PENETRATION TESTING REPORT

FOR

{CLIENT NAME}

FROM

NETWORK INTELLIGENCE
Global cybersecurity provider

CONTENTS

Document Details

DOCUMENT VERSION CONTROL	
Document Title	
Document Id	Company Name/Client Name/Month Year
Version #	
Last Edit Date	15-Feb-19

DOCUMENT SUBMISSION DETAILS	
Date	DD-MM-YYYY
Classification	Client Confidential
Submitted To	
Designation	
Address	
Contact Number	
E-Mail	

DOCUMENT DISTRIBUTION LIST

Sr. No.	**Name**	**Organization**	**Responsibility**
01.		Network Intelligence	Document Preparation
02		Network Intelligence	Document Review
03.		Network Intelligence	Quality Check
04.		Network Intelligence	Document Approval
05.		Client's Name	Document Appraisal and Acceptance

NOTICE

COPYRIGHT

TRADEMARKS

NII CONTACT DETAILS

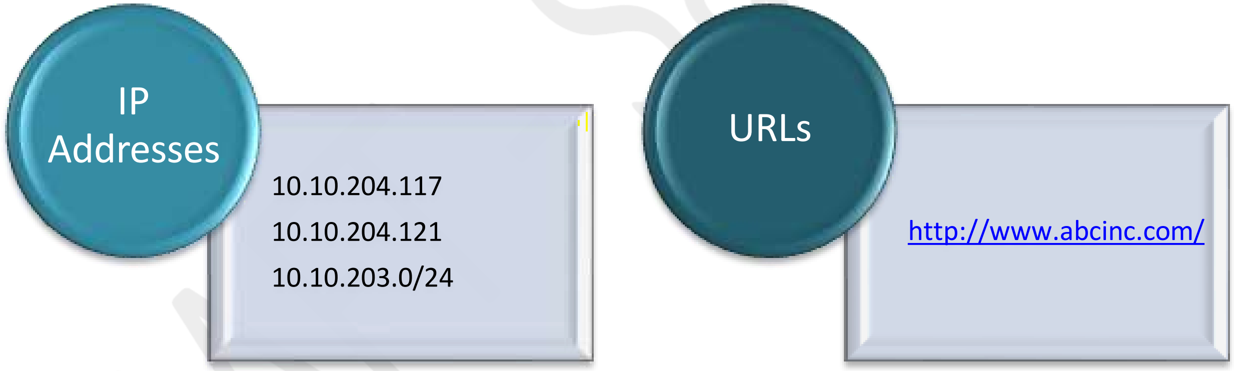
Name	-
Title	-
Company	Network Intelligence Inc.
Address	New York | Dubai | Mumbai | Pune | New Delhi | Bengaluru | Hyderabad
Tel. No	
Mobile No	
E – Mail	info@niiconsulting.com

1 Executive Summary

1. Overview

ABC Inc. had assigned Network Intelligence Inc. the task of carrying out a penetration test of their public facing infrastructure as specified below in the scope of work section. The engagement was conducted in compliance with regulatory requirements to conduct bi-annual vulnerability assessment and penetration tests of the Internet infrastructure of the organization.

