

eLearnSecurity

- Reporting guide -



V1

Armando Romeo

eLearnSecurity © 2010

Summary

Summary.....	2
1. Introduction	3
1.1. Contracts and legal issues	3
1.1.1. Non disclosure and no compete.....	3
1.1.2. Memorandum of Understanding.....	4
1.1.3. Data retention	4
1.2. Rules of engagement	5
2. Reporting	6
2.1. Introduction.....	6
2.2. Structure of a report	7
2.2.1. Executive Summary.....	8
2.2.2. Vulnerability report.....	13
2.2.3. Remediation plan.....	21
2.2.4. Logs	24
3. Conclusions	25

1. Introduction

When you are hired to test the security of networks and applications, you are asked to provide:

- A comprehensive overview of the client's state of the security
- An exhaustive and detailed survey of the security issues you encountered
- The best possible solutions to the above

Your client, and sometimes even your boss, are not aware of penetration testing techniques, exploitation schemes or tools.

Whether you are employed as a penetration tester or you are a freelance you should be able to understand what your counterpart is asking you and what is expecting from you.

A good understanding of the client's expectations at the moment of signing the contract is a milestone that you cannot miss.

1.1. Contracts and legal issues

The client may begin your business relationship with giving you a contract for what their expectations and requirements are for you to do business with them. It is very important to review this contract in detail with Legal Counsel in order to fully understand what is acceptable to the company you will be working with and any limitations they may put on you.

1.1.1. Non disclosure and no compete

These contracts generally contain Non-Disclosure agreements which protect the client (the organization contracting you) from you making any information regarding the company information public, or using their name in any press releases without their consent.

You have to understand that non disclosure agreements pertain not only data included in the report, but also any data that you as a penetration tester will have access to during your engagement.

Employing a strict policy on data leakage on your penetration testing environment is critical in these cases: full disk encryption, physical access control to your machines, patched and up to date software and so on.

Another thing to look for in any contract is a No Compete clause.

No Compete clauses are generally used to ensure you do not do work with any competitor to an organization. While normal contracts may not carry a No Compete clause, some consulting engagements have them as standard language.

If there is a No Compete clause, be sure to get your legal counsel to assist you and ensure that this clause does not preclude you from being able to gain employment at other organizations for which your business may actively solicit.

Also understand that this conduct by your client is very common in certain environment and it is not a mistrust act against you.

1.1.2. Memorandum of Understanding

Memorandum of Understanding (MOU) can be used in cases where the client has regulatory requirements they must maintain for mostly government contracts, but can also be used for non-government entities.

An MOU is basically a brief agreement as to what roles and responsibilities of the entities are for each party, what laws are used within the agreement, and what connectivity is acceptable between the customers. MOUs are similar to Rules of Engagement, but not quite as detailed. An MOU is most often used for roles and responsibilities establishment.

1.1.3. Data retention

During your penetration test you will find and produce a considerable amount of data including:

- Correspondence (email, letters...)
- Graphs, papers, electronic documents
- Logins, passwords, IP addresses, personal data...
- Proof of concepts, exploits code and vulnerabilities
- Screenshots
- Reports and deliverables
- Tools logs

The disclosure of this data, whether you have signed a non-disclosure clause or not, can pose the client's business at high risk. This is also the data that your client's competitors may go after.

It is important to agree with your client the data retention period from your side: how long will you be able to keep this data and how. The less the retention period, the less is the risk for you of losing you client's data. We advise you to destroy data that will not be useful for later engagements with the same client and to encrypt the rest. Two factor authentication (especially biometrics) and encryption is something you should definitely employ in your penetration testing lab.

1.2. Rules of engagement

Rules of engagement documents are the cornerstone of a penetration testers toolkit.

The RoE will give the customer a sense of what you do, how you do it, and what they can expect in return. RoE documents should be very detailed and include your Methodology for testing, the tools that will be used, the tests to be conducted, any access, equipment, or connectivity requirements, and any other provisions a pen-tester would need to complete an engagement.

If possible give the customer a sense of the length of the engagement so they will know when they can expect the engagement to end. Lastly, be sure to detail in full the documentation that will be provided to the client and when they can expect any documentation within the engagement.

This will be the most important document other than the contract that will be exchanged with you and the customer, so take your time and create a comprehensive template and have it looked over by legal counsel to ensure that it is thorough and most importantly, binding.

In a Rules of engagement document you and your client should define the following aspects at a minimum:

- Targets in scope (IP's, domains, servers, departments...)
- Time-frame for the tests (beginning-ending date, hours of the day to conduct the tests...)
- Each person participating to the tests
- Contacts (Phone numbers and emails) that you can call for emergencies during the tests
- Rendez-vous points over-time (weekly progress status notifications...)
- Deliverables, and their level of depth
- Methodology followed
- Tools used
- Agreement on employment of social engineering or other delicate techniques such as password cracking and denial of service

Timing of the tests and scope are two very important constraints that you must take into consideration throughout the whole engagement.

While the tools used in the penetration test are most of the times not an issue (as long as they do not break other terms of the contract in their functioning), violation of scope and timing can really bring you against a court.

2. Reporting

2.1. Introduction

The reporting phase is extremely important in a penetration test.

Regardless of how well you conducted your penetration test, the deliverables of the project is what the client will judge you upon. This part should not be overlooked.

