it is tested and then applied to all relevant Organizational Units etc.
- Patching Policy
- Details of finding
- Example: Several of the latest Microsoft patches were found to be missing
- Recommendation and fix
- Example: Ensure a rigorous patching policy is instigated after first being tested on a development LAN to ensure stability. Review the settings on the WSUS server and ensure that it is regularly updated and an appropriate update strategy is instigated for the domain.
- Anti-virus Policy
- Details of finding
- Example: Several workstations were found to have out of date anti-virus software. In addition where it was found to be installed the actual product was found to be mis-configured and did not provide on-access protection.
- Recommendation and fix
- Example: Ensure all workstations are regularly updated and configured correctly to ensure maximum protection is afforded
- Trust Policy
- Details of finding
- Example: Users from one domain were unable to access resources on another tree.

- Recommendation and fix
- Example: Review transitive and non-transitive trusts and ensure that all relevant trusts have been established.
- Web Server Security
- File System Security
- Details of finding
- Example: i.e. Incorrect permission on www root Recommendation and fix
- Example: Apply more stringent permissions or remove various users/groups that currently have access to this area.
- Password Policy
- Details of finding
- Example: Areas of the website that should be Protected did not have any password mechanism enforced.
- Recommendation and fix
- Example: Ensure areas that require access to be limited are password protected.
- Auditing Policy
- Details of finding
- Example: Web server logs were not being reviewed for illicit behaviors.
- Recommendation and fix
- Example: Regularly review all audit logs.
- Patching Policy
- Details of finding
- Example: The latest patch was not applied to the server leaving it susceptible to a Denial of Service Attack.
- Recommendation and fix
- Example: Apply the latest patch after testing on a development server to ensure compatibility with installed applications and stability of the server is maintained.
- Lockdown Policy
- Details of finding
- Example: The IIS lockdown tool has not been applied to the web server.
- Recommendation and fix
- Example: Apply the IIS lockdown tool to the server after first testing on a development server to ensure compatibility with installed applications and stability of the server is maintained.
- Database Server Security
- File System Security
- Details of finding
- Example: Loose access control permissions were found on directories containing important configuration files that govern access to the server.
- Recommendation and fix
- Example: Ensure stringent access control permissions are enforced.
- Password Policy
- Details of finding
- Example: Clear text passwords were found stored within the database.
- Recommendation and fix
- Example: Ensure all passwords, if required to be stored within the database are encrypted and afforded the maximum protection possible.
- Auditing Policy
- Details of finding
- Example: Reviewing the audit logs from the TNS Listener were not being carried out.
- Recommendation and fix
- Example: Ensure all relevant audit logs are regularly inspected. Audit logs may give you the first clue to possible attempts to brute force access into the database.
- Patching Policy
- Details of finding
- Example: The latest Oracle CPU was not installed, leaving the system susceptible to multiple buffer

and heap overflows and possible Denial of Service attacks.

- Recommendation and fix
- Example: Install the latest Oracle CPU after first testing on a development server to ensure adequate compatibility and stability.
- Lockdown Policy
- Details of finding
- Example: Numerous extended stored procedures were directly accessible by the public role.
- Recommendation and fix
- Example: Ensure the public role is revoked from all procedures that direct access is not required or utilized.
- Trust Policy
- Details of finding
- Example: Clear text Link passwords were discovered.
- Recommendation and fix
- Example: Ensure all Link passwords are encrypted, review the requirement to utilize these Links on a regular basis.
- General Application Security
- File System Security
- Details of finding
- Recommendation and fix
- Password Policy Details of finding
- Recommendation and fix
- Auditing Policy
- Details of finding
- Recommendation and fix
- Patching Policy
- Details of finding
- Recommendation and fix
- Lockdown Policy
- Details of finding
- Recommendation and fix
- Trust Policy
- Details of finding
- Recommendation and fix
- Business Continuity Policy
- Backup Policy
- Details of finding
- Recommendation and fix
- Replacement premises provisioning
- Details of finding
- Recommendation and fix
- Replacement personnel provisioning
- Details of finding
- Recommendation and fix
- Replacement software provisioning
- Details of finding
- Recommendation and fix
- Replacement hardware provisioning
- Details of finding
- Recommendation and fix
- Replacement document provisioning
- Details of finding
- Recommendation and fix
- Annexes

