Vulnus: Visual Vulnerability Analysis for Network Security

Article *in* IEEE Transactions on Visualization and Computer Graphics · August 2018

DOI: 10.1109/TVCG.2018.2865028

5 authors, including:

Marco Angelini
Sapienza University of Rome
64 PUBLICATIONS **453** CITATIONS
SEE PROFILE

Graziano Blasilli
Sapienza University of Rome
11 PUBLICATIONS **42** CITATIONS
SEE PROFILE

Tiziana Catarci
Sapienza University of Rome
263 PUBLICATIONS **2,788** CITATIONS
SEE PROFILE

Simone Lenti
Sapienza University of Rome
16 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Vulnus: Visual Vulnerability Analysis for Network Security

Marco Angelini, Graziano Blasilli, Tiziana Catarci *Member, IEEE*, Simone Lenti, and Giuseppe Santucci *Member, IEEE*

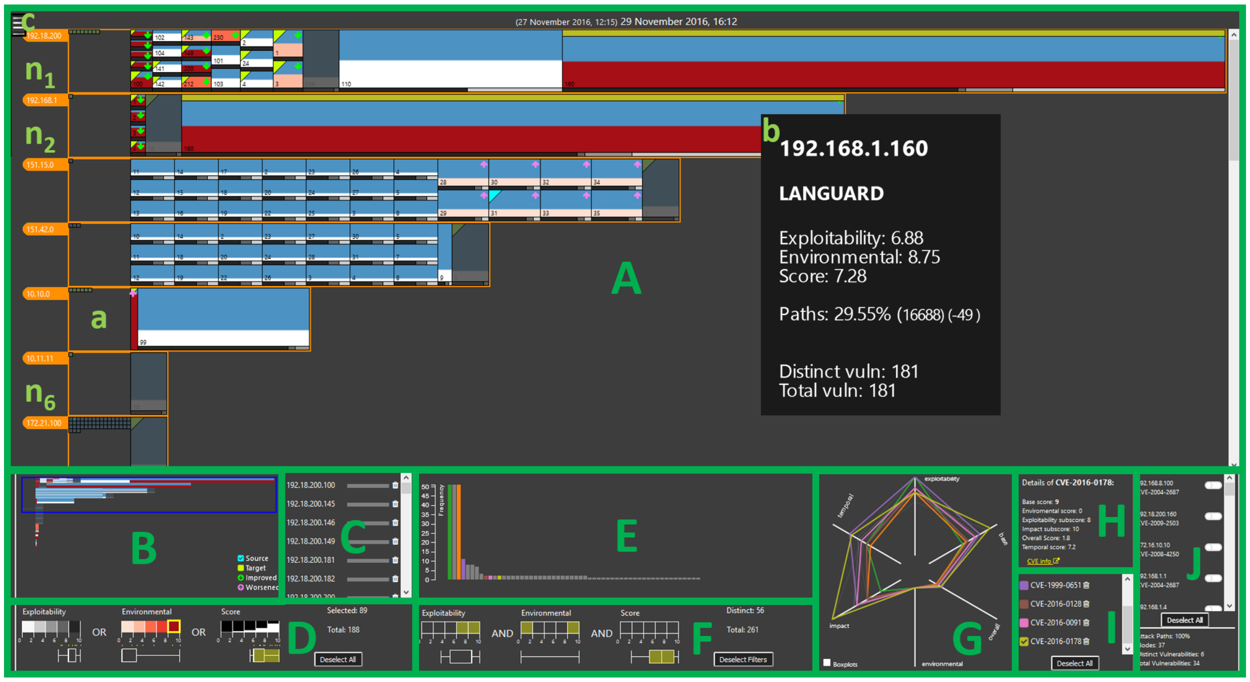


Fig. 1: VULNUS dealing with a real network containing 122 nodes, 22 sub-networks and 846 vulnerabilities. Sub-networks are arranged horizontally, using a modified treemap bar chart representation. Each bar represents a sub-network ($n_1, n_2, \ldots$), showing its nodes; both the node size and the bar length are proportional to the total number of vulnerabilities of the element. Bars are sorted by length and while the user is scrolling into the main pane (A), the pane (B) maintains the visual context. The leftmost part of each bar (a) contains nodes with zero vulnerabilities that are represented as squares of fixed dimension. Mouse-hovering a node (b) reveals its CVSS scores and the number of vulnerabilities. Nodes are selectable individually or by sub-network into the main pane and by uniform and statistical intervals of their scores in the selection pane (D). Selected nodes are listed in (C); their vulnerabilities are presented on a frequency plot (E) and filterable by scores (F). Further selections on the frequency plot are listed in (I), allowing for comparing vulnerability scores on a radar diagram (G) or inspecting numerical scores and accessing the CVE reference page (H). The approximated optimal fixing strategy (J) is automatically computed using the attack graph information and VULNUS allows the security manager to interactively simulate the vulnerabilities fixing in order to explore suboptimal variants of the fixing plan.

Abstract—Vulnerabilities represent one of the main weaknesses of IT systems and the availability of consolidated official data, like CVE (Common Vulnerabilities and Exposures), allows for using them to compute the paths an attacker is likely to follow. However, even if patches are available, business constraints or lack of resources create obstacles to their straightforward application. As a consequence, the security manager of a network needs to deal with a large number of vulnerabilities, making decisions on how to cope with them. This paper presents VULNUS (VULNerabilities visUal aSsessment), a visual analytics solution for dynamically inspecting the vulnerabilities spread on networks, allowing for a quick understanding of the network status and visually classifying nodes according to their vulnerabilities. Moreover, VULNUS computes the approximated optimal sequence of patches able to eliminate all the attack paths and allows for exploring sub-optimal patching strategies, simulating the effect of removing one or more vulnerabilities. VULNUS has been evaluated by domain experts using a lab-test experiment, investigating the effectiveness and efficiency of the proposed solution.

Index Terms—Visual Analytics, Network security, Vulnerability analysis, CVE, CVSS, Attack Graph, Vulnerability triage and management

1 Introduction

Software vulnerabilities are getting more and more attention due to the central role they play as entry points for cyber attacks; centralized repositories of vulnerabilities like CVE (Common Vulnerabilities and Exposures) [41] and associated metrics like CVSS (Common Vulnerability Scoring System) [17] are publicly available and software developers continuously produce patches to fix them. However, in spite of this effort, dealing with vulnerabilities is still a challenging activity, especially when, even if fixes are available, organization mission constraints create obstacles to their straightforward application. As an example, applying a patch might require a system restart, producing an unacceptable interruption of one or more critical services. Consequently, it is quite common that computer networks provide their services while several, known and unfixed vulnerabilities are spread

• *Authors are with University of Rome "La Sapienza". E-mail: [surname]@diag.uniroma1.it.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

across the system. The practical consequence of this situation is that the security managers of complex networks have to deal with a high number of vulnerabilities, trying to get an overall understanding of the situation, in order to prioritize interventions and select appropriate fixes (e.g., applying a patch, closing an IP port, isolating the vulnerable node). New challenges arise from including in the analysis the possible attack paths that the intruders can follow (see, e.g., [34]) and little or no attention has been given to solutions that have the main objective of making sense of large sets of vulnerabilities taking into account a) their intrinsic criticality (as modeled by the CVSS metrics) and b) their actual impact, specific to the organization under analysis, in terms of their role in allowing to reach critical target nodes, through one or more attack paths.

In this paper, we deal with such issues presenting VULNUS, a visual analytics solution that computes the role that each vulnerability has in the attack paths and visualizes such value together with other assessed characteristics (i.e., CVSS metrics), with the overall goal of increasing the situational awareness of the security managers and giving them a means for detecting and prioritizing vulnerability fixes. The proposed solution allows for dynamically inspecting and comparing the characteristics of vulnerabilities spread on a network and simulating the effect that vulnerability fixes have on the system. It allows for getting a quick understanding of a) the network status, by visually classifying nodes according to their vulnerabilities, b) the vulnerabilities impact, either on the nodes that host them and on the whole organization, in terms of the risk they pose on critical nodes, and c) the approximated optimal fixing strategy blocking all the attack paths. Using such an understanding, the security manager can either apply the proposed solution or identify different suboptimal fixing strategies, according to resources, time, and business constraints. Network elements are represented using a modified treemap bar chart visualization based on the vulnerabilities distribution, and adding on it visual marks that convey the most relevant figures. Coordinated panes allow for selecting elements, inspecting and filtering the vulnerability frequency distribution and for comparing vulnerabilities against both CVSS metrics and analytically computed measures. Elements can be grouped according to different paradigms (e.g., sub-networks, CVSS scores, vulnerabilities) in order to deal with large networks and to enable different analysis patterns. The proposed approach has been developed in the context of the FP7 European Panoptesec project [42] along a three years user-centered design activity involving two security managers and four operators of the ACEA company, a large Italian public organization that supplies energy and water to millions of people.