1. Scope of Work



10.10.204.117
10.10.204.121
10.10.203.0/24

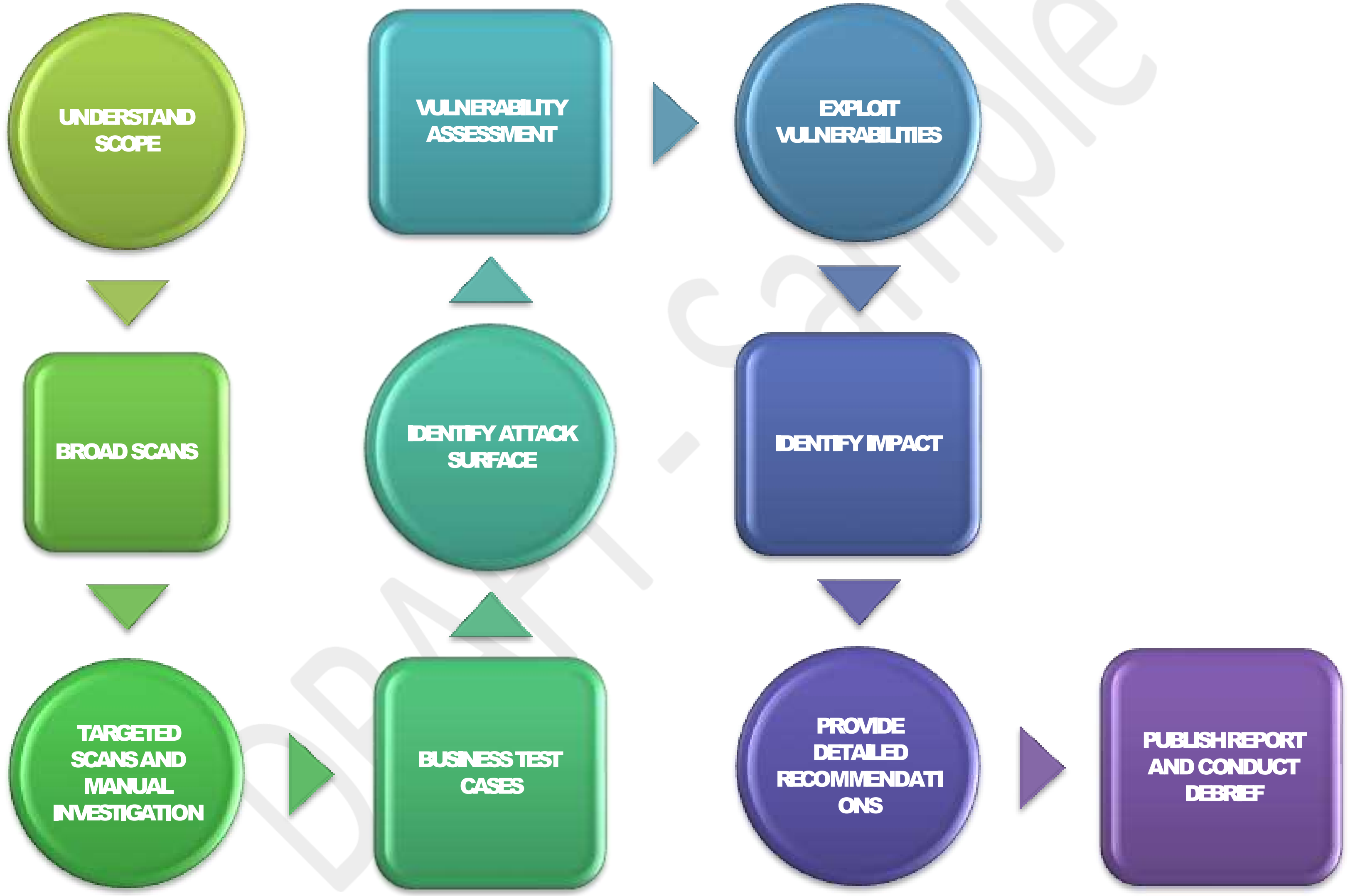
http://www.abcinc.com/

1.1.2 Testing Overview

OBJECTIVE	• Discover vulnerabilities • Exploit non-destructively • Achieve regulatory compliance
DURATION	• 7th October to 30th October.
TESTING TYPE	• Grey Box Testing • Business Risk-Based Approach

**** Refer to Section 5 Appendix B for definitions and details of different types of assessments***

1.2 APPROACH


UNDERSTAND SCOPE
BROAD SCANS
TARGETED SCANS AND MANUAL INVESTIGATION
BUSINESS TEST CASES
IDENTIFY ATTACK SURFACE
VULNERABILITY ASSESSMENT
EXPLOIT VULNERABILITIES
IDENTIFY IMPACT
PROVIDE DETAILED RECOMMENDATIONS
PUBLISH REPORT AND CONDUCT DEBRIEF

1.3 Standards and Framework Followed

1. Open Web Application Security Testing Framework (OWASP)
2. Web Application Security Consortium (WASC)
3. The Open Source Security Testing Methodology Manual (OSSTMM)
4. National Institute of Standards and Technology (NIST)

1.4 Severity Rating

Network Intelligence follows the Common Vulnerability Scoring System (CVSS) scoring system to rate vulnerabilities.

The CVSS assessment measures three areas of concern:

- **Base Metrics** for qualities intrinsic to a vulnerability
- **Temporal Metrics** for characteristics that evolve over the lifetime of vulnerability
- **Environmental Metrics** for vulnerabilities that depend on an implementation or environment

For more information on what the Base, Temporal, and Environment Metrics in the CVSS scoring system are, please visit https://www.first.org/cvss/specification-document

Unless otherwise stated, the findings in the report are scored on the base-metric rating of the vulnerability.

NII may consider Environmental and Temporal Metrics depending on information provided at project initiation and if this is a mandatory client reporting requirement.

Based on the severity of the vulnerability, they are assigned below ratings:

CVSS Qualitative severity rating scale *(For detailed definitions and details of severity ratings, please refer to* appendix C *in this report)*

CVSS Severity Rating	CVSS Score
Critical	9.0 - 10.0
High	7.0 - 8.9
Medium	4.0 - 6.9
Low	0.1 - 3.9
None	0.0

5. Summary Of Findings

1. Vulnerabilities by Severity

Below is the table that summarizes the list of findings discovered during the project:

Sr. No.	Title	Severity Rating	CVSS Rating	Key Remediation
1	SQL Injection	CRITICAL	9.2	• Use parameterized queries. • Use stored procedures • Use input validation (white-listing)
2	Insufficient Authorization	HIGH	8.5	• Use role-based access control • Implement strong access control list
3	Default Installation of PHPMyAdmin Found	HIGH	8.1	• Change default admin user credentials • Restrict access to admin portal to specific user/s
4	CSV Injection	MEDIUM	6.0	• Perform input validation before saving data files
5	Directory Listing enabled	LOW	3.5	• Configure web server to prevent directory listing