- Glossary of Terms
- Buffer Overflow
- Normally takes the form of inputting an overly long string of characters or commands that the system cannot deal with. Some functions have a finite space available to store these characters or commands and any extra characters etc. over and above this will then start to overwrite other portions of code and in worse case scenarios will enable a remote user to gain a remote command prompt with the ability to interact directly with the local machine.
- Denial of Service
- This is an aimed attacks designed to deny a particular service that you could rely on to conduct your business. These are attacks designed to say overtax a web server with multiple requests which are intended to slow it down and possibly cause it to crash. Traditionally such attacks emanated from one particular source.
- Directory Traversal
- Basically when a user or function tries to "break" out of the normal parent directory specified for the application and traverse elsewhere within the system, possibly gaining access to sensitive files or directories in the process.
- Social Engineering
- Normally uses a limited range of distinct subject matter to entice users to open and run an attachment say. Usually associated with phishing/E-mail type attacks. The main themes are:
  - Sexual – Sexual ideas/pictures/websites,
  - Curiosity – Friendly themes/appealing to someone's passion or obsession,
  - Fear – Reputable sources/virus alert,
  - Authority – Current affairs/bank e-mails/company e-mails.
- SQL Injection etc.
- Basically when a low privileged user interactively executes PL/SQL commands on the database server by adding additional syntax into standard arguments, which is then passed to a particular function enabling enhanced privileges.
- Network Map/Diagram
- Accompanying Scan Results – CD-ROM
- Vulnerability Definitions
- Critical
- A vulnerability allowing remote code execution, elevation of privilege or a denial of service on an affected system.
- Important
- A security weakness, whose exploitation may result in the compromise of the Confidentiality, Integrity or Availability of the company's data.
- Information Leak
- Insecure services and protocols are being employed by the system allowing potentially allowing unrestricted access to sensitive information i.e.:
  a. The use of the Finger and Sendmail services may allow enumeration of User IDs.
  b. Anonymous FTP and Web based services are being offered on network devices or peripherals.
  c. Disclosure of Operating System, Application version details and personal details of system administration staffs.
- Concern
- The current systems configuration has a risk potential to the network concerned though the ability to exploit this is mitigated by factors such as default configuration, auditing, or the difficulty level or access level required to carry out an exploit. This includes the running of network-enabled services that are not required by the current business continuity process.
- Unknowns
- An unknown risk is an unclear response to a test or an action whose impact can be determined as having minimal impact on the system. The test identifying this risk may or may not be repeatable. While the results do not represent a security risk per see, they should be investigated and rectified where possible. Unknowns may also be due to false positives being reported, however, do require follow up response.
- Details of Tools Utilized.
- Methodology Utilized.
- Reconnaissance

- The tester would attempt to gather as much information as possible about the selected network. Reconnaissance can take two forms i.e. active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection etc. afforded to the network. This would usually involve trying to discover publicly available information by utilizing a web browser and visiting newsgroups etc. An active form would be more intrusive and may show up in audit logs and may take the form of an attempted DNS zone transfer or a social engineering type of attack.
- Enumeration
- The tester would use varied operating system fingerprinting tools to determine what hosts are alive on the network and more importantly what services and operating systems they are running. Research into these services would then be carried out to tailor the test to the discovered services.
- Scanning
- By use of vulnerability scanners all discovered hosts would be tested for vulnerabilities. The result would then be analyzed to determine if there any vulnerabilities that could be exploited to gain access to a target host on a network.
- Obtaining Access
- By use of published exploits or weaknesses found in applications, operating system and services access would then be attempted. This may be done surreptitiously or by more brute force methods. An example of this would be the use of exploit engines i.e. Metasploit or password cracking tools such as John the Ripper.
- Maintaining Access
- This is done by installing a backdoor into the target network to allow the tester to return as and when required. This may be by means of a rootkit, backdoor trojan or simply the addition of bogus user accounts.
- Erasing Evidence
- The ability to erase logs that may have detected the testing teams attempts to access the network should ideally not be possible. These logs are the first piece of evidence that may prove that a possible breach of company security has occurred and should be protected at all costs. An attempt to erase or alter these logs should prove unsuccessful to ensure that if a malicious attacker did in fact get access to the network then their every movement would be recorded.
- See Penetration Test Framework for more details
- Sources of Information
- National Security Agency
- Microsoft
- Microsoft Windows 2000 Security Configuration Guide.
- Windows Server 2003 Security Guide.
- Windows XP Security Guide.
- The Threats and Countermeasures guide.
- Auscert
- CISSecurity
- ISO27001/2
- Sarbanes Oxley
- HIPAA