Reporting involves writing and presenting skills, so if you're not familiar with office and productivity suites, this is the right time to start.

The reporting phase is not the last part of your engagement. Your contract can include an amount of hours of consultancy over your findings. During this consultancy period, your client (usually the departments entitled to perform the suggested fixes) will contact you, if necessary, to ask for more information or clarifications.

Including even a small set of hours of post-report consultancy will be greatly appreciated by the client and can set you apart from other competitors.

Reporting begins with the penetration test itself. The sooner you begin to collect your information, the faster will be your reporting phase.

You should have already clear that maintaining your penetration testing data organized makes your testing easier and more accurate. This especially applies to the reporting phase.

While performing a penetration testing session you should save:

- Exact timestamp (including your time zone) of the beginning of the test
- Scope of the test for this session (in terms of IP's or domains, or areas of a website and so on)
- Findings
 - Type of vulnerability
 - Exploit used
 - Vulnerable IP/software/domain/page
 - Brief description
- Eventual notes, useful while reporting

If you collect the above and store it within a database, an excel sheet or even on the filesystem and organize it on a per-target basis, you will find writing your report much more accurate and faster.

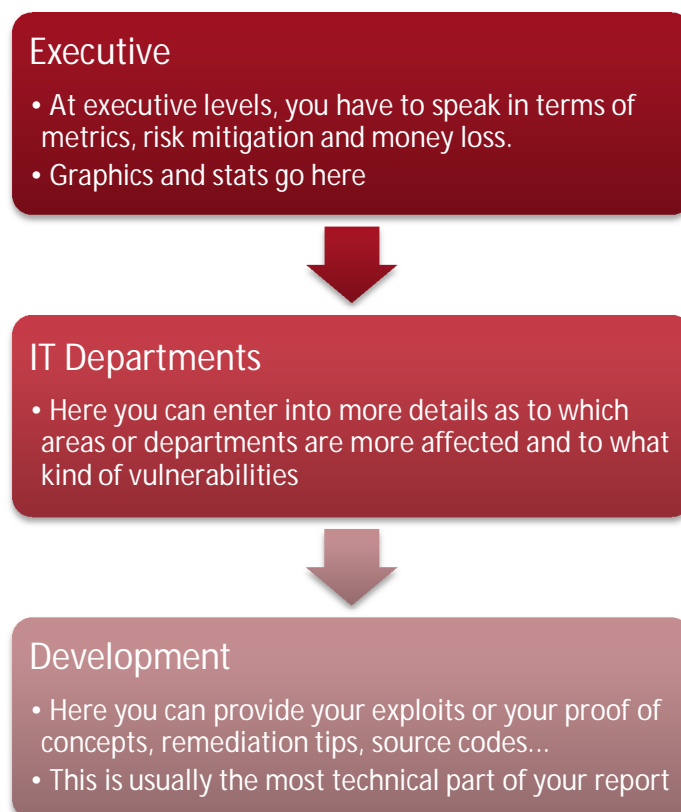
2.2. Structure of a report

A report is the document that contains the history and the result of your project.

As stated in the introduction of this guide, you cannot expect your client to understand your language. You have to make sure that you address all the layers of your clients organization with the right arguments and the appropriate language.

You have to understand the different levels of your client hierarchy you are going to deal with. If you are employed as penetration tester in a security company, you will probably have this function covered by another colleague. It is still important for you to understand what is involved in providing what really matters to your client.

If you want to be a successful and professional penetration tester you have to be able to talk to each layer of your client hierarchy with their own language.



In most of the cases you will deal with these three layers. In smaller assignments you will only deal with two (the Executive and the Development). Either way, make sure to ask the recipients of your deliverables in order to produce a targeted and relevant report.

2.2.1. Executive Summary

The executive summary is the first part of every penetration testing report.

It should not be longer than a few pages and should sum up, at a higher level and without the use of jargon, the overall outcome of your penetration test.

If you agreed to use metrics meaningful to the organization, make sure that you make the current level of security, according to this metric, visible in the first two pages of your executive summary.

The executive summary should absolutely not be pages and pages of text.

The executive level of a company has no time to read your philosophical approach to security. Nor it is interested into which open source "pwnsauce" tool you have used.

You should instead talk by graphs, charts, stats and tables. Text should only be used to explain your charts and to give a final judgment on the status of the security.

Stats relevant to this section of your report are:

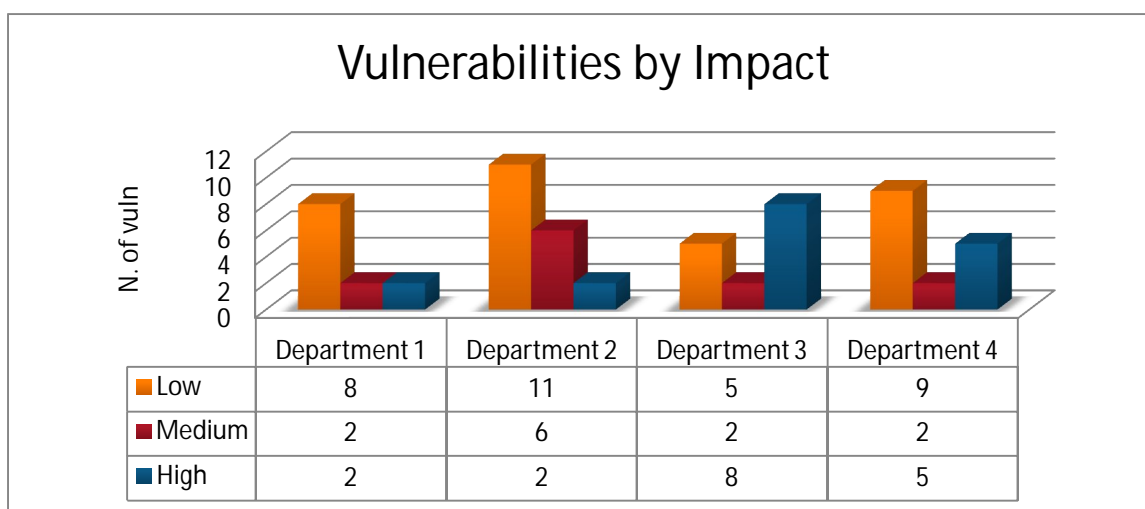
2.2.1.1. *Overall risk level of the targets in scope*

This should be your judgment taking into account the impact of the vulnerabilities found.

Typical qualitative levels are High, Medium, Low. You can safely skip this part if better metrics are used by the organization.

2.2.1.2. *Vulnerabilities by impact*

This is to drill down the risk level and entering more into the details of what kind of impact, exploited vulnerabilities may have on the organization.



The “impact” of a vulnerability is an abused and misused parameter within a penetration testing report. Assigning an impact to a vulnerability in a vacuum is meaningless from a risk assessment point of view.

Since we are talking to the executive level, where “managing risk” is their everyday job, you don’t want to make mistakes on this.

The first thing to do in a penetration test engagement is to understand:

- What the organization does
- Who are the stakeholders
- Who is keeping the organization alive (namely the customers)
- Which assets are the most important for the above

From the above considerations you can understand what impact may a SQL injection have on an isolated community forum compared to a XSS in the company administration panel.

According to [OWASP TOP 10 – 2010](#), the first has more impact than the latter.

However from the 2010 version of the top list you also find another column in the impact formula, the business impact:



This is an extremely important variable: you have to understand the impact of a vulnerability taking into consideration also the assets that it is going to impact in the event of a successful exploitation.

Since you are the penetration tester and already performed the exploitation stage, you already have all the elements to judge the impact level.

Note: the business impact cannot be determined without an asset valuation performed by your client. This is a common practice during risk assessments. You should ask your client to give you access to their most recent risk assessment documentation. If this is not available it is up to you to ask your client to determine what’s vital for their business. Also make sure to ask the same question to the technical departments. You will be amazed at how different answers you get asking the same question to different layers of the organizational chart.

2.2.1.3. Risk exposure over time

The importance of working with reliable metrics lies in the fact that we can reliably measure the level of security over time and act accordingly.

This is critical to your client and their executive level. They are interested into knowing how effective the security expenditure is and what is required to lower the risk to an acceptable level.

This is not the place to discuss the theory of risk assessment or ROSI (return on security investment), however you should understand what the management level is expecting: a mathematical proof that by hiring you and following your remediation guidelines, their risk is lowered.

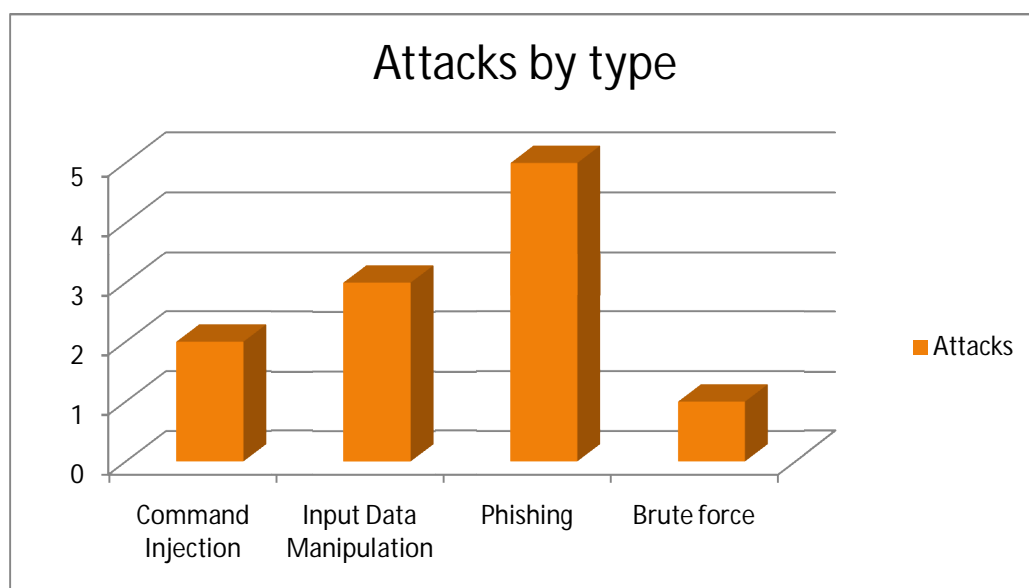
If yours is an intrusion test to check that the remediation plan of a previous report has been carried out properly, you will want to include a graph with the level of risk over time.

You will use your past reports to produce an estimate of the current risk compared to previous risk level.

This is a critical graph. We advise you to include it only if you are familiar with the risk assessment principles and terminology.

