

Project Info

On ΩDATEΩ Lutech S.p.A. has performed for ΩSHORT\_COMPANY\_NAMEΩ an Infrastructure Penetration Test on "ΩREPORT\_NAMEΩ".

The Target Of Evaluation (ToE) includes the following assets: \*\*\*\*\*\*. During tests the following assets have not replied completely or partially to the interactions so it is possibile that these were unavailable, unreachable, did not provide any services or were protected with some security system (i.e. IPS, WAF, etc.): \*\*\*\*\*\*. If none of the above conditions were true it is possible that these assets had no particular vulnerabilities.

ΩSHORT\_COMPANY\_NAMEΩ did not provide any access privileges for the Infrastructure Penetration Test on "ΩREPORT\_NAMEΩ".

The Rules of Engagement (RoE) do not include any kind of restriction.

The Activities were conducted by Lutech S.p.A.’s Ethical Hacking Team specialists using Lutech methodology. The vulnerabilities are evaluated in terms of their own severity. The severity is calculated as a function of vector of attack, attack complexity, required privileges, need to interact with the user, scope and impact in terms of confidentiality, integrity and availability. Details on ’Severity Calculation Model’ chapter.

Activity Details

# Metodology

The Activities were conducted using Lutech S.p.A. methodology. Methodology is based on Lutech Testing Framework that includes elements of the best of breed of testing methodologies like PTES (http://www.pentest-standard.org), OWASP (https://www.owasp.org/index.php/OWASP\_Testing\_Guide\_v4\_Table\_of\_Contents), CEH (http://www.eccouncil.org/Certification/certified-ethical-hacker), SANS (http://pen-testing.sans.org/) and NIST (http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf) methodologies, and includes the security measurable concept by MITRE (http://makingsecuritymeasurable.mitre.org/).

This allows to enumerates common concepts and to use a common information coding language. Enumeration allows to have dictionaries recognized and accepted globally for vulnerabilities (CVE), weaknesses (CWE), attack patterns (CAPEC), configurations (CCE) and platforms (CPE). Common information coding languages provides a way to categorize severities based on objective parameters like access vector, attack complexity, privileges required, user interaction, scope, confidentiality, integrity, availability, exploit code maturity, remediation level, and report confidence.

Methodology involves five steps:

1. Intelligence Gathering, in which, through indirect interactions, information about assets, business organization, people, technologies and potential attack vectors are collected
2. Vulnerability Analysis, where the vulnerabilities are identified, evaluated and classified. Vulnerability Analysis covers the following areas:

* CIFS
* Database
* DNS
* FTP
* LDAP
* NFS
* NTP
* Mail
* Remote Access
* RPC
* SNMP
* SSH
* Web
* Other Infra

1. Exploitation, in which vulnerabilities found in Vulnerability Analysis phase are validated. Proof of exploitation are produced as evidence
2. Post-Exploitation, in which information (files, configurations, accounts, passwords, application/system/network info, etc.) on compromised assets are collected. These information are reused to perform more complex attacks, attacking other assets using compromised assets as pivot, etc.
3. Reporting, in which executive summary, technical summary and remediation plan are produced

Exploitation and Post-Exploitation steps are not performed on Vulnerability Assessment.

# Severity Calculation Model

Vulnerabilities are evaluated using a severity calculation model based on CVSS Standard version 2 (https://www.first.org/cvss/v2/guide). Evaluation captures the characteristics of a vulnerability that are constant with time and across user environments. The Access Vector, Access Complexity, and Authentication metrics capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it. The three impact metrics measure how a vulnerability, if exploited, will directly affect an IT asset, where the impacts are independently defined as the degree of loss of confidentiality, integrity, and availability. For example, a vulnerability could cause a partial loss of integrity and availability, but no loss of confidentiality.

The Score is a function of the Impact and Exploitability sub score equations and is defined as

*Round to 1 decimal(((0.6 × Impact) + (0.4 × Exploitability)−1.5) × f(Impact))*

where "Round to 1 decimal" is defined as the smallest number, specified to one decimal place, that is equal to or higher than its input [for example, Round up (4.02) is 4.1; and Round up (4.00) is 4.0] and f(impact) is:

*f(Impact) = 0 if Impact = 0, 1.176 otherwise*

The Impact is defined as

*Impact = 10.41 × (1−(1−ConfImpact) × (1−IntegImpact) × (1−AvailImpact)) The Exploitability is defined as*

*Exploitability = 20 × AccessV ector × AccessComplexity × Authentication*

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

**Table 1** **- Severity rating mapping**

The Score is mapped to qualitative severity rating as defined on Table 1.