P

From <http://hackingandsecurity.blogspot.com/2016/07/pen-test-report-template.html>

Cherrytree

Saturday, January 5, 2019 12:16 AM

Log all commands of the current session

`script $target.log`
....
commands and output of commands you ran in that 1 terminal sesssion
....
`exit # when finished`
Use Cherrytree or OneNote other to document findings...even a text file!
Create a screenshot of the selected area and save it at home directory

`shift Print Screen`
Set the Target IP Address to the $ip system variable

`export ip=target_ip`
If you're working on a single target it is useful to do the `export ip=target_ip` command before you run Tmux. That way when you create new tabs in Tmux you don't have to run the export command for every new tab.

Tmux Configuration/kali-configuration/tmux-config
Terminator Configuration/kali-configuration/terminator-shortcuts

Eyewitness

Saturday, January 5, 2019 12:16 AM

https://www.christophertruncer.com/eyewitness-usage-guide/

I typically call EyeWitness, provide it a text file (with each URL on a new line), and let it run. If I have a .nessus file or nmap.xml output, and it has more than 350 URLs, I'll run EyeWitness with the –createtargets flag (explained below), and output all the targets to a single text file. I typically then split that file up into roughly 300 URLs per text file, and then either script up EyeWitness to run one after another, or run scans simultaneously. However, different situations might cause EyeWitness to need to be used in a different manner, so hopefully this EyeWitness usage guide can help explain all of its features.

Python:

The bare bones, and likely most common, use of EyeWitness is to provide a single URL, or multiple URLs within a file for EyeWitness to screenshot and generate a report. To provide a single URL, just use the –single flag as follows:

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py --single http://www.google.com
```

EyeWitness also accepts files for providing the URLs. The file can be provided in the following formats:

- Single text file with a URL on each line
- Nmap XML output
- .Nessus file
- amap file output

To perform a scan, using any of these filetypes, just provide the -f flag as follows:

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt
```

By default, EyeWitness will attempt to screenshot the website, and have a max timeout of 7 seconds. If it takes longer than 7 seconds to render the website, EyeWitness will skip to the next URL. If you wish to change the timeout of EyeWitness, use the -t flag and set it to the max number of seconds you want it to wait to render a website.

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt -t 15
```

Once EyeWitness has finished navigating to all URLs, and has generated a report, EyeWitness outputs the report to the same directory EyeWitness is in, and names it based off of the date and time the scan ran. If you want to change the directory name that EyeWitness outputs its report to, use the -d flag and provide the name. When using the -d flag, you can provide just a name, and EyeWitness will create the report using the provided name within the same directory as EyeWitness. You can also provide the full path to a directory, and EyeWitness will create the report folder at that location (just make sure you have the proper write permissions).

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt -d reportout
```

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt -d /tmp/reportout2
```

Sorted reporting was a feature brought up to me by Jason Frank (@jasonjfrank) as something that would be helpful when reviewing the EyeWitness report. If we had a way to make EyeWitness analyze the different web applications, and group similar web apps together, then it would be easy to quickly sort through/review the groups you want to target. We envisioned similar printers, mirrored web pages, etc. all grouped together within the report. Lucky for us, Rohan Vazarkar (@cptjesus) worked on adding this feature in. His pull request was merged in on April 22nd, and EyeWitness will now attempt to sort all results based off of their title within each report generated.

The –localscan option was added based on a request from David McGuire (@davidpmcguire). We wanted a way to perform some basic port scanning for web servers once a machine has been compromised. Currently, one way to do it is to drop Nmap on the compromised machine, but

if we did that, we'd have to install winpcap on the machine, which requires admin rights. Instead of this, you can drop the windows Eyewitness binary, and provide the –localscan option with a CIDR range to scan. EyeWitness will then try to find any ip listening on 80, 443, 8080, and 8443 within the provided range. All live hosts listening on any of those ports will be added to a file that can be fed back into EyeWitness.

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py --localscan 192.168.
1.0/24

################################################################################
#                                EyeWitness                                    #
################################################################################

[*] Scanning 192.168.1.0 on port 80
[*] Scanning 192.168.1.0 on port 443
[*] Scanning 192.168.1.0 on port 8080
[*] Scanning 192.168.1.0 on port 8443
```