2.2.1.4. Successful attacks by type

The executive level may be interested into understanding what kind of attacks you have been able to carry out against each asset of the organization. A good classification of vulnerabilities and attacks is the MITRE [Common Attack Pattern Enumeration and Classification](#) (CAPEC). It offers a comprehensive dictionary on attacks along with useful information (description, vocabulary terms, reference links) that you can use in your report.



You can create a similar graph charting Vulnerabilities by type. You can refer to [WASC Threat classification](#) or [MITRE CWE](#) for a good list of vulnerabilities and their common names.

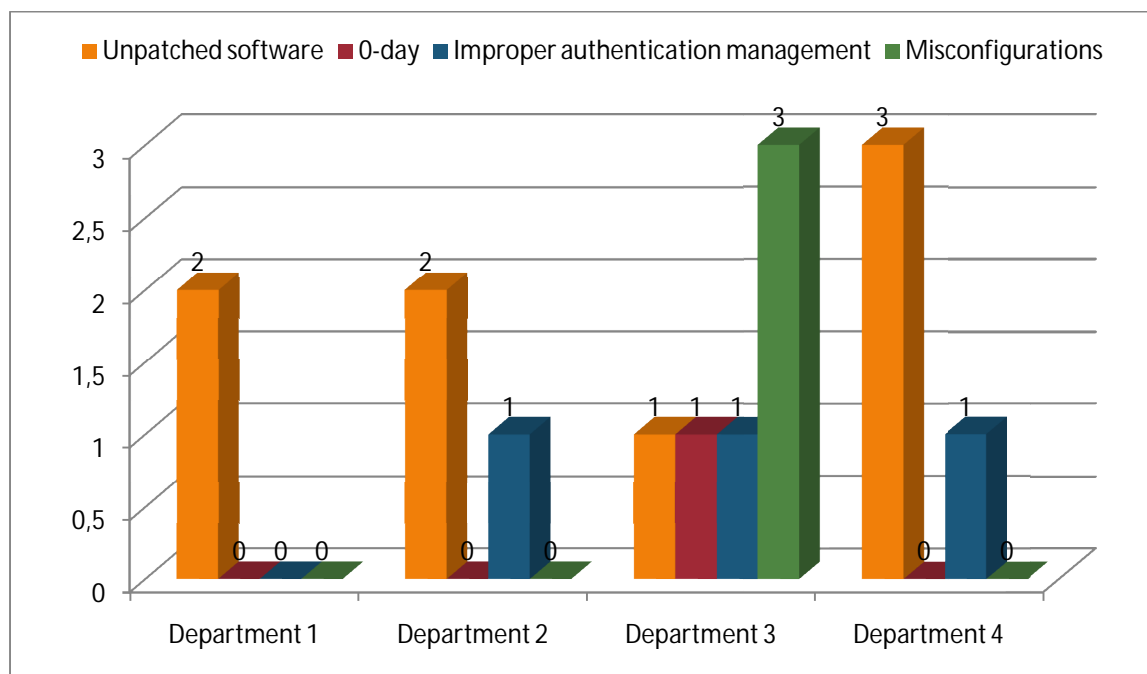
2.2.1.5. Vulnerabilities by cause

You can exploit a system because of:

- 0-day vulnerabilities
- Unpatched software
- Failure in the implementation of security controls
- Improper authentication management
- Misconfigurations
- Other

All the above are means that a threat agent (a hacker, an insider, the competition...) can use to violate your client security.

Charting vulnerabilities by their cause can let the executives identify responsibilities and take actions.



From the graph it is quite clear that a patch management software is required and that someone at Department 3 will be fired.

At the end of the charting part it is important to add the amount of work required to fix all the issues you have found. Executives and manager think about this work in terms of man-hours. If you can estimate this effort, this is a plus. However you are usually not aware of the inner processes or the difficulties that a fix can bring in an environment that you don't know. If you are performing your penetration testing internally (for your own company) you may add an estimate of the man hours. Otherwise you will just provide an overview of the required operations such as:

- Perform input data sanitization
- Use stronger ciphers
- Patch software X
- And so on...

2.2.2. Vulnerability report

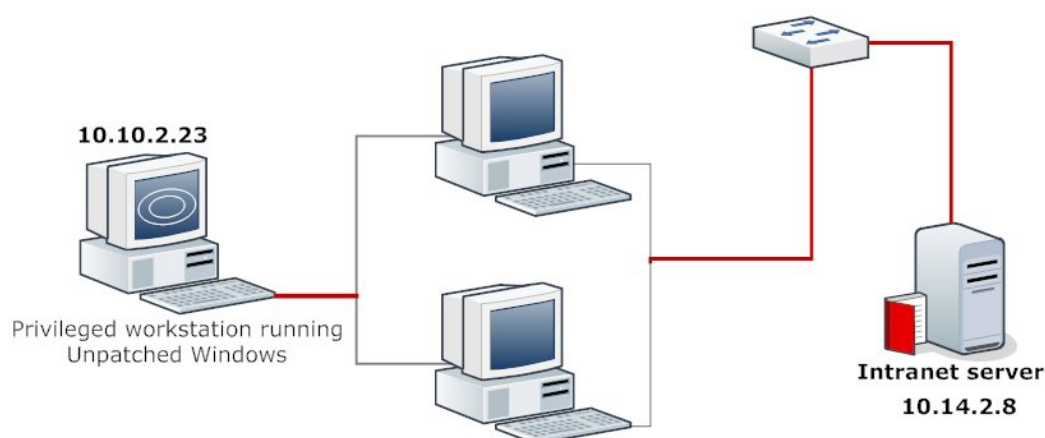
The vulnerability report deals with the actual findings of your penetration test.