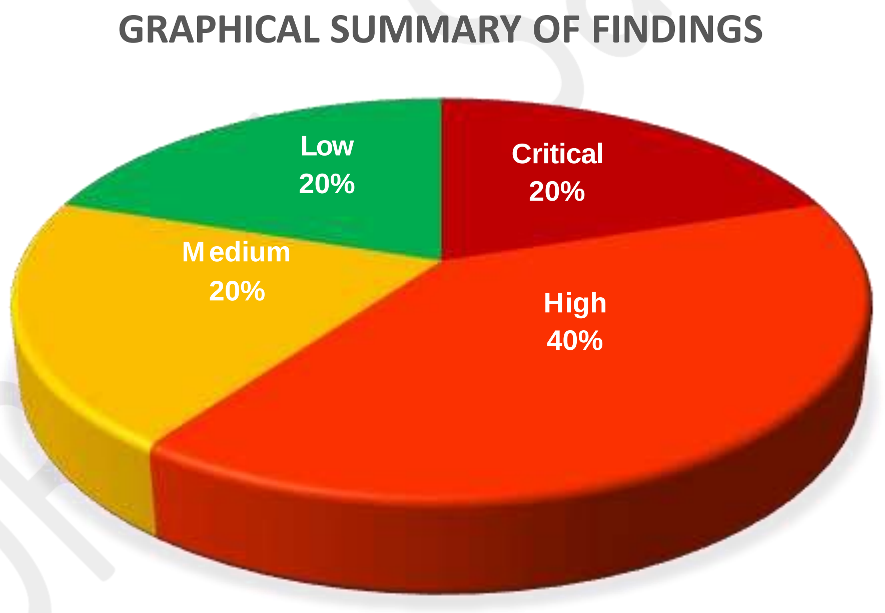
Vulnerability Summary

1.5.2 Tabular Summary

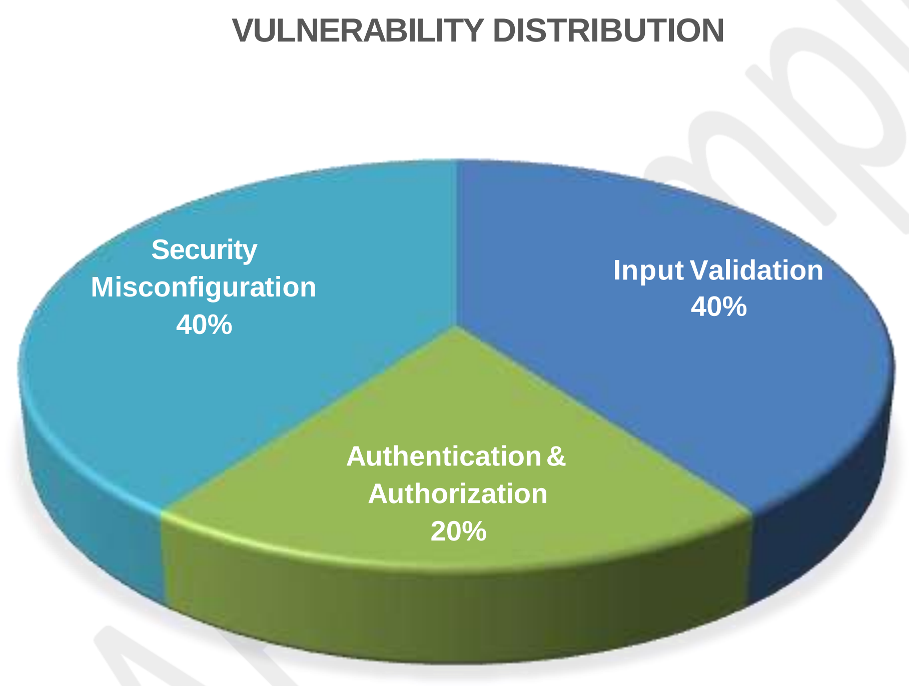
The following table summarizes the results of the assessment:

Category	Description			
	Systems Vulnerability Assessment Summary			
Number of Systems/IP Address and URLs	1+1			
Number of Vulnerabilities found	5			
Critical, High, Medium and Low Severity Vulnerabilities	1	2	1	1

1.5.3 Graphical Summary



1.5.4 Vulnerability Distribution – Areas to Focus



2 DETAILED TECHNICAL REPORT

1. WEB APPLICATION VULNERABILITY ASSESSMENT

1. SQL INJECTION

SEVERITY LEVEL (CVSS RATING)

CRITICAL (9.2)

VULNERABILITY CLASSIFICATION

SQL Injection

AFFECTED URL

http://www.abcinc.com/print
http://www.abcinc.com/forward

DESCRIPTION

SQL Injection is an attack technique used to exploit applications that construct SQL statements from user-supplied input. When successful, the attacker can change the logic of SQL statements executed against the database.

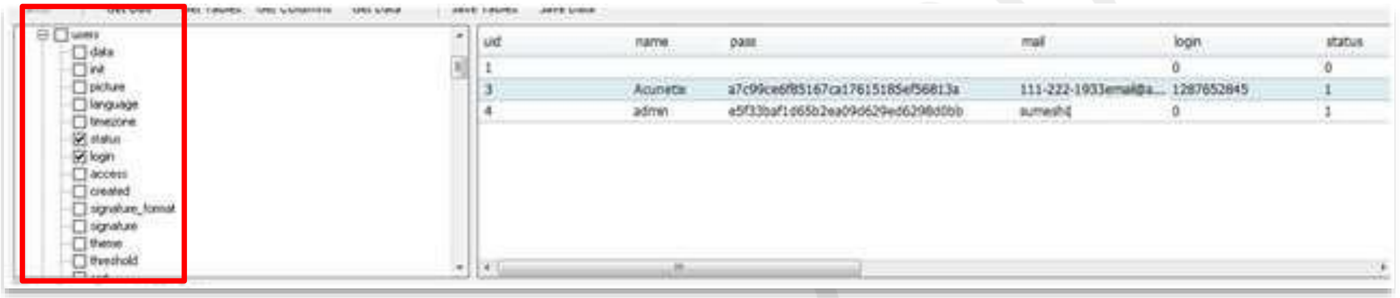
ANALYSIS

It was found that the web application does not validate properly user provided input at the above URLs.
Affected URL: http://www.abcinc.com/print
Vulnerable Parameters: pageID, nid
Method: GET
Malicious Request:
http://www.abcinc.com/print?pageID=37';&nid=1
Affected URL: http://www.abcinc.com/forward

Vulnerable Parameter: email, tomail
Method: POST
Malicious Parameter Value: email=test@test.com%00'

IMPACT

We could enumerate the databases schema of the web application. We could also successfully extract user details as shown in the figure below.



users
- data
- init
- picture
- language
- timezone
- [✓] status
- [✓] login
- access
- created
- signature_format
- signature
- theme
- threshold

uid	name	pass	mail	login	status
1				0	0
3	Acunetix	a7c99ce6f85167ca17615185ef56813a	111-222-1933email@a...	1287652845	1
4	admin	e5f33baf1d65b2ea09d629ed6298d0bb	sumesh[	0	1

FIGURE 1: DATABASE DETAILS ENUMERATED USING SQL INJECTION

RECOMMENDATION

Make use of stored procedures to abstract data access so that users do not directly access tables or views.
Following are the recommendations to prevent SQL injection attack.