Summarizing, the contributions of the paper are the following:

- a modified treemap bar chart [39] visualization, integrated with several coordinated panes and filtering mechanisms that allows for making sense of large set of vulnerabilities spread across multiple sub-networks;
- an automatically computed attack graph based analytical score, Target Environmental, that complements the standard CVSS scores, providing valuable information on vulnerabilities severity in terms of the critical target nodes they contribute to compromise;
- a what-if simulation scenario that allows the user to understand the effects of fixing vulnerabilities: the user can interactively explore and modify the result of a prioritizing analytical strategy that automatically computes the approximated optimal sequence of fixes able to block *all* the existing attack paths;
- a controlled user study, assessing through a lab-test experiment both effectiveness and efficiency of the proposed solution.

The paper is structured as follows: Section 2 discusses related research proposals, Section 3 describes the application domain, Section 4 describes the supported activities, Section 5 details the VULNUS system, Section 6 presents the user study, and Section 7 concludes the paper.

2 Related Work

The application of visual analytics techniques to the cyber-security domain is a well-known research approach (see, e.g., [4, 12]). It encompasses many different topics, ranging from network status [20, 21], to intrusion detection, to log and code analysis [9, 19]. As stated in [3, 35], a critical cyber security issue is the presence of vulnerabilities in a node: exploiting a vulnerability can be the first step for attacking/penetrating a network and recent works acknowledge the fact that security administrators have to deal with non-traditional threats, "e.g., how to live with known, but unpatchable vulnerabilities", see [6].

A vulnerability exploitation is often the initial step of a multistep attack, modeled through an attack path; studies on how to use the vulnerabilities characteristics for enriching the attack model are present in [18] and [10]: our work follows a similar approach but adds the capability to use the resulting enrichment to estimate the effects of a fixing strategy through an algorithmic solution steered by the security manager. Different contributions dealing with the visualization of attack scenarios exist in the literature (see, e.g., [2, 11, 31]). Most of them focus on the visualization and analysis of cyber events at different levels of detail, like [29] and [15]. In particular, the works in [15, 16, 28, 30] use treemaps [36] for visualizing security data and events; however, none of these works copes with vulnerabilities, and our work does not use the classic treemap paradigm [7] but proposes a novel ensemble approach with modified treemap bar chart, allowing to dynamically prioritize elements for analysis, maintaining the right aspect ratio and supporting comparisons among elements. Some other works have sparse information on vulnerabilities, e.g., the work in [46] that allows for a basic grouping of nodes with similar vulnerabilities.

To the best of the authors knowledge, most of the cyber-security commercial tools provide limited visual information about vulnerabilities; among them we inspected CheckPoint NGSE [8], Imperva SecureSphere [24], IBM QRadar [23], RSA enVision [33]. Most of them deal with vulnerabilities in a tabular way with at most simple visualization paradigms (e.g., bar-charts, pie-charts, geographical maps), and do not support more abstract visualization paradigms, best suited for coping with large networks. Following the study in [44] we choose to use an abstract visualization to represent vulnerabilities information, augmenting the characteristics of the treemap elements in order to provide simultaneously all the important aspects of a vulnerability.

A notable exception is the case of VisiTrend [43]: this tool proposes a visual environment representing the network topology in various ways (node-link diagram, treemap) and plotting on them cyber-security parameters. One of its instantiations is directly connected to the visualization of vulnerabilities, using, among other visualizations, a treemap: each rectangle represents a node, with its size mapped to the number of vulnerabilities and its color mapped to the vulnerability state. Our solution improves this approach by allowing to compute and homogeneously visualize important parameters of the CVSS like Exploitability and Target Environmental scores, completely missing from VisiTrend. In particular our visual analytics solution provides a way to automatically compute and visualize the Target Environmental score, a crucial metric for assessing the impact a vulnerability can have on the instantiated network on which it is exploited. Moreover, our solution proposes additional analyses, ranging from single vulnerability analysis to distribution of nodes with respect to the CVSS values, that are not provided by Visitrend, and that contributes important awareness in the comprehension of the vulnerability status of a network.

The work in [22] constitutes a main contribution that coped with this problem presenting NV, a Web-based solution that utilizes treemaps and linked histograms to allow security managers to discover, analyze, and manage vulnerabilities on their networks. Starting from it as an inspiration, we developed VULNUS addressing issues about scalability with respect to the chosen visual paradigm (proposing a comprehensive overview of the vulnerability status of the network in a single view, in contrast with NV that uses layers for different information and asks for several interactions in order to obtain the same information as VULNUS) and introducing additional features that are crucial for raising the situational awareness of the security operator, by conveying relevant vulnerability information (cardinality, dangerousness, exploitability, spread, and impact) in the same representation. Moreover, vulnerabilities are presented at different levels of aggregation (e.g., whole network, sub-networks, single nodes), with the goal of characterizing and comparing several vulnerabilities to identify the most dangerous ones, and inspecting the spread of one or several vulnerabilities through the network, providing a better understanding of the network status, useful to make decisions and prioritize the fixes.

Concerning the analytical engine, VULNUS introduces the notion of Target Environmental, extending the preliminary idea presented in [32], "edges that are common to many δ-optimal paths are likely to represent vulnerable points" and the more recent notion of "attack criticality score" presented in [37] for scoring "attack capabilities which are used by many successful attack methods", formulated as a counting variant of Boolean Satisfiability. In particular, the VULNUS Target Environmental score is defined in the context of the CVE/CVSS environments and computed directly on the attack graph; by using the polynomial greedy set cover algorithm that comes with a guaranteed approximation for calculating the optimal fixing strategy VULNUS computes the *approximated optimal fixing strategy*, AOF in what follows. In this way VULNUS can deal with large attack graphs and supports an interactive what-if analysis useful when the AOF cannot be implemented due to business constraints. Moreover, this metric has been encoded in the system as a first class pre-attentive visual variable, facilitating the identification of critical parts of the network.

3 Application Domain

The main goal of VULNUS is to allow security managers to assess the spread, impact, and dangerousness of the vulnerabilities that affect a network in order to understand the whole picture and to prioritize fixes. The proposed approach relies on both general and business specific vulnerabilities characteristics that are modeled using the CVSS scores and the ad-hoc computed Target Environmental score, respectively. Unique to our approach is that the Target Environmental score is automatically computed, using information coming from the attack graph associated with the actual detected vulnerabilities and network topology, avoiding the burden of manually evaluating and inputting such data.

This section presents a high level overview of the attack graph approach and the CVSS metrics, useful for better understanding the visual analytics solution described in Section 5.

3.1 Attack Graphs

Attack graphs [27, 34] are widely recognized as a tool of choice to model and better understand the various possible successions of steps that attackers could perform to reach their objectives. Each step is represented by the exploitation of a vulnerability (belonging to the set of known network vulnerabilities V) on a node (belonging to the set of network nodes N) to gain a non-intended access to its resources.

Some specific nodes are flagged as *target nodes* T, i.e., nodes that are goals for the attackers. These nodes are typically mission-critical, i.e., nodes that, if reached by the attackers, can compromise relevant parts of the mission of the organization they belong to. Other nodes are flagged as *source nodes* S, i.e., nodes that an attacker could use to initiate a multi-step attack, and typically they correspond to nodes that are visible on the Web.

Links between nodes represent the possible transitions for an attacker to go from one step of the attack to the next one. The sequence of steps that, starting from a source node, reaches a target node is an *attack path*; the union of all the attack paths AP produces the *attack graph*, i.e., a direct multi-graph.