The –createtargets option came about when I wanted to have EyeWitness just provide me a list of all web servers from the XML output of Nmap or Nessus. All web servers that EyeWitness finds within Nmap's xml output, or the nessus file will be added to a file containing the target servers. Just provide the filename you want the your targets file to be called.

```
root@VG-kali:~/gitrepos/EyeWitness# ./EyeWitness.py -f web.xml --createtargets n
mapout.txt

################################################################################
#                                EyeWitness                                    #
################################################################################

Creating text file containing all web servers...
Target file created (nmapout.txt).
```

The user agent definition and cycling came about from working with Micah Hoffman (@webbreacher), Robin Wood (@digininja), and Chris John Riley (@ChrisJohnRiley). After a lot of discussion on how best to carry out user agent switching and comparison, the feature was added in. First, you can simply provide the –useragent option, and it will use any string you provide as the user agent.

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt --useragent Mozilla/4.0
```

You can also use the –cycle option along with either browser, mobile, crawler, scanner, misc, or all. When using this option, EyeWitness makes a baseline request. It will then make subsequent requests with user agents of the "type" you specified. If the subsequent requests deviate "too much" from the baseline request, the subsequent request will be added in to the report letting you know it was different from the baseline. The deviation is currently based on the length of the source code the web server provides to EyeWitness. By default, the deviation that's used to measure if the requests are different is set to 50. To change this value, use the –difference flag and provide the new value to use.

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py --cycle all --single
 christophertruncer.com

################################################################################
#                                EyeWitness                                    #
################################################################################

Trying to screenshot http://christophertruncer.com
[*] Now making web request with: wget1.9.1
[*] Now making web request with: MSIE6.0
[*] Hit timeout limit when connecting to: http://christophertruncer.com
[*] Now making web request with: OperaMobile12.02
```

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt --cycle Browser --difference 75
```

Finally, the –jitter option was one that was discussed about at a NovaHackers meeting, and also requested by @ruddawg26. To use this option, provide all the scan parameters you would normally provide, but add on the –jitter parameter at the end, and provide the base number of seconds that it deviates from. Now, EyeWitness will randomize the order of the URLs provided (via text or XML), and will also have a random delay between each request.

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f ct.xml --jitter 30
```

```
################################################################################
#                                EyeWitness                                    #
################################################################################

Trying to screenshot 4 websites...

Attempting to capture: http://198.23.154.4:80  (1/4)
[*] Hit timeout limit when connecting to: http://198.23.154.4:80
[*] Sleeping for 26.4 seconds...
```

Finally, EyeWitness has a –open flag. If you provide the –open flag, each URL passed into EyeWitness will also be opened up in a web browser. Your command string might look similar to the following:

```
root@VG-kali:/mnt/hgfs/gitrepos/EyeWitness# ./EyeWitness.py -f test.txt --open
```

Ruby:

```
################################################################################
#                                EyeWitness                                    #
################################################################################

Usage: [options]
    -f, --filename file.txt          File containing URLs to screenshot
        --nessus file.nessus         Nessus .nessus file output
        --nmap file.xml              Nmap XML file output
    -s, --single URL                 Single URL to screenshot
        --skip-sort                  Do not group similar pages together
        --no-dns                     Parse nmap XML and only use IP address,
                                     instead of DNS for web server
        --createtargets Filename     Create a file containing web servers
                                     from nmap or nessus output.

    -t, --timeout 7                  Maximum number of seconds to wait while
                                     requesting a web page.
        --jitter 15                  Number of seconds to use as a base to
                                     randomly deviate from when making requests.
```

To generate a report for a single website, you need to use the -s or –single flag and provide the URL.

For file based input, you will need to specify the filetype that you are providing. If giving just a normal text file with each URL on a new line, use the -f or –filename switch. If using providing Nmap XML output, you'll need to use the –nmap flag, and .nessus based input requires the –nessus flag.