1. Use of prepared statements (Parameterized Queries)

```
$name = $_GET['username'];
if ($stmt = $mysqli->prepare("SELECT password FROM tbl_users WHERE name=?")) {
  // Bind a variable to the parameter as a string.
   $stmt->bind_param("s", $name);
  // Execute the statement.
   $stmt->execute();
   // Get the variables from the query.
   $stmt->bind_result($pass);
  // Fetch the data.
   $stmt->fetch();
   // Close the prepared statement.
   $stmt->close();
}
```

2. Use of stored procedures
3. Escaping all user supplied input.
4. Enforce least privilege.
5. Also, perform white list input validation. Sanitize input to exclude context-changing symbols such as:
   - ' (single apostrophe)
   - " (quotation mark)
   - \' (backslash-escaped apostrophe)
   - \" (backslash-escaped quotation mark)
   -) (closing parenthesis)
   - ; (semicolon)

Reference

- CVSS Vector: NII - CVSS:3.0 Calculator
- OWASP - https://www.owasp.org/index.php/SQL_Injection
- OWASP SQL Injection Prevention Cheat Sheet - https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- SQL Injection - http://projects.webappsec.org/SQL-Injection
- Wikipedia - SQL Injection http://en.wikipedia.org/wiki/SQL_injection
- Stop SQL Injection Attacks Before They Stop You - http://msdn.microsoft.com/msdnmag/issues/04/09/SQLInjection/
- Understanding and Preventing SQL Injection Attacks - http://www.silksoft.co.za/data/sqlinjectionattack.htm
- Penetration Testing for Web Applications (Part Two) - http://www.securityfocus.com/infocus/1709

2.1.2 INSUFFICIENT AUTHORIZATION

SEVERITY LEVEL (CVSS RATING)

HIGH (8.5)

VULNERABILITY CLASSIFICATION

Insufficient Authorization

AFFECTED MODULE & URL

https://abcinc.com

Parameter: Investment ID

DESCRIPTION

Insufficient Authorization results when an application does not perform adequate authorization checks to ensure that the user is performing a function or accessing data in a manner consistent with the security policy. Authorization procedures should enforce what a user, service or application is permitted to do. When a user is authenticated to a web site, it does not necessarily mean that the user should have full access to all content and functionality.

ANALYSIS

During analysis, we found that "Account Statement" for other user accounts can be downloaded by any user. Following screenshots confirms the proof of concept.

- The screenshot below shows a drop-down list with Account IDs. Choose any one, intercept the request and copy the text highlighted in figure.

<SCREENSHOT REDACTED FOR SECURITY>

Figure 2: list of IDs

<SCREENSHOT REDACTED FOR SECURITY>

Figure 3: highlighted text copied

- Following screenshot shows Account ID of 2nd account and its details. Now, again choose any one to make a request to the server, intercept it and replace the highlighted text in screenshot with the one copied earlier.

<SCREENSHOT REDACTED FOR SECURITY>

Figure 4: List from 2nd user account

<SCREENSHOT REDACTED FOR SECURITY>

Figure 5: Text Replaced with Copied One

- Following screenshot shows details of 1st user account's Account statement.

<SCREENSHOT REDACTED FOR SECURITY>

Figure 6: Details of 1st user account displayed

IMPACT

An attacker can see the victim's account details; this could lead to compromise of a customer's confidential private data and regulatory non-compliance.

RECOMMENDATION

It is recommended to implement strong authorization in the application.

1. Divide the software into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) to enforce the roles at the appropriate boundaries.
2. Ensure that access control checks related to your business logic are performed. These checks may be different from the access control checks that are performed on more generic resources such as files, connections, processes, memory, and database records.
3. For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page.
4. Ensure that all pages containing sensitive information are not cached
5. Use the access control capabilities of your operating system and server environment and define your access control lists accordingly. Use a "default deny" policy when defining these ACLs.

REFERENCE

- CVSS Vector: NII - CVSS:3.0 Calculator
- Insufficient Authorization: http://projects.webappsec.org/w/page/13246940/Insufficient%20Authorization
- OWASP Testing for Privilege Escalation: https://www.owasp.org/index.php/Testing_for_Privilege_escalation_(OTG-AUTHZ-003)
- OWASP Access Control Cheat Sheet: https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

2.1.3 Default Installation of phpMyAdmin found

Severity Level

HIGH (8.1)

Vulnerability classification

Server Misconfiguration

Affected URL

http://www.abcinc.com/phpMyAdmin/

Description

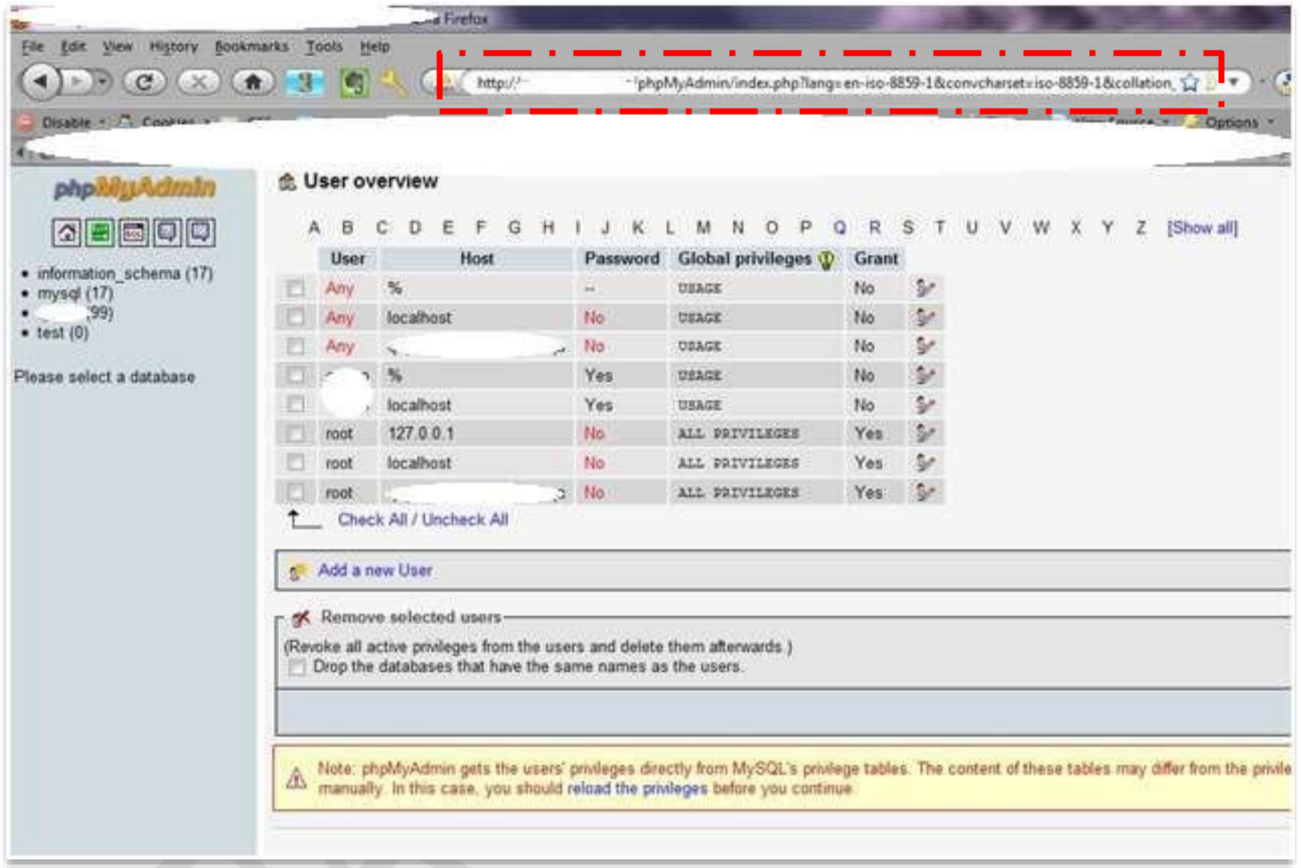
Server mis-configuration attacks exploit configuration weaknesses found in web servers and application servers. Many servers come with unnecessary default and sample files, including applications, configuration files, scripts, and web pages. They may also have unnecessary services enabled, such as content management and remote administration functionality. Servers may include well-known default accounts and passwords. Failure to fully lock down or harden the server may leave improperly set file and directory permissions.