This part will be read by technicians and sometimes even managers and it is used to understand in detail what is wrong about the organization security.

You will be able to talk in terms of specific vulnerabilities, exploitations, affected hosts (or domains or whatever in scope), attack vectors and so on.

To introduce the section you can still use graphs as long as they are relevant to the recipients of the documents. If your penetration test has involved more stages or steps for the exploitation you will want to expose these steps graphically.

For example, if you managed to steal intellectual property from the intranet server of the organization through the privileged login credentials of an employee's desktop, stolen exploiting an un-patched desktop machine, you can easily picture it:



The bottom line is: you can show anything you want graphically, but do not forget that the vulnerability report is used by the organization to understand the vulnerabilities that allowed you to exploit their systems (in the above example: the un-patched desktop). Do not over-do with graphics in this part. Pick the most complex 3 exploitations of the project and picture them.

You have different options as to how to arrange the information within this section.

Most of the times you will deal with the same vulnerability being repeated on many hosts or on many pages of a website. In this case you will report vulnerabilities by their type.

2.2.2.1. Vulnerabilities by type

This arrangement of the information lets you concentrate more on the vulnerability and less on the target IP/Server/Website affected.

For each vulnerability you have found (and successfully exploited) you will use this schema:

Name of the vulnerability

- Brief description
- Impact (CVSSv2) - Business impact factored in
- References to classifications (WASC, MITRE CWE, OWASP)
- Vulnerability ID (OSVDB, Bugtraq ID, CVE)

Exploitation proof of concept

- Screenshots
- Exploitation code

Affected targets

- IP addresses : port
- Website pages
- Applications

The *name* of the vulnerability should be one of the [WASC Threat Classification](#), [MITRE CWE](#) or, for applications-specific vulnerabilities, [OWASP Top 10](#).

Make sure to adhere to one of the above according to the scope and stick with it.

For example, WASC offers a great threat classification for web applications, so you can use it for your web application testing engagements.

The *description* of the vulnerability can be taken by the above sources as well or, if the vulnerability is in a common off-the-shelf software, you will include its description from NIST or [OSVDB](#).

You should integrate the above with further explanation when the description sounds meaningless or too generic. Make sure that your description is always relevant to your client environment and that provides more information pertaining to the specific situation you encountered during tests.

Beside the name of vulnerability you should assign an impact according to:

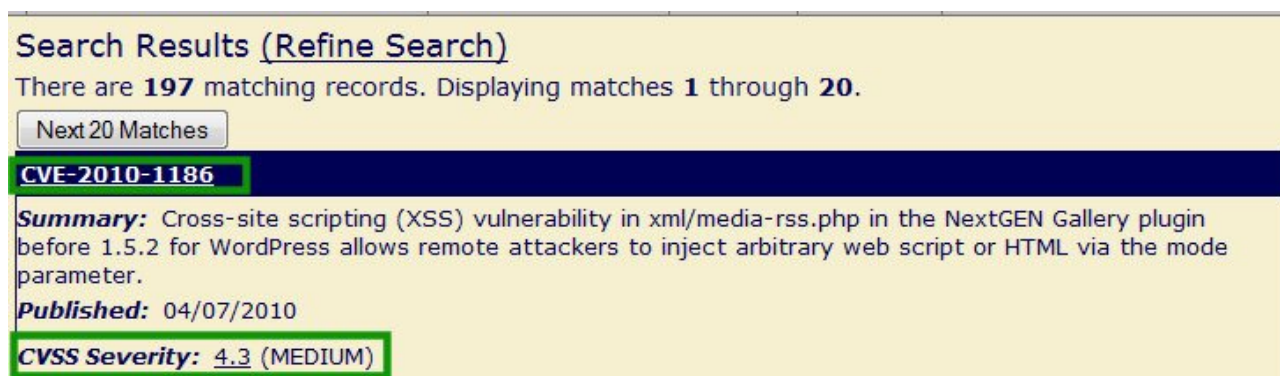
- Difficulty of the exploitation
- Affected systems (according to their asset value)
- Exposure (is it a remote vulnerability? Local? Does it require a privileged account?...)
- Availability (is there a public exploit? A metasploit module?)

The OWASP Top 10 – 2010 already assigns an impact to every vulnerability. Remember to make this impact meaningful for your client by adjusting this information with the value that the client poses into the affected systems.

The *vulnerability ID* is important to your client organization to gather more information or for statistical scopes. You have a vulnerability ID for publicly known vulnerabilities into software, devices and OS's.

The Mitre CVE-ID's are "unique, common identifiers for publicly known information security vulnerabilities". This means that, for publicly known vulnerabilities you will have to include the corresponding ID from at least Mitre CVE and OSVDB including a link to their page.