An attack path ap is modeled as a sequence of *exploitation steps*:

$$ap = < n_0 >, < v_1, n_1 >, \ldots, < v_k, n_k > \qquad (1)$$

where $v_i \in V$, $n_0 \in S$, $n_k \in T$, and $n_i \in N \; for \; i = 1 \ldots k-1$, representing a path, from the source node n_0 to the target node n_k, that is traversed exploiting the vulnerabilities $v_1, \ldots, v_k$ on nodes $n_1, \ldots, n_k$. Attack paths overlap each other, in terms of nodes and exploited vulnerabilities and we define the set ES of all possible exploitation steps as the set of all tuples <v, n> that appear in any attack path.

While in other cyber-security applications attack graphs are used to display the possible paths to target nodes or the progress of an actual attack (see, e.g., [2]), in this paper an attack graph is used to associate a Target Environmental score to each pair $< vulnerability, node >$ reflecting the vulnerability criticality in terms of the target nodes it allows to reach, through one or more attack paths that exploit *vulnerability* on the specific *node*.

3.2 CVE and CVSS

Vulnerabilities are continuously discovered, analyzed and stored in public databases like the CVE (Common Vulnerability and Exposures) NIST National Vulnerability Database [41] and several metrics have been developed to estimate their severity; in the following we provide an overview of CVSS (Common Vulnerability Scoring System) that organizes the metrics in three main groups: 1. **Base metrics**, mandatory, representing immutable aspects of a vulnerability; e.g., they encompass information on the complexity of the steps needed to exploit it, i.e., *Exploitability score*, and on the exploitation impact on the node in which the vulnerability exists; 2. **Temporal metrics**, optional, representing aspects that may change over time; 3. **Environmental metrics**, optional, that specialize the vulnerability impact to the environment in which it exists.

These groups are summarized by *Base score*, *Temporal score*, and *Environmental score* that are used to compute the *Overall score*, ranging between 0 and 10, used to summarize the overall vulnerability severity. The complexity of the CVSS structure pushes most of the network security managers to inspect only the Overall score, neglecting details. The system presented in this paper tries to overcome such a problem visually encoding a) the Base score, b) the paper defined Target Environmental score, and c) the Exploitability score that is a key component of the Base score. In this way the security manager is presented with an overview of the nodes status and vulnerabilities main characteristics.

It is worth noting that the CVSS Environmental score is computed without considering the network topology, and, as a consequence, it provides an estimation of the vulnerability severity that is independent of the target nodes it contributes to reach. VULNUS, instead, relies on the Target Environmental score, characterizing the impact of vulnerabilities exploitation in terms of the potential impact on target nodes.

4 Activities and tasks collection

Before the implementation of VULNUS, the ACEA security operators were used to monitor the network through recurrent vulnerabilities scans, using vulnerability scanner tools like Nessus [40] that ranks vulnerabilities on five severity levels (Critical, High, Medium, Low, Info) using CVSS and produces textual reports organized by vulnerability or host, allowing to compare scan results to highlight changes, still in textual format. When a scan reveals new vulnerabilities in the range Critical/High or new vulnerabilities affecting relevant servers, security operators make a decision about a fix plan, balancing vulnerabilities criticality and business constraints; it is worth to recall, however, that even disregarding business constraints, the fix plan based on CVSS is very far from the optimum, due to the absence of the VULNUS Target Environmental metric, see Figure 5. The fix plan is then applied, activity that is out of the scope of VULNUS. When a new scan is available (after or during the fix plan application), the security operators analyze the new situation, in which some old vulnerabilities have been fixed and new ones have been discovered, and start the process again. In summary, disregarding the quality of the fix plan, the security operators were cycling along three main phases: monitoring the network status, analyzing the vulnerability level of criticality, and making decision on the fix plan. The goal of VULNUS was to better support these phases and we conducted an iterative requirements collection with 2 security managers and 4 security operators of the ACEA company aiming at identifying the main activities related to the management of the network vulnerabilities. Results are summarized in the following, specializing activities in main tasks (labeling the major original users' requirements with RXy):

- **R1 - Monitoring**: this activity is carried on every time a new scan is available, and allows the security manager to quickly identify areas of possible risk and getting insights on the global status of the environment. It is further decomposed in:
  - **R1a - Top Level Overview**: this task allows the security manager to identify the global network vulnerabilities status and the elements that requires detailed analysis ("I do not like zooming in, I want the overall information on the nodes", "I want to compare both nodes and sub-networks");
  - **R1b - Changes Identification**: this task allows the security manager to keep track of the changes in the vulnerability status due to several events, e.g., the installation of a new software with a vulnerability, the discover of a previous unknown vulnerability, the fix of an existing one ("I do not care inspecting nodes with zero vulnerabilities","I want to locate immediately worsened elements").
- **R2 - Analysis**: this activity allows the security manager to conduct analyses on the elements that constitute the environment: given the broad analysis possibilities, we further refined this activity in the following tasks, driven by the granularity of the analysis:
  - **R2a - Node driven analysis**: this analysis is conducted on a node basis (a single appliance of the network) and allows to comprehend and evaluate the cyber-security status of a set of nodes of the network ("I want to quickly locate the most vulnerable nodes", "I want to distinguish between source and target nodes");
  - **R2b - Vulnerability driven analysis**: this analysis is conducted starting from the occurrences of vulnerabilities in the network; e.g., a security manager focuses on a specific vulnerability and wants to evaluate its spread in the network, how it compares to other known vulnerabilities, which risk it creates on the node in which it is instantiated, and, more important, on the whole network ("I prefer to focus on one aspect at time, like first inspecting a bad node/sub-network and after that to deal with its vulnerabilities");
  - **R2c - Metrics driven analysis**: in this case, security operators conduct their analysis using some numerical characteristics of the vulnerabilities; e.g., they need to analyze all the nodes above a specific threshold level of Exploitability, Base, or Target Environmental scores, or a combination of them, ("I want to group elements by different scores").
- **R3 - What-if analysis**: this activity allows the security manager to decide the fix plan, inspecting the automatically generated optimal fixing strategy, tuning it with respect to business constraints. This is obtained simulating the effects that fixes would have on the network in terms of reducing vulnerabilities exposure and related risks. At each step, the security manager can include the simulation in the current fix plan or choose a different fix, balancing the reduction of risk exposure with her or his knowledge about resources, costs, and business constraints ("I want to check the effect of a patch", "I want to see the role a vulnerability has in compromising target nodes").

5 The VULNUS solution

In order to perform the activities and tasks described in the previous section, VULNUS combines together data coming from vulnerabilities and network scanners (i.e., OpenVas [13], Nessus [40] , and LanGuard [38]), which are able to identify the set of vulnerabilities affecting the inspected nodes and their connectivity. Manual intervention is only required to define the set of source nodes, i.e., the nodes that an intruder can use as starting point of an attack and the set of target nodes, i.e., the nodes that host relevant organization processes. Data about vulnerabilities are matched with the metrics and definitions contained in the CVE and CVSS repositories; these data, together with the connectivity, source nodes, and target nodes are used to compute the attack graph associated with the scanned network. An ad-hoc module, developed within the Panoptesec project [42], triggered by changes in vulnerabilities or topology, generates the attack graph and computes the Target Environmental metric; it has been tested against a synthetic network of 10,000 nodes containing 20 source nodes and 100 target nodes, and it is able to generate a new attack graph in less than 3 minutes; however, further detailing such activities is out the scope of the paper.

5.1 Analytical component

The availability of information about the attack graph allows for computing the Target Environmental score and prioritizing the vulnerability fixing. We consider two different kinds of vulnerability:

1. vulnerabilities that are not involved in any exploitation step. Such vulnerabilities represent a security risk only for the nodes on which they exist and are characterized by standard CVSS scores;
2. vulnerabilities that are involved in at least one exploitation step. Such vulnerabilities represent a security risk also for the target nodes they contribute to reach and we characterize them with both the standard CVSS scores and the Target Environmental one.