The –skip-sort flag is used to tell EyeWitness to not auto-group similar web pages together in the report. This can be helpful if you want to see report pages as they are available, instead of waiting until the very end. However, if this flag is used, similar pages will not be grouped together.

The –no-dns flag is used when you want EyeWitness to find web servers via their IP address, not their DNS name, while parsing Nmap XML output.

From <https://www.christophertruncer.com/eyewitness-usage-guide/>

Tmux and Terminator

Saturday, January 5, 2019 12:16 AM

Tmux Configuration

Last updated 16 days ago

I've started to use Terminator instead of Tmux

Terminator Configuration
/kali-configuration/terminator-shortcuts

Anyway - here is the tmux config which worked for me.

```
nano /root/.tmux.conf

# 0 is too far from ` ;)
set -g base-index 1

# Automatically set window title
set-window-option -g automatic-rename on
set-option -g set-titles on

#set -g default-terminal screen-256color
set -g status-keys vi
set -g history-limit 10000

setw -g mode-keys vi
setw -g mode-mouse on
setw -g monitor-activity on

bind-key v split-window -h
bind-key s split-window -v

bind-key J resize-pane -D 5
bind-key K resize-pane -U 5
bind-key H resize-pane -L 5
bind-key L resize-pane -R 5

bind-key M-j resize-pane -D
bind-key M-k resize-pane -U
bind-key M-h resize-pane -L
bind-key M-l resize-pane -R

# Vim style pane selection
bind h select-pane -L
bind j select-pane -D
bind k select-pane -U
bind l select-pane -R
```

```
# Use Alt-vim keys without prefix key to switch panes
bind -n M-h select-pane -L
bind -n M-j select-pane -D
bind -n M-k select-pane -U
bind -n M-l select-pane -R

# Use Alt-arrow keys without prefix key to switch panes
bind -n M-Left select-pane -L
bind -n M-Right select-pane -R
bind -n M-Up select-pane -U
bind -n M-Down select-pane -D

# Shift arrow to switch windows
bind -n S-Left  previous-window
bind -n S-Right next-window

# No delay for escape key press
set -sg escape-time 0

# Reload tmux config
bind r source-file ~/.tmux.conf

# THEME
set -g status-bg black
set -g status-fg white
set -g window-status-current-bg white
set -g window-status-current-fg black
set -g window-status-current-attr bold
set -g status-interval 60
set -g status-left-length 30
set -g status-left '#[fg=green](#S) #(whoami)'
set -g status-right '#[fg=yellow]#(cut -d " " -f 1-3 /proc/loadavg)#[default] #[fg=white]%
H:%M#[default]'
```

Privilege Escalation - Previous
Untitled

Next - Kali Configuration
Terminator Configuration

From <https://guide.offsecnewbie.com/kali-configuration/tmux-config>

Terminator Configuration

Last updated yesterday

I've started to use Terminator instead of tmux - and I actually prefer it.

Setup

In Preferences:

Infinite scrollback is selected
Profiles>colors>Change palette to "White on Black"
Profiles>Background>Solid Color

Google Search Plugin

https://github.com/msudgh/terminator-search

Shortcuts

Ctrl + Shift + O = Virtual Split
Ctrl + Shift + E = Horizontal Split

Ctrl + Shift + Z = Maximizes a current tabbed window to full screen and then restores to tabbed by pressing again
Ctrl + Shift + T = Opens a new tab

Ctrl + Shift + C = Copy to clipboard
Ctrl + Shift + V = Paste
double click on tab name to rename