**Level of Exposure** value is based on the most critical vulnerability found: a single vulnerability of maximum severity is a symptom of a high degree of exposition on a single point of break that is enough to compromise the ToE. If the number of vulnerabilities that belongs to the highest severity level is very high in respect to the total number of vulnerabilities found, the Level of Exposure can be raised up to the next level. There is not chance to lower the Level of Exposure to the previous level.

# Metrics

Access Vector (AV)

This metric reflects how the vulnerability is exploited. The possible values for this metric are listed in Table 2. The more remote an attacker can be to attack a host, the greater the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Network (N) | A vulnerability exploitable with network access means the vulnerable software is bound to the network stack and the attacker does not require local network access or local access. Such a vulnerability is often termed "remotely exploitable". An example of a network attack is an RPC buffer overflow. |
| Adjacent Network (A) | A vulnerability exploitable with adjacent network access requires the attacker to have access to either the broadcast or collision domain of the vulnerable software. Examples of local networks include local IP subnet, Bluetooth, IEEE 802.11, and local Ethernet segment. |
| Local (L) | A vulnerability exploitable with only local access requires the attacker to have ei- ther physical access to the vulnerable system or a local (shell) account. Examples of locally exploitable vulnerabilities are peripheral attacks such as Firewire/USB DMA attacks, and local privilege escalations (e.g., sudo). |

**Table 2 - Access Vector (AV)**

Access Complexity (AC)

This metric measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. For example, consider a buffer overflow in an Internet service: once the target system is located, the attacker can launch an exploit at will. Other vulnerabilities, however, may require additional steps in order to be exploited. For example, a vulnerability in an email client is only exploited after the user downloads and opens a tainted attachment. The possible values for this metric are listed in Table 3. The lower the required complexity, the higher the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Low (L) | Specialized access conditions or extenuating circumstances do not exist. The following are examples: - The affected product typically requires access to a wide range of systems and users, possibly anonymous and untrusted (e.g., Internet-facing web or mail server). - The affected configuration is default or ubiquitous. - The attack can be performed manually and requires little skill or additional information gathering. - The race condition is a lazy one (i.e., it is technically a race but easily winnable). |
| Medium (M) | The access conditions are somewhat specialized; the following are examples: - The attack- ing party is limited to a group of systems or users at some level of authorization, possibly untrusted. - Some information must be gathered before a successful attack can be launched. - The affected configuration is non-default, and is not commonly configured (e.g., a vulner- ability present when a server performs user account authentication via a specific scheme, but not present for another authentication scheme). - The attack requires a small amount of social engineering that might occasionally fool cautious users (e.g., phishing attacks that modify a web browsers status bar to show a false link, having to be on someones buddy list before sending an IM exploit). |
| High (H) | Specialized access conditions exist. For example: - In most configurations, the attacking party must already have elevated privileges or spoof additional systems in addition to the attacking system (e.g., DNS hijacking). - The attack depends on social engineering methods that would be easily detected by knowledgeable people. For example, the victim must perform several suspicious or atypical actions. - The vulnerable configuration is seen very rarely in practice. - If a race condition exists, the window is very narrow. |

**Table 3 - Access Complexity (AC)**

Authentication (Au)

This metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability. This metric does not gauge the strength or complexity of the authentication process, only that an attacker is required to provide credentials before an exploit may occur. The possible values for this metric are listed in Table 4. The fewer authentication instances that are required, the higher the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Multiple (M) | Exploiting the vulnerability requires that the attacker authenticate two or more times, even if the same credentials are used each time. An example is an attacker authenticating to an operating system in addition to providing credentials to access an application hosted on that system. |
| Single (S) | The vulnerability requires an attacker to be logged into the system (such as at a command line or via a desktop session or web interface). |
| None (N) | Authentication is not required to exploit the vulnerability. |

**Table 4 - Authentication (Au)**

Confidentiality Impact (C)