5.1.1 Computing the Target Environmental score

The Target Environmental score $TE(es)$ aims at defining the dangerousness of a vulnerability for the target nodes, and it is computed considering the set of all the attack paths AP, defining for each exploitation step $es =< vulnerability, node >$ the set $S(es)$ of all the attack paths that contain es. In particular, $TE(es)$ is defined as the proportion (normalized in the range $[0,10]$, to be consistent with the CVSS standard) of the attack paths that use es. Intuitively the more attacks use a specific exploitation step, the more that step is dangerous for the environment. Formally:

$$TE(es) = 10 \cdot MinMaxNormalization(\frac{|S(es)|}{|AP|}) \tag{2}$$

5.1.2 Prioritizing Vulnerability fixing

The simplest way to block an attack path ap threatening the target node t is to fix any of its exploitation steps $es =< v,n >$, e.g., patching the vulnerability v on node n, or closing the IP port that is used for the exploit. It is worth noting, however, that the target node t can be still reached by other attack paths that need to be blocked in order to secure it; moreover the number of attack paths to consider become very large even for small networks, making the analysis and decision process an hard task. For this reason, the VULNUS system computes the approximated optimal fixing strategy AOF that is a partially ordered sequence of exploitation steps that have to be fixed in order to block all the attack paths; the computed solution minimizes the AOF cardinality with guaranteed approximation factor relative to the optimum. The consideration that two or more attack paths may share the same exploitation step $es =< v,n >$ and the definition of $S(es)$ allowed for reformulating this optimization process in terms of set cover problem that, in its most general form, is defined as follows.

Let U be a set of elements and $SU = \{S_1, \cdots, S_m\}$ a set of subsets of U. A set cover is a collection C of these subsets whose union is equal to U. Each subset S_i has an associated weight $w_i \geq 0$ and the goal of the problem is to find a collection C that minimizes the sum of the weights. A simpler version of the problem is setting all weights to the same value, for example $w_i = 1$. In this case the solution is optimal when it contains the smallest number of subsets. The problem is NP-Hard, but there exists a greedy polynomial time algorithm that produces a solution, i.e., AOF, with guaranteed approximation factor relative to the optimum. The greedy algorithm maintains a set R of uncovered elements and builds the solution one set at a time: at each step, the chosen element is the subset that minimizes

$$m_i = \frac{1}{|S_i \cap R|} \tag{3}$$

In our approach U is the set of attack paths AP and $SU = \{S(es) | es \in ES\}$, i.e., the set of all possible $S(es)$, with associated weights equal to 1. Solving this set cover problem produces an approximated solution C minimizing the number of subsets, i.e., the number of vulnerabilities to be fixed. Elements belonging to C are ordered using their cardinality, producing a list $FS = S(v_1,n_1), S(v_2,n_2), \ldots, S(v_k,n_k)$ that provides the minimum number (k) of vulnerability fixes needed to block *all* the attack paths to the user; moreover, the (partial) order provides an indication of the fixing efficacy: fixing v_1 on node n_1 will maximize the number of blocked attack paths with a single fix (disregarding the set covering solution approximation).

However, given a critical network there might be some vulnerabilities in AOF that cannot be fixed; in such a case VULNUS allows for simulating the fixes in any order and, once a vulnerability is removed, the partial order is recomputed, giving the user the possibility of manually exploring a suboptimal of AOF. It is worth noting that each time the AOF is (re)computed, the top selected $S(v_1,n_1)$ is characterized by the highest target environmental score: using equation 2, and considering that in the first algorithm step $R = AP$ we can rewrite m_i (equation 3) as follows:

$$m_i = \frac{1}{|S(es) \cap R|} = \frac{1}{|S(es)|} = \frac{1}{\frac{TE(es)\cdot|AP|}{10}} = \frac{10}{TE(es)\cdot|AP|} \qquad (4)$$

that is minimized by the maximum of TE(es). That provides a nice intuition of the TE(es) meaning: each time a new AOF list is computed, the top element has the highest TE(es).

5.2 Visual component

In VULNUS we adopted the structure of the classical User Centered Design (UCD) approach, based on iterating the activities of specifying the context of usage; collecting and eliciting user and organizational requirements; producing designs and prototypes; carrying out assessment and evaluation, until the system satisfies the specified requirements. However, we added to classical UCD a participatory flavor. Indeed, while in classical UCD the design&development team generates solutions placing users mainly in a reactive role, in participatory design users take a more active involvement in the design process and become a key group of stakeholders. In VULNUS this active role was played by two security managers and four operators of ACEA who worked with the team for three years in a cyclic process that eventually converged to produce the proposed system. As a consequence of this tight interaction, a large number of the initial visualization choices have been modified according to the users' input (e.g., the position of node with 0 or few vulnerabilities on the left side of the modified treemap, the slope of the window covering the nodes and representing the base score, etc.).

The requirements were mainly collected by direct observation of the operators dealing with vulnerabilities, semi-structured interviews of the key users involved in the design process, comparative study of the potential competitors. Scenarios and tasks have then been elicited out of the requirements and assessed by the key users.

Here we report the main design choices, binding them to the relevant requirements.

5.2.1 Global visualization

One of the most challenging requirements was to provide the user with an effective overview of vulnerability spread, cardinality, and severity (Section 4, requirement R1a). The initial solution, based on standard treemaps with node size proportional to vulnerabilities cardinality and organized in a hierarchy composed by network level, nodes level, and vulnerability level, failed to satisfy this requirement. Indeed, descending the tree has the well known drawback of requiring complex browsing techniques (see, e.g., [5]) to guide the navigation and increases the chance for the user to be lost in the information space. The feedbacks provided by the users produced the visualization shown in Figure 1.*A*, that fulfills the objective of getting an overview both at node and sub-network levels, conveying vulnerabilities cardinality and type of nodes (source, target, not vulnerable, and vulnerable nodes). In particular, the proposed visualization is a modification of the treemap bar chart [39], accommodating nodes that have a null area (i.e., nodes with no vulnerability) in a fixed area on the left, satisfying the requirements of easily distinguish nodes with no vulnerabilities and having all nodes visible on the screen. The main pane (Figure 1.*A*) represents the focus of the system: networks are represented through modified treemap bar charts, in which each bar contains the elements that belong to that particular group, either sub-networks, nodes, or vulnerabilities. At high level the bar chart organization allows to show a prioritized view with respect to the total size of the groups: for example, in (Figure 1.*A*), the groups are sub-networks and the elements are nodes, with the size encoding the number of vulnerabilities on each node. The visualization prioritizes the sub-networks with highest number of vulnerabilities ordering them from the top to the bottom of the screen. The ordering is replicated inside each bar, with nodes ordered from left to right by ascending number of vulnerabilities. In order to allow comparisons, each bar is scaled proportionally to the top-most one, effectively minimizing the change in aspect-ratio of the leaves and allowing a consistent comparison among elements belonging to different bars. The visual encoding of a single element of a bar is described in the following.

5.2.2 Node representation

Requirements from ACEA security managers pointed out that the number of vulnerabilities (Section 4, requirement R2b), the Base score, the Exploitability score, and the Target Environmental score were the most relevant measures to visually encode on each node (Section 4, requirement R2c), and we converged on the idea of representing each visual attribute, but the size, in five intervals, increasing the separation among visual attribute values; the agreed final version of the visual encoding is the following (see Figure 2):

- The size encodes the cardinality of vulnerabilities;
- Nodes have a blue background and the mean of the Base scores is represented by a five position sliding window (Figure 2.*a*), from the bottom to the top (bottom=0, top=10), progressively covering the blue background of the node, giving the visual impression of how much the actual node is threatened by its vulnerabilities.
- The maximum of the Target Environmental scores is encoded on the sliding window color, with a five values red scale (Figure 2.*b*), where strong red corresponds to the highest and more dangerous interval;
- The distribution of the vulnerability Exploitability scores is encoded with a horizontal thin bar-gram (Figure 2.*c*) that represents five Exploitability score intervals using a five level grey scale (black = easy to exploit, white = hard to exploit). The design choice prevents simultaneous brightness contrast, while the Chevreul illusion increases the separation between adjacent intervals (see, e.g., [45]);
- On the top left of the node, the presence of a colored triangle (Figure 2.*d*) indicates whether the node is a target (yellow triangle) or a source (light blue triangle).

The system allows to select a color blind safe scale for color blind people. This visual encoding allows security managers to relate several values by looking at the visual elements, quickly identifying problematic nodes from a multidimensional point of view. Some color patterns clearly emerge as dangerous situations, e.g., *black + red = bad*, i.e, high Exploitability and high Target Environmental score: risk for the whole system, or *black + very high and light red sliding window*: high risk only for the actual node. To support the detection of changes in the vulnerability status of the network (Section 4, requirement R1b), an arrow is shown on the top right of the node (Figure 2.*e*). Its presence indicates that the node Target Environmental score raised (upward pink arrow) or lowered (downward green arrow) with respect to the previous scan.