zsh configuration

nice looking terminal with syntax highlighting

nano ~/.zshrc

```
# If you come from bash you might have to change your $PATH.
# export PATH=$HOME/bin:/usr/local/bin:$PATH

# Path to your oh-my-zsh installation.
export ZSH="/root/.oh-my-zsh"

# Set name of the theme to load --- if set to "random", it will
# load a random theme each time oh-my-zsh is loaded, in which case,
# to know which specific one was loaded, run: echo $RANDOM_THEME
# See https://github.com/robbyrussell/oh-my-zsh/wiki/Themes
ZSH_THEME="agnoster"

# Set list of themes to pick from when loading at random
# Setting this variable when ZSH_THEME=random will cause zsh to load
# a theme from this variable instead of looking in ~/.oh-my-zsh/themes/
# If set to an empty array, this variable will have no effect.
# ZSH_THEME_RANDOM_CANDIDATES=( "robbyrussell" "agnoster" )

# Uncomment the following line to use case-sensitive completion.
# CASE_SENSITIVE="true"

# Uncomment the following line to use hyphen-insensitive completion.
# Case-sensitive completion must be off. _ and - will be interchangeable.
# HYPHEN_INSENSITIVE="true"

# Uncomment the following line to disable bi-weekly auto-update checks.
```

```
# DISABLE_AUTO_UPDATE="true"

# Uncomment the following line to change how often to auto-update (in days).
# export UPDATE_ZSH_DAYS=13

# Uncomment the following line to disable colors in ls.
# DISABLE_LS_COLORS="true"

# Uncomment the following line to disable auto-setting terminal title.
# DISABLE_AUTO_TITLE="true"

# Uncomment the following line to enable command auto-correction.
# ENABLE_CORRECTION="true"

# Uncomment the following line to display red dots whilst waiting for completion.
# COMPLETION_WAITING_DOTS="true"

# Uncomment the following line if you want to disable marking untracked files
# under VCS as dirty. This makes repository status check for large repositories
# much, much faster.
# DISABLE_UNTRACKED_FILES_DIRTY="true"

# Uncomment the following line if you want to change the command execution time
# stamp shown in the history command output.
# You can set one of the optional three formats:
# "mm/dd/yyyy"|"dd.mm.yyyy"|"yyyy-mm-dd"
# or set a custom format using the strftime function format specifications,
# see 'man strftime' for details.
# HIST_STAMPS="mm/dd/yyyy"

# Would you like to use another custom folder than $ZSH/custom?
#ZSH_CUSTOM=~/.oh-my-zsh/custom/plugins

# Which plugins would you like to load?
# Standard plugins can be found in ~/.oh-my-zsh/plugins/*
# Custom plugins may be added to ~/.oh-my-zsh/custom/plugins/
# Example format: plugins=(rails git textmate ruby lighthouse)
# Add wisely, as too many plugins slow down shell startup.
plugins=(
git
colored-man-pages
zsh-syntax-highlighting
zsh-autosuggestions
)
source $ZSH/oh-my-zsh.sh

# User configuration

# export MANPATH="/usr/local/man:$MANPATH"

# You may need to manually set your language environment
# export LANG=en_US.UTF-8
```

```
# Preferred editor for local and remote sessions
# if [[ -n $SSH_CONNECTION ]]; then
#   export EDITOR='vim'
# else
#   export EDITOR='mvim'
# fi

# Compilation flags
# export ARCHFLAGS="-arch x86_64"

# ssh
# export SSH_KEY_PATH="~/.ssh/rsa_id"

# Set personal aliases, overriding those provided by oh-my-zsh libs,
# plugins, and themes. Aliases can be placed here, though oh-my-zsh
# users are encouraged to define aliases within the ZSH_CUSTOM folder.
# For a full list of active aliases, run `alias`.
#
# Example aliases
# alias zshconfig="mate ~/.zshrc"
# alias ohmyzsh="mate ~/.oh-my-zsh"
alias ss="searchsploit"
alias l='ls -la'
alias webup='python -m SimpleHTTPServer 80'
```

Reporting Tools

Friday, January 4, 2019 11:21 PM

Draids
Dradis is an open source browser-based application, which can be used to combine the output of different tools and generate a report. It is extremely easy to use and comes preinstalled with Kali. However, running it may show errors. So, we will reinstall it and then learn how to use it.

Using MagicTree
MagicTree is a data management and reporting tool similar to Dradis. It is preinstalled on Linux and it organizes everything using a tree and node structure. It also allows us to execute commands and export the results as a report. In this recipe, we will look at some of the things we can do using MagicTree to ease our pentesting task.

Generating reports using Dradis and MagicTree

Introduction
In this chapter, we will go through one of the most important steps of a pentesting project, the report. A good report must contain every detail of the vulnerability. Our agenda is to keep it as detailed as possible, which may help the right person in the department understand all the details and work around it with a perfect patch.