Analysis

PHPMyAdmin is a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web. It was found that the PHPMyAdmin installation at the affected URL was not been configured securely. The credentials required to login are the same as the default.

The credentials we used to login are:
username: **root**
password: <blank>

username: **abcinc**
password: <blank>


http://[redacted]/phpMyAdmin/index.php?lang=en-iso-8859-1&convcharset=iso-8859-1&collation_

User overview

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [Show all]

User	Host	Password	Global privileges	Grant
Any	%	--	USAGE	No
Any	localhost	No	USAGE	No
Any	[redacted]	No	USAGE	No
[redacted]	%	Yes	USAGE	No
[redacted]	localhost	Yes	USAGE	No
root	127.0.0.1	No	ALL PRIVILEGES	Yes
root	localhost	No	ALL PRIVILEGES	Yes
root	[redacted]	No	ALL PRIVILEGES	Yes

Check All / Uncheck All

- information_schema (17)
- mysql (17)
- [redacted] (99)
- test (0)

Please select a database

Add a new User

Remove selected users

(Revoke all active privileges from the users and delete them afterwards.)

Drop the databases that have the same names as the users.

Note: phpMyAdmin gets the users' privileges directly from MySQL's privilege tables. The content of these tables may differ from the privile manually. In this case, you should reload the privileges before you continue.

FIGURE 7: LOGGED IN TO THE 'PHPMYADMIN' INTERFACE USING CREDENTIALS ROOT:<BLANK>

IMPACT

Once we had access to the phpMyAdmin interface, we could potentially execute SQL queries and manipulate the values in the database.

We were also able to read the /etc/passwd file of the Apache webserver. This indicates that the webserver could be running with the privileges equivalent to that of the root.

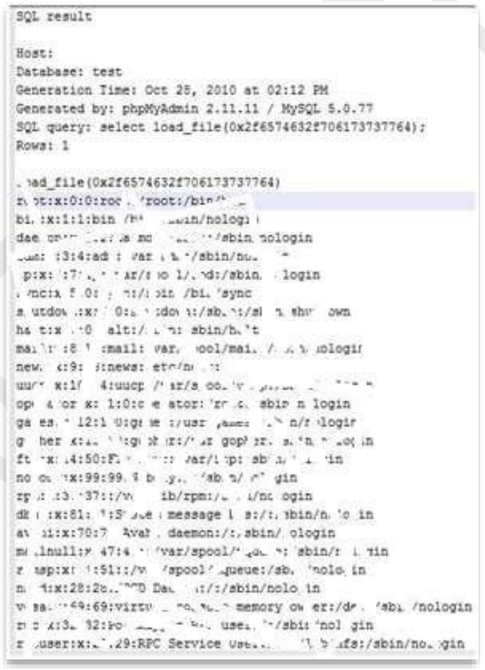
The query executed using phpMyAdmin was:

```
SELECT load_file(0x2F6574632F706173737764);
```

0x2F6574632F706173737764 is the equivalent form of **/etc/passwd** when encoded in **HEX-encoding**

```
[illegible]
```