VULNUS visual encoding can scale to relatively large networks: we have tested it displaying at node level a synthetic network hosting 25,000 vulnerabilities spread across 10,000 nodes and 25 subnetworks.

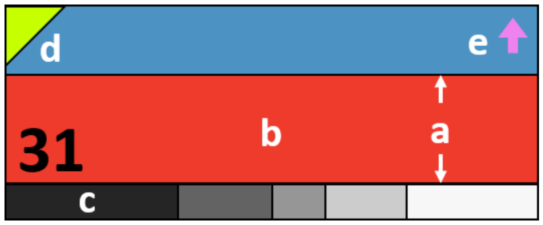


Fig. 2: Encoding node type and scores on the node 31. **a)** the base score is encoded by a red sliding windows that progressively covers the node blue background and it is in the fourth level (range [6, 8)); **b)** the target environmental score is encoded by the sliding window red color and it is in the fourth level (range [6, 8]); **c)** the exploitability bar-gram shows that about 25% of the vulnerabilities have a very high exploitability (black, range [8, 10]); **d)** the yellow triangle classify the node as a target node; **e)** the pink upwards arrow indicates the Target Environmental has increased with respect to the last scan.

Tuning the height of the modified treemap bar chart guarantees to the smallest nodes enough room for the visual encoding; however, it is worth noting that relevant nodes are always easily detectable thanks to the encoding design: node with high number of vulnerabilities are large and on the right of the bar; high Target Environmental nodes (red color) are preattentively detected, even if of small size.

As a last consideration, it is worth pointing out that going above these figures is not the target of the VULNUS system: it is unlikely that a single cyber control center is managing the security of critical networks hosting tens or hundreds of thousands nodes.

5.2.3 Analysis instruments

VULNUS provides different analysis patterns that rely on the basic iteration of two steps:

- Incremental selection of interesting elements according to different perspectives (Section 4, requirement R2a);
- Inspecting (and filtering) the associated frequency distribution of the vulnerabilities belonging to the selected elements (Section 4, requirement R2b).

VULNUS provides different ways to select elements: by manual selection, by group selection, by type selection, and by metric interval selection. It is possible to select elements from the main pane (Figure 1.*A*) by clicking individual elements on the bars or by clicking on the orange whisker on the left of a bar to select all its elements. Furthermore, from the context pane (Figure 1.*B*) the user can select elements of four types: the source nodes, the target nodes, the nodes whose Target Environmental score has decreased with respect to the previous scan of the network, and those for which it has increased. Finally, it is possible to select elements with respect to their scores interacting with the selection pane (Figure 1.*D*). Filtering operations are applied by the interactive legends, showing the visual encodings used for the three metrics (Base, Exploitability, Target Environmental scores); it is possible to click on one or more intervals for each metric adding to the current selection all the elements that have a score belonging to these intervals. Each metric is accompanied with an interactive box-plot that shows the distribution of its values in all the elements of the network.

Once the security manager has made a selection, he or she can analyze the frequency distribution (Figure 1.*E*) of the vulnerabilities of the active selection (or the ones of the whole network if there is no active selection). Mouse-overing on a vulnerability triggers the list of its scores (Figure 1.*H*) and their plotting on the the radar-chart (Figure 1.*G*). The security manager can select a vulnerability by clicking on the associated bar adding it to the list of selected vulnerabilities (Figure 1.*I*) that are all highlighted on the radar-chart. Clicking on an element of the list visualizes a colored bar on all the nodes that share that vulnerability (see Figure 3).

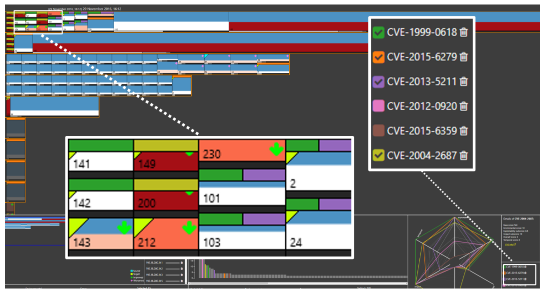


Fig. 3: Detail of the vulnerabilities spread on the network. The security operator has selected six vulnerabilities from the frequency distribution and then he or she has decided to lookup the nodes affected by four of them.

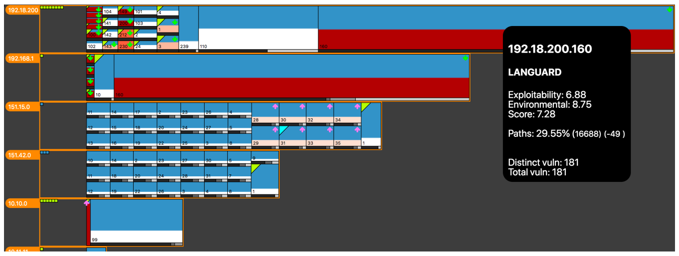
This mechanism allows for easily comparing multiple vulnerabilities by their representations in the radar-chart. If the number of vulnerabilities in the selection is very high the user might be interested only in those that respect certain metrics constraints and a filtering mechanism is provided (Figure 1.*F*). The metrics are divided in five intervals and their distribution is shown on box-plots; clicking on these elements permits filtering out the vulnerabilities that do not have a score belonging to the selected intervals.

In order to accomplish different types of analysis, the security manager can switch environment (Figure 1.*c*) to obtain different representations of the network in the main pane (Figure 4). In the Sub-Network Environment, each bar represents a sub-network and the elements represent its nodes (see Figure 4.*a*). Alternatively, when the security manager is interested in nodes that have certain metrics characteristics (e.g., bounds) regardless of the network topology, he or she can change the grouping of nodes. For instance, Figure 4.*b* shows the Score Environment in which elements still represent nodes but they are grouped by intervals of the Base score. When a higher level and more compact representation of the network is needed (for example, to get an overview during the Monitoring activity), the Network Environment (see Figure 4.*c*) represents the whole network as a single bar in which the elements represent its sub-networks. The last environment is the Vulnerability Environment (see Figure 4.*d*) in which all the vulnerabilities of the network are represented as elements of a single bar. The size of an element is proportional to the number of nodes of the network that share that vulnerability and the color encodes the maximum Target Environmental score associated to that vulnerability for all the nodes in which it appears.

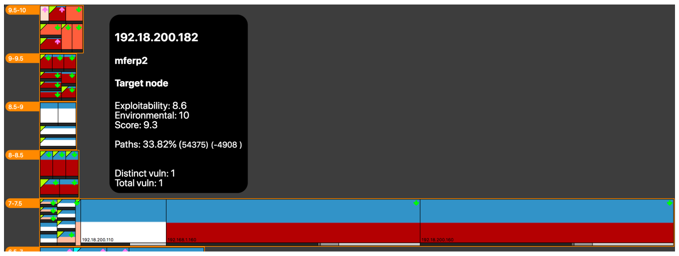
5.2.4 What-if analysis

The last supported activity is the identification of the fixing strategy (Section 4, requirement R3). The system computes the *AOF* strategy using the analytics presented in (5.1) that is proposed in the what-if pane (Figure 1.*J*) as a list of couples node-vulnerability fixes that are presented in a descending order of utility, i.e., from the one that removes the highest number of attack paths to the one that removes the smallest number of attack paths. However, it is very likely that this strategy may not be directly applicable because of internal or external constraints (e.g., business continuity) making a fix not applicable. To deal with this issue, the system implements a what-if analysis, simulating the application of a fix by clicking on the toggle next to the vulnerability instance to fix: the environment updates, showing the state that could be obtained by the application of the fix. The elements in the main pane are updated, indicators that compare the real state with the previous scan are removed and blinking green downward arrows are added to indicate the reduction of the Target Environmental of a node in the simulation (see the supplemental video).