There are different ways to create a pentesting report. In this chapter, you will learn a few tools that we can use to create a good report that covers everything in detail. Let's look at some of the key points that should always be included in the report:

- Details of the vulnerability
- The CVSS score
- Impact of the bug on the organization
- Recommendations to patch the bug
- Common Vulnerability Scoring System (CVSS) is a standardized method for rating IT vulnerabilities and determining the urgency of a response.
- You can read more about CVSS at https://www.first.org/cvss.

Generating reports using Dradis
Dradis is an open source browser-based application, which can be used to combine the output of different tools and generate a report. It is extremely easy to use and comes preinstalled with Kali. However, running it may show errors. So, we will reinstall it and then learn how to use it.

How to do it…
Following is the recipe for using Dradis:

1. First, we need to install the dependencies by running the following commands:
apt-get install libsqlite3-dev
apt-get install libmariadbclient-dev-compat
apt-get install mariadb-client-10.1

apt-get install mariadb-server-10.1
apt-get install redis-server

2. We then use the following command:
git clone https://github.com/dradis/dradis-ce.git

3. Then, we change our directory:
cd dradis-ce/

4. Now we run the following command:
bundle install --path PATH/TO/DRADIS/FOLDER

5. We run this command:
./bin/setup

6. To start the server, we run this:
bundle exec rails server

7. We can access Dradis on https://localhost:3000 now.
8. Here, we can set up our password to access the framework and log in with the password:
9. We will be redirected to the dashboard:
10. The free version of Dradis supports plugins of various tools such as Nmap, Acunetix, and Nikto.
11. Dradis allows us to create methodologies. It can be considered a checklist, which can be used while performing a pentest activity for an organization:
12. To create a checklist, we go to Methodologies and click on Add new:
13. We then assign a name and click on Add to Project:
14. We should now see a sample list created for us. We can edit it by clicking on the Edit button on the right-hand side:
15. Here, we see that the list is created in XML. We can edit and save it by clicking on Update methodology:
16. Now let's look at how we can organize our scan reports better. We go to the nodes option on the left-hand side menu and click on the + sign; a pop-up box will open and we can add a network range and then click on Add:
17. To add a new subnode, we select the node from the left-hand side pane and then choose the Add subnode option. This can be used to organize a network-based activity based on the host's IP addresses.
18. Next, we can add notes and screenshots as PoC of the bugs we find:
19. We can even import results of various tools to Dradis. This can be done by choosing Upload Output from tool from the top menu:
20. Here, we upload our output file. Dradis has inbuilt plugins, which can parse reports of different tools:
21. Once the import is done, we will see the results on the left-hand side pane under the title plugin output:
22. We can see the output of the scan results we just imported:
23. Similarly, different scans can be imported and combined together and can be exported as one single report using the Dradis framework:

More information on Dradis can be found on the official website at https://dradisframework.com/.

How to do it…
Following is the recipe for using MagicTree:

1. We can run it from the Application menu.
2. We accept the terms and the application will open up:
3. Next, we create a new node by going to Node | AutoCreate:
4. In the box that opens, we type the IP address of the host we want to be added.
5. Once the node is added, it will appear in the left-hand side pane:
6. To run a scan on a host, we go to the Table View; at the bottom, we will see an input box titled Command:
7. We will run an Nmap scan on the host we just added.
8. MagicTree allows you to query the data and send it to the shell. We click on the Q* button, and it will automatically select the hosts for us:
9. Now we just need to type the following command:
nmap -v -Pn -A -oX $results.xml $host
The following screenshot shows the output of the preceding command:
10. Since hosts are already identified, we do not need to mention them here. Then, we click on Run:
11. We will see a window that shows the scan being executed along with the output.
Once the scan is complete, we can click on Import, and it will be imported intothe tool.
12. Similarly, we can run any other tool and import its report to MagicTree. We cangenerate a report by navigating to Report | Generate Report...:
13. In the next window, we can browse the list of templates we would like to use tosave the report:
14. Then, we click on the Generate Report button, and we will see a report beinggenerated:

There are other tools that can be used for report generation, such as the following:
Serpico: https://github.com/SerpicoProject/Serpico
Vulnreport: http://vulnreport.io/

NMAPgrapher

Saturday, January 5, 2019 5:42 AM

NMAPgrapher can be found at https://github.com/attactics/NMAPgrapher

What is it?