You can search for the ID corresponding to a vulnerability in a given software on [NIST pages](#) :



The screenshot shows a search results page from NIST. At the top, it says "Search Results (Refine Search)" and "There are 197 matching records. Displaying matches 1 through 20." Below this is a button labeled "Next 20 Matches". The first result is highlighted with a green border and contains the following information:

- CVE-2010-1186**
- Summary:** Cross-site scripting (XSS) vulnerability in xml/media-rss.php in the NextGEN Gallery plugin before 1.5.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the mode parameter.
- Published:** 04/07/2010
- CVSS Severity:** 4.3 (MEDIUM)

The above image shows the first result of a search for keyword "Wordpress", a common PHP application for web blogging. You can note the CVE ID (CVE-2010-1186), a summary that you can use in your report, and the CVSS Severity.

CVSS is a score assigned by NIST to every vulnerability according to its type, exposure and technical impact. You will want to take this score into consideration when assigning your impact.

The same search on OSVDB yields this result:

ID	Disc Date	Title
63574	2010-04-06	NextGEN Gallery Plugin for WordPress wp-content/plugins/nextgen-gallery/xml/media-rss.php mode Parameter XSS

NextGEN Gallery Plugin for WordPress contains a flaw that allows a remote cross site scripting (XSS) attack. This flaw exists because the application does not validate the 'mode' parameter upon submission to the 'wp-content/plugins/nextgen-gallery/xml/media-rss.php' script. This may allow a user to create a specially crafted URL that would execute arbitrary script code in a user's browser within the trust relationship between their browser and the server.

You have an OSVDB-ID and a description.

By clicking on the ID above (63574), you will find out the OSVDB has done most of the work for you:

Solution	Upgrade to version 1.5.2 or higher, as it has been reported to fix this vulnerability. An upgrade is required as there are no known workarounds.						
Products	Unknown or Incomplete						
References	<ul style="list-style-type: none">• CVE ID: 2010-1186 (see also: NVD)• Bugtraq ID: 39250• Secunia Advisory ID: 39341• Vendor Specific News/Changelog Entry: http://wordpress.org/extend/plugins/nextgen-gallery/changelog/• Other Advisory URL: http://www.coresecurity.com/content/nextgen-gallery-xss-vulnerability						
Credit	Unknown or Incomplete						
CVSSv2 Score	<div>CVSSv2 Base Score = 4.3</div> <div>Source: nvd.nist.gov Generated: 2010-04-08 Disagree?</div> <table><tr><td><div>Access Vector</div><div>Local</div><div>Adjacent Network</div><div>Remote</div><div>1.0</div></td><td><div>Access Complexity</div><div>Low</div><div>Medium</div><div>High</div><div>0.61</div></td><td><div>Authentication</div><div>Multiple Instances</div><div>Single Instance</div><div>None</div><div>0.704</div></td><td><div>Confidentiality</div><div>None</div><div>Partial</div><div>Complete</div><div>0.0</div></td><td><div>Integrity</div><div>None</div><div>Partial</div><div>Complete</div><div>0.275</div></td><td><div>Availability</div><div>None</div><div>Partial</div><div>Complete</div><div>0.0</div></td></tr></table>	<div>Access Vector</div> <div>Local</div> <div>Adjacent Network</div> <div>Remote</div> <div>1.0</div>	<div>Access Complexity</div> <div>Low</div> <div>Medium</div> <div>High</div> <div>0.61</div>	<div>Authentication</div> <div>Multiple Instances</div> <div>Single Instance</div> <div>None</div> <div>0.704</div>	<div>Confidentiality</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.0</div>	<div>Integrity</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.275</div>	<div>Availability</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.0</div>
<div>Access Vector</div> <div>Local</div> <div>Adjacent Network</div> <div>Remote</div> <div>1.0</div>	<div>Access Complexity</div> <div>Low</div> <div>Medium</div> <div>High</div> <div>0.61</div>	<div>Authentication</div> <div>Multiple Instances</div> <div>Single Instance</div> <div>None</div> <div>0.704</div>	<div>Confidentiality</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.0</div>	<div>Integrity</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.275</div>	<div>Availability</div> <div>None</div> <div>Partial</div> <div>Complete</div> <div>0.0</div>		

You have a suggested solution provided by the author of the vulnerability advisory, the References to other vulnerabilities databases (including CVE and Bugtraq) and also vendor specific news that you may want to check to ensure that an available patch has been issued for this vulnerability (this will be important in the remediation section).

Finally the CVSSv2 score, found in the National Vulnerability Database, here is depicted graphically. You can also get all the factors that contribute to the score assigned to this vulnerability (4.3).

If you go to the CVSSv2 [calculator](#) you will find out that you can calculate the impact of a vulnerability factoring in many aspects:

- Attack complexity
- Level of authentication needed
- Impact metrics (what will this vulnerability affect if successfully exploited) in terms of Integrity, Confidentiality and Availability
- Temporal metrics

This is at now the best way to score the impact of a vulnerability (especially if it is not a publicly known vulnerability but a vulnerability that you have found in ad-hoc client's code) because it also takes into account the business impact for your client organization:

Update Scores Reset Scores View Equations		Environmental Score Metrics	
CVSS Base Score	Undefined	<div style="border: 2px solid green; padding: 5px;"> This section addresses metrics that describe the effect of a vulnerability within an organization's environment. These metrics must be calculated separately for each organization. </div>	
Impact Subscore	Undefined		
Exploitability Subscore	Undefined		
CVSS Temporal Score	Undefined		
CVSS Environmental Score	Undefined		
Overall CVSS Score	Undefined		
Base Score Metrics		General Modifiers	
These metrics describe inherent characteristics of the vulnerability. All of these metrics must be filled in to perform any CVSS scoring.		<div style="border: 2px solid green; padding: 2px;"> Organization specific potential for loss (CollateralDamagePotential) </div> Not Defined ▼	
		<div style="border: 2px solid green; padding: 2px;"> Percentage of vulnerable systems (TargetDistribution) </div> Not Defined ▼	
Exploitability Metrics		Impact Subscore Modifiers	
Related exploit range (AccessVector)	Undefined ▼	System confidentiality requirement (draft proposal) (ConfidentialityRequirement) Not Defined ▼	
Attack complexity (AccessComplexity)	Undefined ▼	System integrity requirement (draft proposal) (IntegrityRequirement) Not Defined ▼	
Level of authentication needed (Authentication)	Undefined ▼	System availability requirement (draft proposal) (AvailabilityRequirement) Not Defined ▼	

By giving qualitative judgments for the different aspects of the vulnerability, the CVSSv2 formula will get you a numeric value of its impact.

You can use this numeric value as part of the metrics for measuring the organization security. Remember that what is measurable can be improved.

2.2.2.2. *Exploitation proof*

This section should not be too long. It is intended for your audience to understand how the exploitation was carried out and, more specifically, to the development team to reproduce it.

You will include:

- Snapshots
- Exploit payloads

Exploit payloads can include HTTP request and response headers in the case of web application testing and only if these are relevant in the exploitation. You basically want to keep redundant information at a minimum to improve the readability of the report.

Most of the times you will need to explain the exploitation further if it is not straightforward. Remember that:

1. Including the 100's SQL queries that gave you the password of a user in the organization's database is not much relevant. You would just prove the existence of a SQL injection vulnerability through a very simple proof of concept exploit injection query
2. Including the detailed exploitation, including payload, of a remote buffer-overflow in an ad-hoc application used by the organization is very relevant and actually required to reproduce the vulnerability.

2.2.2.3. *Affected systems*

The systems affected by the same vulnerability can be listed in a raw table or can be categorized by department, function, responsibility area and so on.

Department	Hosts
Marketing	10.10.2.3
	10.10.2.4
	10.10.2.78
	10.10.2.29
Production	10.2.3.12
R&D	10.4.3.1
	10.4.3.98

The above table groups IP addresses sharing the same vulnerability by Department.

You can use a variant of the above to list targets affected by the same vulnerability.

2.2.2.4. Vulnerabilities by target

If vulnerabilities are heterogeneous, or the number of targets in scope is little, you can arrange the above information on a per target basis.

Target (IP/domain/devices...)

- General information about the target
- Graph with the vulnerabilities found by type or impact

Vulnerability 1

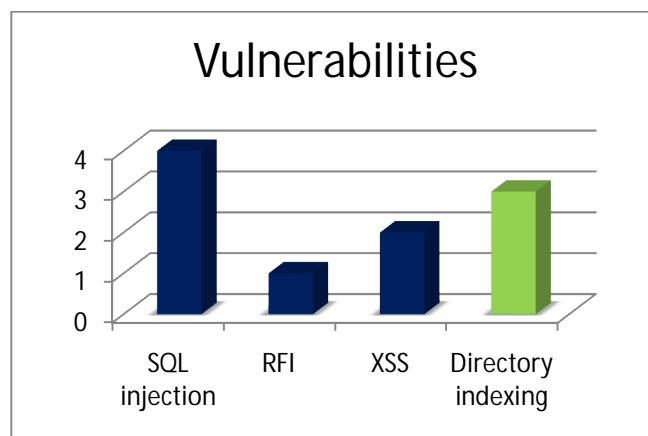
- Brief description
- Impact (CVSSv2) - Business impact factored in
- References to classifications (WASC, MITRE CWE, OWASP)
- Vulnerability ID (OSVDB, Bugtraq ID, CVE)

Vulnerability 2

- Brief description
- Impact (CVSSv2) - Business impact factored in
- References to classifications (WASC, MITRE CWE, OWASP)
- Vulnerability ID (OSVDB, Bugtraq ID, CVE)

The above information is the same as the previous sample however we can give more importance to the target here.

You can include a couple of graphs to highlight the types of vulnerabilities found in the single target. For example, if the target is a web application and the target is a domain, you can show this graph:



You can use different colors according to the impact level. (Make sure that you use the same colors for each impact level throughout the report).

When the scope of the test is a web application with many URL's, you will want to pick the *Vulnerability by type* model (par. 2.2.2.3) in which you will have a section for each vulnerability and then list all the URL's that are affected by that particular vulnerability:

SQL Injection		
SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker is able to change the logic of SQL statements executed against the database. [...]		
Vulnerable urls		
Url	Parameter	Method
/faq.php	id	GET
/downloads/get.php	url	GET
/members/register.php	username, country	POST
...

Consider the above an incomplete skeleton of the scheme you can use when your scope includes a web application.