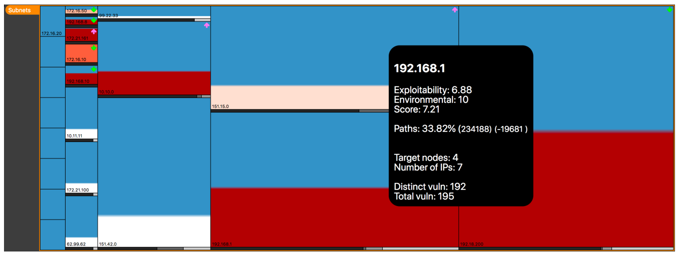
In this way the security manager can match the approximated optimal fixing strategy *AOF* (that is recomputed every time that a fix is simulated) against his or her knowledge regarding the applicability of the fixes, manually identifying a suboptimal *AOF*, and estimating its


192.18.200.160
LANGUARD
Exploitability: 6.88
Environmental: 8.75
Score: 7.28
Paths: 29.55% (16688) (-49)
Distinct vuln: 181
Total vuln: 181

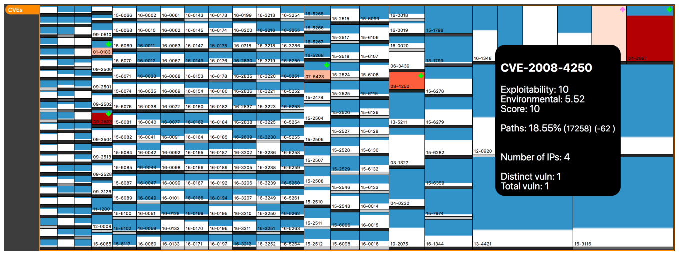
(a) Sub-network Environment


192.18.200.182
mferp2
Target node
Exploitability: 8.6
Environmental: 10
Score: 9.3
Paths: 33.82% (54375) (-4908)
Distinct vuln: 1
Total vuln: 1

(b) Score Environment


192.168.1
Exploitability: 6.88
Environmental: 10
Score: 7.21
Paths: 33.82% (234188) (-19681)
Target nodes: 4
Number of IPs: 7
Distinct vuln: 192
Total vuln: 195

(c) Network Environment


CVE-2008-4250
Exploitability: 10
Environmental: 5.52
Score: 10
Paths: 18.55% (17258) (-62)
Number of IPs: 4
Distinct vuln: 1
Total vuln: 1

(d) Vulnerability Environment

Fig. 4: a) The Sub-network Environment shows a bar for each sub-network where the elements represent its nodes; b) In the Score Environment, the elements still represent nodes but they are grouped by intervals of the base score; c) In the Network Environment, the network is represented as a single bar in which the elements represent the sub-networks; the elements in the leftmost column are sub-networks without vulnerabilities; d) In the Vulnerability Environment all the vulnerabilities of the network are shown as elements of a single bar and the size of each element is proportional to the number of nodes of the network that share that vulnerability.

effects. Figure 5 shows how much the optimal fixing strategy reduces the number of fixes needed to interrupt all the attack paths with respect to the strategy used before implementing VULNUS, that uses the Base score to prioritize vulnerabilities. Even assuming that *AOF* is not completely applicable, it allows the user to reason about a much smaller set of fixes and to focus just on those that drastically reduce the number of attack paths.

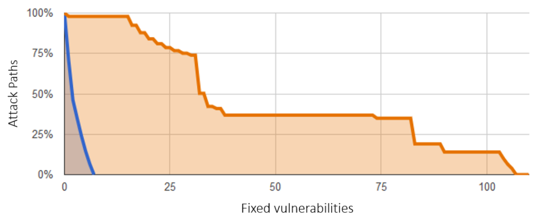
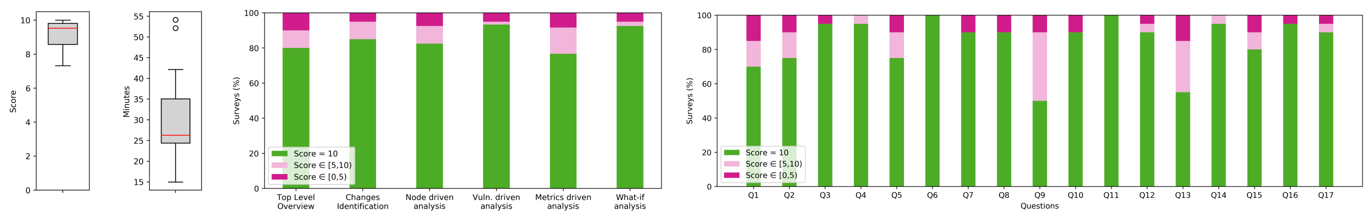


Fig. 5: Comparison of the attack paths reduction using the best fixing strategy (blue) computed by the proposed algorithm and the strategy (orange) in which fixes are ordered according to the CVSS base score (strategy used before implementing VULNUS), from the highest to the lowest. The best fixing strategy interrupts all the attack paths after having fixed 7 vulnerabilities while, with the other strategy, more than 100 vulnerabilities have to be fixed.

6 Evaluation

We conducted a quantitative evaluation in the lab in order to describe the current usability of the interface in terms of effectiveness (i.e., the accuracy and completeness with which the user achieves specific goals, see [25]) and efficiency (i.e., the effort in relation to effectiveness, see [25]). We might say that this is a summative evaluation in the usability vernacular since it is done towards the end of the project to validate a complete solution against the intended goals through precise metrics and quantitative measures. It comes after many formative evaluations carried on during the project development - mainly gathering qualitative data - in order to discover (and possibly remedy) usability problems. Some types of summative evaluations require the collection of baseline data to be compared with the evaluation results. In our case we did not have the possibility of collecting such data since no directly comparable solutions existed, as well as no benchmarks are available for VULNUS-like systems.

The involved subjects were 16 male and 4 female cyber security experts having an average experience in the field of more than 2 years (Education: 5 PhD, 7 Master, 5 Bachelor degree, 3 High school degree; Expertise: 1 Expert, 5 Highly knowledgable, 10 Knowledgable, 4 Some knowledge; Age: 8 in range 18-24, 10 in range 25-34, 2 in range 35-44; Position: 3 Bachelor students, 1 External collaborator, 6 Master students, 3 PhD students , 3 Post-Doc, 1 Project Manager, 1 Software Engineer, 2 Researchers). It is worth noting that we considered as experts also people with Bachelor and High school degrees because they have been involved in the National Cyber Challenge and, after one year of intensive training, they were very skilled in analyzing and patching vulnerabilities and familiar with attack graphs, CVE, and CVSS. Participants were asked to answer to 17 questions and their activities were traced; traces include question reading time, answering time, and actual interactions with the system, like tooltip visualizations, node selections, filters adjusting, etc. To correctly guide the user along the execution steps, i.e., reading questions, performing actions on the system, and reporting responses on a questionnaire, an evaluation environment has been designed and implemented [1]. The evaluation environment encapsulates the VULNUS system and is composed of two parts: a) the system to evaluate and b) the evaluation component, displayed on the left and the right side of the screen respectively. The evaluation component shows a question at a time, controls the state of the system, traces and recognizes the participants' actions, and automatically populates the response to the current question using the users' interaction with the system, reporting it on the screen for user inspection. From the point of view of the user's cognitive workload, this approach has the great advantage of letting her or him focusing on the system and preventing to overload or distract the subject with the need of transferring answers on a questionnaire, with the additional benefit of avoiding oversights while copying IP addresses or CVE IDs. All user actions and elapsed times for reading and answering are stored together with the answers, allowing a deeper and better evaluation of the system (this information was not provided to the users during the evaluation, in order to avoid to affect their behavior).



(a) Scores (b) Times (c) Responses per task (d) Responses per question

Fig. 6: The VULNUS system evaluation results at a glance. Each question has given a score in the range 0-10: an answer with a score of 10 is considered correct. a) The box-plot shows the distribution of the average scores of the 20 participants: median=9.53, mean=9.15, and standard deviation=0.80; only two questionnaires got a below 8 result. b) The box-plot depicts the average total completion time distribution (in minutes): median=26, mean=31, and standard deviation=11; there are two outliers above 50 minutes. Concerning the score distributions of tasks (c) and questions (d) we report the proportion of answers scores grouped in three intervals: $[0,5)$, $[5,10)$ and 10, because the mode of 10 score is so high that box-plots are visually useless. c) The bar-chart shows the proportion of correct answers aggregated per task, with an overall percentage of correct answers of 84% and a local minimum of of 77% in Metrics driven analysis. d) The bar-chart illustrates the proportion of correct answers for each question; we have nine questions with a correct percentage above 90% and the correct percentage of all questions is above 70% but Q9 and Q13 whose correct percentages are 50% and 55%, respectively.