FIGURE 8: LOCAL SYSTEM FILE '/ETC/PASSWD' EXTRACTED USING 'PHPMYADMIN' INTERFACE



```
SQL result

Host:
Database: test
Generation Time: Oct 25, 2010 at 02:12 PM
Generated by: phpMyAdmin 2.11.11 / MySQL 5.0.77
SQL query: select load_file(0x2f6574632f706173737764);
Rows: 1

load_file(0x2f6574632f706173737764)
r ot:x:0:0:roc  'root:/bin
bi :x:1:1:bin /b   in/nologi
dae on  [illegible]  /sbin nologin
 :3:4:ad  var  /sbin/no
 p:x: :7  ar/  1/ d:/sbin   login
  nc  5 0:  :/ in /bi 'sync
s utdo  :x  0:  do  :/sb  :/s   shu  own
h  t:x  :0  alt:/  n: sbin/h 't
mai  :8  :mail: var  ool/mai  /  nologi
new  x:9:  :news: etc/n
uuc  x:1   4:uucp /  ar/s oo  [illegible]
op   or x: 1:0:o e ator: 'r    sbin n login
ga es  12:1 0:g  e  /usr  ame     n/  logi
g  her x:1   :g  :r:/  ar gop er  s  n  o  in
ft  x: 14:50:F     :  var/  p: sb       in
no o   x:99:99  b  y   /sb n/ o  gin
rp   :3  37::/v    ib/rpm:/   /no ogin
dk   :x:81:  :S   e  message b  s:/  sbin/n   o in
av  i:x:70:7  Ava   daemon:/  sbin/ ologin
m  lnull:x 47:4   /var/spool/  qu  : sbin/   in
r  sp:x:  :51::/v  /spool/ queue:/sb   nolo in
n  d:x:28:28  D Da    :/:/sbin/nolo in
v sa  69:69:virtu    memory ow er:/de   /sbi /nologin
r  x:3  32:Po    user  /sbi /nol gin
r  user:x:  29:RPC Service Use      fs:/sbin/no  gin
```

RECOMMENDATION

It is recommended to change the default username and accounts on the PHPMyAdmin and MySQL applications.

Also, the Apache server should be configured in a chroot environment to prevent attackers from accessing the local system in the event of a compromise.

Detailed information on setting up a CHROOT environment can be found in the links mentioned in the references section.

All accounts should have complex passwords defined for them:

- Should be atleast 8 characters
- Should be alphanumeric (i.e. should contain letters and numbers)
- Should have atleast one special character (!,@,#,$,%,&,*)
- Should not contain user, vendor, organization or service names
- Should have one or more uppercase characters

REFERENCE

- CVSS Vector: NII - CVSS:3.0 Calculator
- Insufficient Authentication: http://projects.webappsec.org/Insufficient-Authentication
- Securing Apache Step by Step: http://www.symantec.com/connect/articles/securing-apache-step-step
- Linux.com – Apache Chrooting: http://www.linux.com/archive/feed/36331

2.1.4 CSV INJECTION

SEVERITY LEVEL

MEDIUM (6.0)

VULNERABILITY CLASSIFICATION

Improper Input Validation

AFFECTED URL

Affected Functionality:

a. Functionality 1 ➐N e w
b. Functionality 2 ➐View/Reply/Follow ➐Export toCSV

DESCRIPTION

Many web applications offer spreadsheet export functionality, allowing users to download data in a .csv file. The resulting spreadsheet's cells often contain user-supplied data which is risky, because any cells starting with the '=, +, -' character will be interpreted by the spreadsheet software as formulae. So, when a victim downloads the CSV file and opens it, the malicious code gets executed.

ANALYSIS

We observed that "Export to CSV" feature of the application was not sanitized by implementing "escape" technique for user input taken via the form fields. Because of this, the data entered into the form fields can became an active content.

<u>**Below are the steps to reproduce this vulnerability:**</u>

1. We logged in to the application, navigated to "Functionality 1 ➐N e w " and provided the payload ***-2+3=cmd|' /C calc'!A0*** on the fields as shown below.

<SCREENSHOT REDACTED FOR SECURITY>

FIGURE 9: SHOWS FIELD EDITED WITH PAYLOAD

2. We exported the data in CSV format as shown in below screenshot.

<SCREENSHOT REDACTED FOR SECURITY>

FIGURE 10: SHOWS EXPORT TO CSV

3. Payload was executed as we opened the exported document in MSEXCEL as shown in below screenshot.

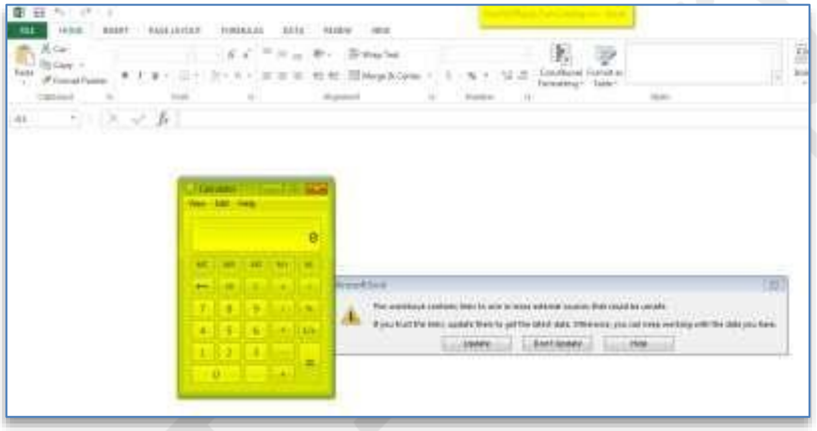


FIGURE 11: SHOWS PAYLOAD EXECUTED

IMPACT

Successful exploitation will allow an attacker to execute arbitrary code with the privilege of currently logged in user of the system. This may cause serious damage to victim's system such as wiping out an entire partition or backdoors creation for lateral access. Many other attacks are possible depending upon the creativity of the attacker.

RECOMMENDATION

It is recommended to append a single quote (') to the list of formula triggers (=, +, -) before saving it in the database. LibreOffice or Excel will ignore the single quote and just show the malicious formula as a string and it won't be interpreted as a formula.