This metric measures the impact on confidentiality of a successfully exploited vulnerability. Confidentiality refers to limiting information access and disclosure to only authorized users, as well as preventing access by, or disclosure to, unauthorized ones. The possible values for this metric are listed in Table 5. Increased confidentiality impact increases the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Complete (C) | There is total information disclosure, resulting in all system files being revealed. The attacker is able to read all of the system’s data (memory, files, etc.). |
| Partial (P) | There is considerable informational disclosure. Access to some system files is possible, but the attacker does not have control over what is obtained, or the scope of the loss is constrained. An example is a vulnerability that divulges only certain tables in a database. |
| None (N) | There is no impact to the confidentiality of the system. |

**Table 5 - Confidentiality Impact (C)**

Integrity Impact (I)

This metric measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and guaranteed veracity of information. The possible values for this metric are listed in Table 6. Increased integrity impact increases the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Complete (C) | There is a total compromise of system integrity. There is a complete loss of system pro- tection, resulting in the entire system being compromised. The attacker is able to modify any files on the target system. |
| Partial (P) | Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited. For example, system or application files may be overwritten or modified, but either the attacker has no control over which files are affected or the attacker can modify files within only a limited context or scope. |
| None (N) | There is no impact to the integrity of the system. |

**Table 6 - Integrity Impact (I)**

Availability Impact (A)

This metric measures the impact to availability of a successfully exploited vulnerability. Availability refers to the accessibility of information resources. Attacks that consume network bandwidth, processor cycles, or disk space all impact the availability of a system. The possible values for this metric are listed in Table 7. Increased availability impact increases the vulnerability score.

|  |  |
| --- | --- |
| Metric value | Description |
| Complete (C) | There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable. |
| Partial (P) | There is reduced performance or interruptions in resource availability. An example is a network-based flood attack that permits a limited number of successful connections to an Internet service. |
| None (N) | There is no impact to the availability of the system. |

**Table 7 - Availability Impact (A)**

Findings

Reading Guide

The Reading guide describes the structure of Vulnerabilities chapters. In detail, Vulnerability chapters are hierarchically organized to describe the attack tree used for the assessment phase. Attack tree includes:

* Test Category
* Test
* Finding

**Test Category** describes the macro categories of test. Test categories differs for *application, infrastructure,* and *mobile* assessment. **Test** describes the vulnerability and contains also the external references to go in deep into the problem.

**Finding** describes the instance of a vulnerability for a specific attack point (e.g. IP-Port for infrastructure, Method-URL-Parameter for application, etc.). For each finding there is a vulnerability ID (VID) that is a unique identifier that identifies the issue. Finding also includes:

* Severity: the graphical representation of severity on a 4-level rank (low, medium, high and critical)
* CAPEC: the Common Pattern Enumeration and Classification attack associated reference
* CVSS: the Common Vulnerability Scoring System value calculated for vulnerability
* Confidentiality: the impact on confidentiality of a successfully exploited vulnerability
* Integrity: the impact to integrity of a successfully exploited vulnerability
* Availability: the impact to availability of a successfully exploited vulnerability
* Finding Details: the extensive vulnerability explanation of vulnerability for the specific attack  point that expands the Test definition (if necessary)
* Remediation: one or more remediations for fix vulnerability

If available, there are also the evidences of the vulnerability and related notes.

For Application Findings there are also:

* Method: the HTTP Method used to exploit vulnerability
* URL: the vulnerable URL
* Parameter: the vulnerable parameter
* OWASP: the nearest OWASP Top 10 Security Risk associable to the issue
* CWE: the Common Weakness Enumeration associated reference
* Attack Vector: the context by which vulnerability exploitation is possible
* Attack Complexity: the conditions beyond the attacker’s control that must exist in order to exploit the vulnerability
* Privilege Required: the level of privileges an attacker must possess before successfully exploiting the vulnerability
* User Interaction: the requirement for a user, other than the attacker, to participate in the successful compromise of the vulnerable component
* Scope: the ability for a vulnerability in one software component to impact resources beyond its means, or privilege

For Infrastructure Findings there are also:

* Host: the vulnerable IP
* Port: the port of vulnerable service
* Exploit: the number of exploits available for the vulnerability
* CVE: the Common Vulnerability and Exposure associated reference
* Access Vector: how the vulnerability is exploited
* Access Complexity: the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system
* Authentication: the number of times an attacker must authenticate to a target in order to exploit a vulnerability
  1. For Mobile Findings there are also:
* Platform: the mobile platform affected
* Component: the component affected
* Details: the details associated to the vulnerability
* OWASP: the nearest OWASP Top 10 Mobile Security Risk associable to the issue
* CWE: the Common Weakness Enumeration associated reference
* Attack Vector: the context by which vulnerability exploitation is possible
* Attack Complexity: the conditions beyond the attacker’s control that must exist in order to exploit the vulnerability
* Privilege Required: the level of privileges an attacker must possess before successfully exploiting the vulnerability
* User Interaction: the requirement for a user, other than the attacker, to participate in the successful compromise of the vulnerable component
* Scope: the ability for a vulnerability in one software component to impact resources beyond its means, or privilege