6.1 Evaluation sub-tasks

Participants were asked to perform 17 sub-tasks, grouped according the requirements presented in Section 4. A large part of the sub-tasks is about analysis, covering most of the interactions the user will need as atomic bricks to perform complex activities (e.g., inspecting single nodes, sub-networks, and vulnerabilities, selecting subset of elements according to different criteria, and filtering large set of vulnerabilities using their properties).

Monitoring: R1a - Top level overview. These sub-tasks validate the system capability of providing an overview of the whole network and subnetworks.

- **Q1** Point out the sub-network with the highest Environmental and the lowest number of vulnerabilities (total).
- **Q2** Point out the two sub-networks with the highest Exploitability value.
- **Q3** List the five most frequent vulnerabilities in the whole network.

Monitoring: R1b - Changes identification. These sub-tasks validate the capability of identifying changes, providing details about them.

- **Q4** How many nodes have been changed after the previous network scan?
- **Q5** Considering only the worsened nodes, select the three most dangerous ones.

Analysis: R2a - Node driven analysis. These sub-tasks validate the capability of identifying network nodes based on their characteristics.

- **Q6** List the four nodes of the whole network with the highest number of vulnerabilities.
- **Q7** List all the source nodes with at least one vulnerability.

Analysis: R2b - Vulnerability driven analysis. These sub-tasks validate the capability of identifying critical vulnerabilities.

- **Q8** Among the five most frequent vulnerabilities of sub-network 192.18.200 point out the one with the highest Environmental.
- **Q9** Among all the vulnerabilities of sub-network 151.15.0 list the four most frequent vulnerabilities that have a Score above the Median.
- **Q10** Point out the most frequent vulnerability of the whole network that has Score and Exploitability in the interval 8-10.
- **Q11** Point out the vulnerability with the highest Target Environmental among all the nodes with Exploitability>=8
- **Q12** List the two vulnerabilities with the highest number of IPs that are easily exploitable.

Analysis: R2c - Metrics driven analysis. These sub-tasks validate the capability of inspecting network nodes using the scores of their vulnerabilities.

- **Q13** Point out how many nodes have the Score (strictly) above 7.5.
- **Q14** List the five nodes with the highest Score.
- **Q15** List all the vulnerabilities that belong to nodes with a Score in the range 4-6.

R3 - What if analysis These sub-tasks validate the capability of providing evidence of network changes after the simulation of one vulnerability fix.

- **Q16** Considering the proposed best fixing strategy, what is the minimum number of fixes that is needed to reduce the overall attack paths to less than 50%?
- **Q17** How many fixes are needed to reduce the overall attack paths to less than 50%, without the possibility to fix CVE 2004-2687 on node 192.168.8.100?

6.2 Methodology

As first activity a simplified pilot test was conducted involving very expert security operators, i.e. members of the relevant population since the system is intended for expert users. This test produced a rough estimation of 25 minutes as an adequate time for completing the sub-tasks also not evidentiating any particular problem and/or adverse factor in the study design. Before starting the experiment, people were instructed through a PowerPoint presentation (lasting 30-40 minutes) and videos, introducing the theoretical background and showing a practical use of the system on a training example. After that, participants spent about 20 minutes on the VULNUS system using the training example, in order to get familiar with the system structure and commands. Then, they were presented with the evaluation environment - that relies on a different network - and they were first asked to answer some general questions about themselves, then to answer the 17 questions. At the end participants were asked to give a written feedback on their experience in using the system in terms of encountered difficulties and gained benefits, as well as to comment on the overall interaction. Concerning the questions scores, in order to take into account the wrongness of an answer, we used Normalized Discounted Cumulative Gain (NDCG) [26], which is a standard measure of ranking quality. Each question has given a score in the range $[0,10]$, considering correct an answer with a score equal to 10, and incorrect otherwise.

6.3 Results

Collected results are shown on Figure 6. Median and average of total scores are quite high and the percentage of correct responses on all sub-tasks is 84.4%; for each sub-task the percentage of correct answers is above 80%, but Metrics driven analysis (77%); the total percentage of sub-tasks with a score greater than or equal to 5 is 92.1%. That provides a way to estimate the VULNUS effectiveness. However, question Q9 got the highest percentage of wrong answers, and this issue is discussed in the next section. Considering efficiency, we cannot make formal claims because we do not have a "ground truth" about the "right" completion time of the sub-tasks. However, we can notice that 31 minutes mean and 26 minutes median are in line with the expected results coming from the pilot test; nevertheless, the large standard deviation and the presence of two outliers above 50 minutes pushed us to better investigate the response time issue, as discussed in the next section. Moreover, the questions answered after sub-tasks completion provide the perceived usefulness and simplicity of use of the four environments, and their informative capability with respect to the three main activities. Table 1 presents user votes about environments; as an example, the first row tells us that most of the users (16) think that the most useful environment is the Sub-network Environment that, in the second row, is considered by 12 users as the easiest one to use. Table 2, instead, presents the averages of the perceived informativeness scores (from 0=not informative at all, to 10=completely informative) for the three activities, showing which is perceived as the best supported one.

Table 1: User preferences about environments

	Network	Sub-network	Score	Vulnerability
Most useful	2	16	1	1
Easiest to use	4	12	2	2
Most difficult	1	3	6	10

Table 2: System informativeness with respect to the 3 main activities

Activity	Average informativeness score
Monitoring	8.05
Analysis	8.85
What if	8.7

6.4 Discussion

We conducted an explorative analysis of experiment results based on user traces, completion time, and quality of result, looking for common patterns among answers to the same question. However, it is out of the scope of this paper to provide a full characterization of this analysis, and we use only some of the collected evidences to comment questions Q9 and Q13 that have the lowest answers correct percentages. Inspecting the traces of question Q9, we discovered that all the correct answers eventually followed the same predominant solution pattern (i.e., *group-selection*, *vulnerability-filter*, *vulnerability-selections*), and an half of them with the same traces. Regarding the wrong answers, we noted that participants made errors principally on filter selections that led to a high number of following actions with the result of going far from correctly answer the question. Considering Q13, the right answer was 24 and the best solution strategy was to select clusters of nodes using CVSS score ranges; several participants, instead, selected nodes one by one missing one or two nodes; other participants included the lowest score in the search, ignoring the word "strictly". Moreover, the analysis of traces patterns revealed that the 20 solution strategies are quite different. We can conclude that using box-plots as a selection means and grouping nodes by CVSS scores are a possible source of errors. Furthermore, the score performance of Q1, a relative simple question, that had the highest number of incorrect answers pushed us to speculate on some learning activity of the VULNUS system during the first questions of the experiment. It is worth to recall that participants were totally unaware of the VULNUS system, which is the result of a three year user-centered design; getting familiar with it is not a simple task, and likely 20 minutes of system usage were not always enough.

Concerning the total response time we investigated the two outliers above 50 minutes, discovering that both participants were very slow on the first question and on the first change of environment (about 10 minutes per question against an average of 2 minutes). After that, their speed was in line with other results. Our hypothesis is that these two participants did not digest the basic system interactions and that the extra time they used was mainly for learning purposes. Another evidence of this behavior is that their traces exhibit an anomalous number of tooltip visualizations at the first question (about 5 times more than the average): likely they were just browsing the nodes. That confirms the above hypothesis of a learning period during the first question Q1. Considering the 6 tasks (Figure 6.*c*), we can conclude that the lower performance of Metrics driven analysis is due to the problems associated with Q13, and that, taking into account this issue, the situation among tasks is more homogeneous. As a last consideration, we report some comments from participants, asking for more complex direct manipulation interactions like "...the option to select elements by dragging an area around them in the view", for the encoding of more data in the treemap like visualization "would reconsider the main pane in relation to a much aggressive visual approach", for a less packed interface "The interface is quite clean and clear, but I guess it could be better by increasing the spaces between different interfaces/panels", for including operational information in the system "I would like to have the estimated fixing time", or providing enthusiastic feedback "Congratulation for the tool!".