REFERENCE

- CVSS Vector: [NII - CVSS:3.0 Calculator]
- CSV Injection Vulnerability: http://www.securityfocus.com/archive/1/537963

2.1.5 Directory Listing enabled

Severity Level

LOW (3.5)

Vulnerability classification

Directory Indexing / Security Misconfiguration

Affected URL

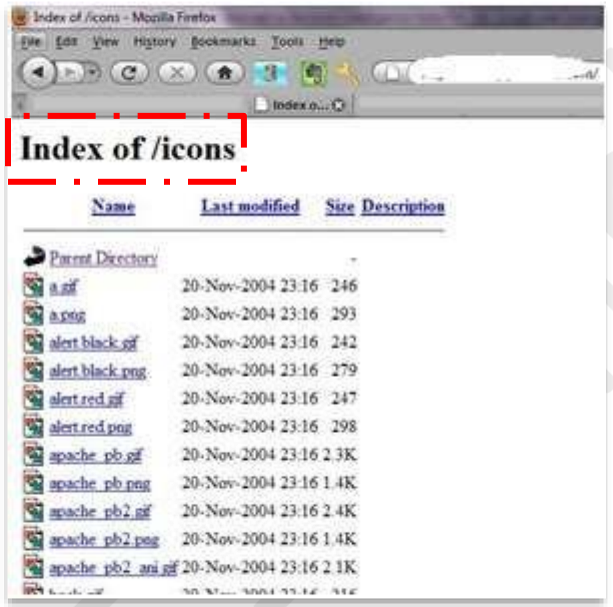
http://www.abcinc.com/icons/
http://www.abcinc.com/icons/small/

Description

Automatic directory listing/indexing is a web server function that lists all of the files within a requested directory if the normal base file (index.html/home.html/default.htm/default.asp/default.aspx/index.php) is not present. When a user requests the main page of a web site, they normally type in a URL such as: http://www.example.com/directory1/ - using the domain name and excluding a specific file. Essentially, this is equivalent to issuing an "ls" (UNIX) or "dir" (Windows) command within this directory and showing the results in HTML form.

Analysis

Directory listings do not necessarily constitute a security vulnerability. Any sensitive resources within your web root should be properly access-controlled in any case, and should not be accessible by an unauthorized party who happens to know the URL. We found that directory listing is enabled at the above URLs on the website.



Index of /icons - Mozilla Firefox

Index of /icons

```
Name                Last modified      Size Description
Parent Directory                          -
a.gif               20-Nov-2004 23:16   246
a.png               20-Nov-2004 23:16   293
alert.black.gif     20-Nov-2004 23:16   242
alert.black.png     20-Nov-2004 23:16   279
alert.red.gif       20-Nov-2004 23:16   247
alert.red.png       20-Nov-2004 23:16   298
apache_pb.gif       20-Nov-2004 23:16  2.3K
apache_pb.png       20-Nov-2004 23:16  1.4K
apache_pb2.gif      20-Nov-2004 23:16  2.4K
apache_pb2.png      20-Nov-2004 23:16  1.4K
apache_pb2_ani.gif  20-Nov-2004 23:16  2.1K
```

FIGURE 12: DIRECTORY LISTING ENABLED ON DIRECTORY /ICONS

IMPACT

Directory Listings can aid an attacker by enabling them to quickly identify the resources at a given path, and proceed directly to analyzing and attacking them.

RECOMMENDATION

Configure your web server to prevent directory listings for all paths beneath the web root;

REFERENCE

- CVSS Vector: NII - CVSS:3.0 Calculator
- Directory Indexing: http://projects.webappsec.org/Directory-Indexing
- Disable directory listing in Apache: http://www.htaccess-guide.com/disable-directory-listings/

3 CONCLUSION

Based on our testing of the target systems included in the scope, below are issues that need to be given due attention on priority basis

1. The default installation of PHPMyAdmin should be immediately password protected. Also, if the functionality is not in use, access to the same should be disabled. Alternatively, the access to PHPMyAdmin could be restricted only to a trusted host.

2. The webserver is configured to run with the privileges equivalent to the Unix superuser root. This poses a significant threat because if the attacker can exploit this vulnerability, they can take complete control of the server. It is highly recommended to create a chroot jail environment forApache.

3. The most critical issue in case of the web applications stands to be the SQL injection attack. It is highly recommended to sanitize any input coming from the client (user) using some validation function before the request is processed by the server.

4. The server should be adequately hardened and fully patched before it is put in production. Please follow CI Security benchmarks (www.cisecurity.org) to harden all components of the server (operating system, web server, database server).

4 APPENDIX A: OWASP TOP TEN 2013

Name	**Description**
A1-Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as a part of the command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2 – Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3 – Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4 – Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 – Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6 – Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7 – Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests to access functionality without proper authorization.
A8 - Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests that the vulnerable application thinks are legitimate requests from the victim.
A9 - Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10 – Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

5 Appendix B: Types of Assessments

Type of Penetration Testing Approach	**Description**
Red Team Assessments	A Red Team Assessment comes closest to simulating a real-world attack. Here, the scope of work is usually the entire organization, limited only by deciding whether the Red Team comes onsite also or does all their hacking remotely. The exercise is time bound and the results are very often eye-opening.
Bug Bounties	Bug bounties are now the norm, and we propose that you could run a private bug bounty program for a limited duration and a limited scope to understand the value you can get from it. In traditional types of security assessment (see the list below), you pay for consultant effort, and not just for the results. With bug bounties, you pay only for the results.
Black Box Approach	Here, we only know the URL of the website. Enumeration of technologies, mapping of the website, identification of fault injection points, determining input validation vulnerabilities, or logical security vulnerabilities, and the OWASP top 10 attacks are all part of this exercise.
Gray Box Approach	Often enough, a web application involves authentication and authorization of components. To be able to test these, we request for a dummy user account with the least level of privileges within the application. Using this account, we can log in and test for various flaws in the authentication scheme, as well as an attempt to escalate our privileges and bypass authorization restrictions.
War Dialing	Is a Penetration Test technique in which a list of telephone numbers are called to search for computers, systems, and fax machines. The purpose is to gain as much business-critical data as possible with the help of this technique.
Wireless Hacking	Wireless hacking is done to gather all loopholes possible in an organization's wireless infrastructure. This is done with an intention to gain unauthorized access and to try and exploit as much resources as available.
Social Engineering	Controls can be put on systems and devices but same does not hold true for theobjects using these systems (employees/temporaries). Social Engineering is the method by which all the hackers try and get the confidential and business critical information by using various techniques. This test focuses on exploiting and finding out all the possible loopholes pertaining to this domain so that your organization is geared up to face social engineering attacks in life.
War Driving	War driving can be carried to test the range and strength of your organization's wireless network's signal. This will help to gauge the extent of the threat exposure area for your organization.
Business Risk Based Approach	Traditional Penetration Testing approach only focuses on the technical vulnerabilities. But Business Risk based approach not only focuses on the technical vulnerabilities but also on the risks presumed to the organization's business. First, test cases pertaining to the business threat model are developed and Penetration test is carried out focusing majorly on these cases. This method has many advantages over the traditional Penetration Test methodology. And one of the biggest advantages it has is that of being business focused.
Source Code Review	Source code review focuses on detecting the vulnerabilities early in the Software Development Life Cycle (SDLC) such as Dataflow attacks, Cross Site Scripting (XSS), Injection (SQL, File, XPATH, reflection, etc.). File Inclusion/execution and Information Leakage and this methodology will help the organization to close the loopholes during the development and testing phase.