A tool I created to assist me in providing supplementary data with my penetration test reports. I'm releasing it to the public to do as you wish with it. The primary purpose of this tool is to allow the user to quickly and easily output exploratory data from NMAP XML files. The tool is capable of generating the following:

- Most and least common services
- Most and least common ports
- Most and least common operating systems
- Hosts with most and least number of open ports
- HTML document with tables including each host and open services / ports

Most of the data sets can be output to:

- PNG
- SVG
- CSV
- HTML

In order to make this tool more useful to others, I have added the ability to skin the HTML output with CSS styles, found in the css folder. In order to generate the included template I used csstablegenerator. If you want to create your own css styles, just make sure to name the css class name 'table'.

SVG and PNG graphs can be styled by modifying NMAPgrapher.py. I tried to use pygal's CSS styles and have a separate file for editing, however cairosvg was not liking it.

Quick Primer on Use

While NMAPgrapher has a fairly extensive help menu, here is the general usage structure:

```
NMAPgrapher.py [inputFile] [outputBaseName] [outputType]
```

where:

- **inputFile** is the NMAP XML file you wish to process
- **outputBaseName** is the base name of the output files. All output files will be placed in a subfolder from where you ran NMAPgrapher
- **outputType** is the type of output you want (csv, png, html, svg)

The default behavior is to produce all possible outputs for the output type specified. Some outputs such as the host lost detailing all open ports and services is only output in HTML for the time being. This will automatically be output in HTML regardless of the type you specify.

There are also a number of optional flags to only output certain types of data, for more information on these please invoke `NMAPgrapher.py -h`. They should be pretty straight forward.

Dependencies

NMAPgrapher requires a number of libraries to operate. I recommend using pip to install them. You can install them with pip like so:

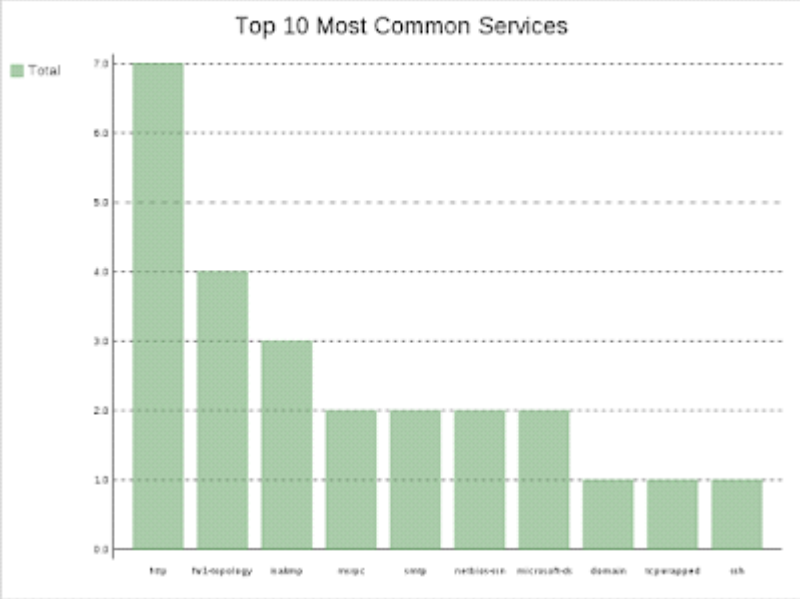
```
pip install pygal cairosvg cssselect tinycss lxml elementtree
```

Bugs and Feature Requests

I'm not a python ninja (...yet?). I know the code is not the prettiest and I will work on refining it as time allows. If there are any bugs or feature requests, please message me on twitter (@evasiv3) and I'd be glad to help.

Sample Graph and Table Screenshots

The following are two example outputs from NMAPgrapher. The visuals are customizable.



192.168.1.1
TCP tcp/80 http
TCP tcp/323 fw1-topology
TCP tcp/443 http
TCP tcp/500 isakmp

192.168.1.2
TCP tcp/25 smtp
TCP tcp/135 msrpc
TCP tcp/139 netbios-ssn
TCP tcp/445 microsoft-ds

192.168.1.1
TCP tcp/80 http
TCP tcp/264 fw1-topology
TCP tcp/443 http
TCP tcp/500 isakmp

From <http://hackingandsecurity.blogspot.com/2017/07/nmapgrapher-tool-to-generate-graph-and.html>