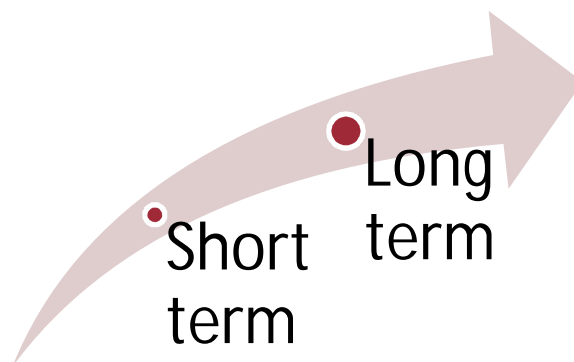
2.2.3. Remediation report

The remediation report should be the last part to be done in the reporting phase: by the time you have compiled the vulnerability report, you will already have all the information required to produce a prioritized action plan for your client organization.

This is the section where you have to give your own advices on how to better solve the security issues that had let you violate the organization's security in the penetration testing phase.

The remediation plan can be left to a different professional in your team. However, if you work as a freelance you better master this part.

In this part you can work on two different horizons:



In the short term you want the remediation team to address the most important vulnerabilities as soon as possible.

You should suggest your client to provide you with an emergency phone number where you can immediately call the development team of the organization should you come across vulnerabilities that pose the organization's vital assets at risk (Very short term): sometimes it takes weeks if not months from the beginning of the tests to the release of the deliverables.

You can also suggest long-term actions. Examples are implementation of SDLC, the employment of security checks early in business processes or the use of different platforms or frameworks.

Long term actions will bring benefits on the long run and is something the organization will not be able to do in the next 6-12 months and not without a good investment of time and money.

Your job is to increase the security (and knocking the risk) of your client organization, not to merely exploit their machines.

2.2.3.1. *The remediation*

Providing suggestions on how to remediate common vulnerabilities is most of the times trivial.

If the vulnerability exploited was in a publicly available application (OS, COTS applications, databases...) you will just add references to available patches, upgrades, hotfixes or workarounds. You usually find all you need within the vulnerabilities databases that we have covered in the previous sections or within security advisories.

Other times you will have to suggest patches to the code or solutions according to the type of vulnerability: an input validation vulnerability can be solved by properly handling input data.

The arrangement of the information in this section should reflect the one used in the vulnerability report. If you have listed your vulnerabilities by type you will do the same here.

You will have to prioritize your remediation plan according to the impact level you have given in the vulnerability report. Make sure that the first issues to fix are the most important.

Double check and adjust your priority according to your common sense and experience as well.

An example of a remediation item is the following schema:

#1 Microsoft IE mshtml.dll Use-after-free code execution (MS10-002)

- CVSS = 9.3
- CVE-2010-0249 - OSVDB ID: 61697
- Vector: Remote

Action: Apply patch 978207

- Vendor patch:
<http://www.microsoft.com/technet/security/Bulletin/MS10-002.mspx>
- Estimated time: 30 minutes
- Pre-requisites: none

Actionable targets

- 10.2.3.5
- 10.2.971

Note: you wouldn't use the above style in your report. You would provide many more details and sorted out in a more clear way. Consider it sample schema!

The item has the name of the vulnerability as you can see.

You will provide a list of targets where the patch needs to be applied and an estimated effort for the fix of each single actionable item. You can also include whose participation is required to apply the suggested solution: developers or system administrators (if it is a system-wide patch that will affect the functioning of other components).

It is advisable that you include a good amount of details when you are trying to have a 0-day vulnerability fixed. You need to provide what is required to reproduce the exploitation. Moreover you have to provide a potential solution on top of your understanding of the inner functioning of the remote target.

Sometimes this can be a challenge since penetration testing is usually a black box security audit. This is when the remediation team would call you back for further explanation, if the solution you provided is too generic.

In such a case the single remediation item would look like this:

#4 SQL Injection

- Description of the vulnerability
- Who: Developer
- CVSS = 8.1
- Vector: Remote
- Type: Error Based
- Proof of concept: `/faq.php?id=0 or db_name(0)=0;--`

Action: Employ input validation

- Short term:
Convert input to integer using functions `is_integer()` and `int()`
- Long term:
Employ prepared statements

Actionable targets

- `www.targetscope.com/faq.php`

2.2.4. Logs

The logs is the history table keeping track of the following information

- Date of the test
- Target tested
- Type of tests performed
- Source IP address (your IP address)
- Tools used
- Member of the penetration testing team

This log may be included within the report if required by your rule of engagement.

Most of the times this is not required but you want to still save it for yourself as a proof of what happened any time during your project. It will turn out very useful if a new test will be scheduled much time later.

3. Conclusions

This reporting guide has the goal of introducing the reader to the main aspects of producing a penetration testing report that delves into the business needs of the client and provides real solutions for real problems.

You, as a penetration tester, should not underestimate the importance of presenting your work: the sad truth is that you will not be judged by the difficulty of your exploitations but by how well you understand what that exploitations means for your clients in terms of monetary loss, business damage and loss of customer confidence into your client's brand.

Armando Romeo



Training Areas

- Web Application Security
- Source Code Audit
- Security Awareness
- Application Security Testing
- Security management
- Security Policy Implementation
- SDLC Development
- Threat Modeling

eLearnSecurity

Information Security Training Solutions

eLearnSecurity is a global Information Technology Security Training Solutions provider, catering to Government bodies, Educational Institutions and IT Security Professionals around the world.

E-mail: contactus@elearnsecurity.com

Website: <http://www.elearnsecurity.com>

Address:

Head Office
Pisa, Italy