Internal Penetration Testing	Many organizations secure their organization from outside threats but leave their internal network security comparatively weak. And it has been proved repeatedly that the organization's security was compromised from within the network. Internal Penetration Test focuses on identifying these loopholes and recommending solutions to make the internal network secure to thwart internal threats and attacks.
Vulnerability Assessment	Vulnerability Assessment is the process of identifying, quantifying, and prioritizing the vulnerabilities of the components of IT infrastructure.
Stress Testing	As applications move to the Web, into a server based environment, it becomes increasingly important to be able to gauge the performance and load capability of an application. Stress testing involves testing the web-application's ability to handle the load/stress resulting from an increase in hits, which maybe a result of the sudden change affecting the organization's business activity directly.
Denial of Service Testing	A very serious and significant threat that web applications face is from DoS attacks. In this attack, the attacker sends so many packets to a website that it cannot service the legitimate users that are trying to access it. This leads to denial of service to the legitimate users, thus affecting the business directly. So, assessment needs to be carried out to check for the organization's preparedness to face such an attack.

6 APPENDIX C: SEVERITY RATING DETAILS

"The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to the threat. Scores are calculated based on a formula that depends on several metrics that approximate *ease of exploit* and the *impact of exploit*. Scores range from 0 to 10, with 10 being the most severe."

CRITICAL	CVSS Rating Scale: 9.0 - 10.0

Critical severity vulnerabilities usually have most of the following traits:

a. A successful attack may lead to complete compromise of the system
b. Exploitation can be done remotely over an untrusted connection – such as the Internet
c. Exploiting the vulnerability is very easy or straightforward. It may not require privilege accounts or user interaction.
d. It has a very significant impact on confidentiality, integrity and/or availability of the targeted system

Some examples of a Critical vulnerability are:

- Microsoft IIS6.0 Buffer Overflow Vulnerability(CVE-2017-7269) – **10.0**
- VMware Guest to Host Escape Vulnerability (CVE-2012-1516) - **9.9**
- GNU Bourne-Again Shell (Bash) 'Shellshock' Vulnerability (CVE-2014-6271) - **9.8**
- Golden Ticket (CVE-2014-6324) – **9.0**

Issues classified as 'Critical' need to be resolved on priority basis, either by applying a patch or upgrading the system. If such controls are not possible, strong countermeasures need to be put in place.

HIGH	CVSS Rating Scale: 7.0 - 8.9

High severity vulnerabilities demonstrate some of the following characteristics:

a. It is not straight-forward to exploit and may require some user interaction and has minimum dependency.
b. Usually gives an elevated privilege.
c. It has a significant impact on confidentiality, integrity, and availability

High severity vulnerabilities are defined by some of the following examples:

- Cisco IOS Arbitrary Command Execution Vulnerability (CVE-2012-0384) – **8.8**

- Apple iWork Denial of Service Vulnerability (CVE-2015-1098) – **7.8**
- OpenSSL Heartbleed Vulnerability (CVE-2014-0160) – **7.5**
- Ticketbleed (CVE-2016-9244)– **7.5**
- SSL/TLS MITM Vulnerability (CVE-2014-0224) – **7.4**

MEDIUM	**CVSS Rating Scale: 4.0 - 6.9**

Vulnerabilities scored medium generally:

a. Require specific user privileges or conditions to execute the attack.
b. Grant attacker access to non-critical privileged functionality/data.
c. Moderately affects Confidentiality, Integrity, or integrity.

Medium severity vulnerabilities are defined by some of the following examples:

- MySQL Stored SQL Injection (CVE-2013-0375) - **6.4**
- phpMyAdmin Reflected Cross-site Scripting Vulnerability (CVE-2013-1937) – **6.1**
- Joomla Directory Traversal Vulnerability (CVE-2010-0467) – **5.8**
- Cisco Access Control Bypass Vulnerability (CVE-2012-1342) – **5.8**
- DokuWiki Reflected Cross-site Scripting Attack (CVE-2014-9253) – **5.4**
- FREAK (CVE-2015-0204) – **5.3**
- Apache Tomcat XML Parser Vulnerability (CVE-2009-0783) – **4.2**

LOW	**CVSS Rating Scale: 0.1 - 3.9**

Vulnerabilities scored low usually exhibit some of the following characteristics:

a. They have little impact on the confidentiality, integrity or availability of the organization's data or assets.
b. Requires great amount of computational power to exploit the vulnerability.
c. Require excessive privileges or access to execute an attack
d. Exploits are not known publicly

Low severity vulnerabilities are defined by some of the following examples:

- LOGJAM Attack (CVE-2015-4000) – 3.7
- SSLv3 POODLE Vulnerability (CVE-2014-3566) – 3.1

INFORMATIONAL/NONE CVSS Rating Scale: 0.0

Issues that do not reflect a direct security risk but are informational in nature or may be useful to attackers in specific context. Issues classified as 'Informational' or 'None' may also be risks which could not be verified.

Informational severity vulnerabilities are defined by some of the following examples:

- IIS 7.1/8
- Port Scan Status

This rating is reserved for system vulnerabilities that will result in serious impact to the organization. Depending on the criticality of the system, risks of this magnitude could represent a financial impact or damage customer and partner relationships.