Findings list

|  |  |  |
| --- | --- | --- |
| *Finding Name* | *Type* | *CVSS* |
| Critical Risk Findings |  |  |
| æreport/findings\_list/findings:::cvss\_score>=9æ ∞title∞ | ∞type∞ | ∞cvss\_score∞ |
|  |  |  |
| High Risk Findings |  |  |
| æreport/findings\_list/findings::: cvss\_score <9.0::: cvss\_score >=7.0æ∞title∞ | ∞type∞ | ∞cvss\_score∞ |
|  |  |  |
| Medium Risk Findings |  |  |
| æreport/findings\_list/findings::: cvss\_score <7.0::: cvss\_score >=4.0æ∞title∞ | ∞type∞ | ∞cvss\_score∞ |
|  |  |  |
| Low Risk Findings |  |  |
| æreport/findings\_list/findings::: cvss\_score <4.0::: cvss\_score >0æ∞title∞ | ∞type∞ | ∞cvss\_score∞ |
|  |  |  |
| Informational Findings |  |  |
| æreport/findings\_list/findings::: cvss\_score =0æ∞title∞ | ∞type∞ | ∞cvss\_score∞ |
|  |  |  |

¬report/findings\_list/findings::: cvss\_score>=9¬

|  |  |  |
| --- | --- | --- |
| **πtitleπ** | |  |
| CVSS | πcvss\_scoreπ | |
| Severity | πcvss\_severityπ | |
| Type | πtypeπ | |

***Summary***

¬overview/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Proof***

¬poc/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Remediation***

¬remediation/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***References***

¬references/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Affected hosts***

¬affected\_hosts/paragraph¬π.π∆

∆

¬report/findings\_list/findings:::cvss\_score<9.0:::cvss\_score >=7.0¬

|  |  |  |
| --- | --- | --- |
| **πtitleπ** | |  |
| CVSS | πcvss\_scoreπ | |
| Severity | πcvss\_severityπ | |
| Type | πtypeπ | |

***Summary***

¬overview/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Proof***

¬poc/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Remediation***

¬remediation/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***References***

¬references/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Affected hosts***

¬affected\_hosts/paragraph¬π.π∆

∆

¬report/findings\_list/findings:::cvss\_score<7.0:::cvss\_score >=4.0¬

|  |  |  |
| --- | --- | --- |
| **πtitleπ** | |  |
| CVSS | πcvss\_scoreπ | |
| Severity | πcvss\_severityπ | |
| Type | πtypeπ | |

***Summary***

¬overview/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Proof***

¬poc/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Remediation***

¬remediation/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***References***

¬references/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Affected hosts***

¬affected\_hosts/paragraph¬π.π∆

∆

¬report/findings\_list/findings:::cvss\_score<4.0:::cvss\_score>0¬

|  |  |  |
| --- | --- | --- |
| **πtitleπ** | |  |
| CVSS | πcvss\_scoreπ | |
| Severity | πcvss\_severityπ | |
| Type | πtypeπ | |

***Summary***

¬overview/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Proof***

¬poc/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Remediation***

¬remediation/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***References***

¬references/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Affected hosts***

¬affected\_hosts/paragraph¬π.π∆

∆

¬report/findings\_list/findings:::cvss\_score=0¬

|  |  |  |
| --- | --- | --- |
| **πtitleπ** | |  |
| CVSS | πcvss\_scoreπ | |
| Severity | πcvss\_severityπ | |
| Type | πtypeπ | |

***Summary***

¬overview/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Proof***

¬poc/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Remediation***

¬remediation/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***References***

¬references/paragraph¬ µzzzzµ π.π

ƒcodeƒ π.π

ƒitalicsƒ *π.π*

* ƒbulletƒ π.π

ƒh4ƒ **π.π**

÷ π.π ≠

***Affected hosts***

¬affected\_hosts/paragraph¬π.π∆

∆