7 Conclusion

The paper presented the VULNUS solution used to analyze the vulnerabilities of computer networks, providing both an overview and details of large set of vulnerabilities spread across network nodes. The actual solution has been designed together with 6 ACEA security people, along a 3 year user-centered design activity. A distinguishing feature of the system is its strong connection with the CVSS and attack graph data, providing network analysts with details about vulnerability characteristics and assessing their dangerousness. In particular, VULNUS proposes a modification of the CVSS environmental score that allows for automatically computing the approximated optimal fixing strategy that dramatically reduces the number of interventions on the network. A novel what-if scenario permits the exploration of the effect of fixing activities without applying them to the real system, helping the user in the situations in which applying the approximated optimal fixing strategy is not viable for organization constraints.

The developing of the system along a three year project, beside confirming the usefulness and efficacy of user centered design development, taught us several key considerations. The first one was about the usage of standard visualizations, that typically represent the first design solution to user requests: sometimes they do not fully satisfy the user requirements and it is a good idea to be ready to go for somehow unconventional but more effective solutions. As an example, we moved from a standard treemap bar chart with a 3 levels hierarchy (network-node-vulnerability) to a treemap bar chart with just one hierarchical level (so it is not a treemap anymore, it is just a bar filled with elements whose area is proportional to the number of vulnerabilities) to avoid browsing issues. Later on, users asked to explicitly represent nodes with zero vulnerabilities (area equal to zero) and we split the bar in two: one part, of fixed size, hosting zero vulnerabilities nodes with a predefined area and one part, of variable size, hosting the nodes with one or more vulnerabilities, ordering them by area from left to right. The whole process lasted about 1.5 years. The second lesson learned is that automated solutions, in complex network scenarios, even if optimal may be not applicable for problems external to our analytical environment (e.g., business constraints) and that a simulation scenario for exploring alternative strategies can be useful even if the best solution is known. The last insight we got while developing the system is about filtering. Users requested us a complex interaction strategy, using direct manipulation on nodes and on frequency distribution, multiple uniform bar-grams and box plots, using mixed logical operators (OR, AND). This produced the drawback that users experienced difficulties in modifying the current selection because it is not easy to relate it to filtering actions, and we are currently investigating on how to present the user with additional feedback to mitigate such an issue.

Concerning the VULNUS implementation we are addressing some minor usability problems arisen during the user study and we are extending the environment in order to include information about the organization, mission, target nodes criticality, fixing costs (time, manpower), etc. Regarding research activities we plan to proceed along three main directions: a) Performing a full analysis of traces: an on-line version of the user study, collecting additional user traces is now available (`http://151.100.59.83:11768/vulnus/`); b) Exploring new patching strategies (e.g., [14]); c) Challenging the generality of the proposed visual encoding, testing it on multivariate data coming from different domains (e.g., healthcare or financial data).

References

[1] M. Angelini, G. Blasilli, S. Lenti, and G. Santucci. STEIN: Speeding up Evaluation Activities With a Seamless Testing Environment INtegrator. In J. Johansson, F. Sadlo, and T. Schreck, eds., *EuroVis 2018 - Short Papers*. The Eurographics Association, 2018. doi: 10.2312/eurovisshort.20181083

[2] M. Angelini, N. Prigent, and G. Santucci. Percival: proactive and reactive attack and response assessment for cyber incidents using visual analytics. In *Visualization for Cyber Security (VizSec), 2015 IEEE Symposium on*, pp. 1–8. IEEE, 2015.

[3] P. Barford, M. Dacier, T. G. Dietterich, M. Fredrikson, J. Giffin, S. Jajodia, S. Jha, J. Li, P. Liu, P. Ning, et al. Cyber sa: Situational awareness for cyber defense. In *Cyber Situational Awareness*, pp. 3–13. Springer, 2010.

[4] R. Bearavolu, K. Lakkaraju, W. Yurcik, and H. Raje. A visualization tool for situational awareness of tactical and strategic security events on large and complex computer networks. In *Military Communications Conference, 2003. MILCOM '03. 2003 IEEE*, vol. 2, pp. 850–855 Vol.2, Oct 2003. doi: 10.1109/MILCOM.2003.1290234

[5] R. Blanch and E. Lecolinet. Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1248–1253, Nov 2007. doi: 10.1109/TVCG.2007.70540

[6] D. Borbor, L. Wang, S. Jajodia, and A. Singhal. Securing networks against unpatchable and unknown vulnerabilities using heterogeneous hardening options. In G. Livraga and S. Zhu, eds., *Data and Applications Security and Privacy XXXI*, pp. 509–528. Springer International Publishing, Cham, 2017.

[7] M. Bruls, K. Huizing, and J. J. Van Wijk. Squarified treemaps. In *VisSym*, pp. 33–42. Springer, 2000.

[8] Checkpoint Software solutions ltd. Ckeckpoint Next Generation SmartEvents. `https://www.checkpoint.com/products/smartevent/`.

[9] V. Y. Chen, A. M. Razip, S. Ko, C. Z. Qian, and D. S. Ebert. Multi-aspect visual analytics on large-scale high-dimensional cyber security data. *Information Visualization*, 14(1):62–75, 2015.

[10] P. Cheng, L. Wang, S. Jajodia, and A. Singhal. Refining cvss-based network security metrics by examining the base scores. In *Network Security Metrics*, pp. 25–52. Springer, 2017.

[11] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer. Visualizing attack graphs, reachability, and trust relationships with navigator. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, pp. 22–33. ACM, 2010.

[12] G. Conti. *Security data visualization: graphical techniques for network analysis*. No Starch Press, 2007.

[13] O. Developers. The open vulnerability assessment system (openvas), 2012.

[14] K. A. Farris, A. Shah, G. Cybenko, R. Ganesan, and S. Jajodia. Vulcon: A system for vulnerability prioritization, mitigation, and management. *ACM Trans. Priv. Secur.*, 21(4):16:1–16:28, June 2018. doi: 10.1145/3196884

[15] F. Fischer, J. Davey, J. Fuchs, O. Thonnard, J. Kohlhammer, and D. A. Keim. A visual analytics field experiment to evaluate alternative visualizations for cyber security applications. In *Proc. of the EuroVA International Workshop on Visual Analytics*, 2014.

[16] F. Fischer and D. Keim. Vacs: Visual analytics suite for cyber security-visual exploration of cyber security datasets. In *IEEE VIS*, 2013.

[17] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS-SIG). `https://www.first.org/cvss/`.

[18] M. Frigault, L. Wang, S. Jajodia, and A. Singhal. Measuring the overall network security by combining cvss scores based on attack graphs and bayesian networks. In *Network Security Metrics*, pp. 1–23. Springer, 2017.

[19] J. R. Goodall, H. Radwan, and L. Halseth. Visual analysis of code security. In *Proceedings of the seventh international symposium on visualization for cyber security*, pp. 46–51. ACM, 2010.

[20] J. R. Goodall and M. Sowul. Viassist: Visual analytics for cyber defense. In *Technologies for Homeland Security, 2009. HST'09. IEEE Conference on*, pp. 143–150. IEEE, 2009.

[21] V. T. Guimaraes, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, and L. Z. Granville. A survey on information visualization for network and service management. *IEEE Communications Surveys Tutorials*, 18(1):285–323, Firstquarter 2016. doi: 10.1109/COMST.2015.2450538

[22] L. Harrison, R. Spahn, M. Iannacone, E. Downing, and J. R. Goodall. Nv: Nessus vulnerability visualization for the web. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security*, pp. 25–32. ACM, 2012.

[23] IBM. IBM Qradar. `http://www-03.ibm.com/software/products/en/qradar`.

[24] Imperva. Imperva SecureSphere. `http://www.imperva.com/Products/WebApplicationFirewall`.

[25] ISO. ISO 25010 standard. `http://iso25000.com/index.php/en/iso-25000-standards/iso-25010/`.

[26] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002. doi: 10.1145/582415.582418

[27] R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs. Prepared for the Department of the Air Force under Contract F19628-00-C-0002, Mar. 2005.

[28] A. Makanju, S. Brooks, A. N. Zincir-Heywood, and E. E. Milios. Logview: Visualizing event log clusters. In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*, pp. 99–108. IEEE, 2008.

[29] F. Mansmann, F. Fischer, D. A. Keim, and S. C. North. Visual support for analyzing network traffic and intrusion detection events using treemap and graph representations. In *Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology*, p. 3. ACM, 2009.

[30] F. Mansmann, D. A. Keim, S. C. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, 2007.

[31] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pp. 109–118. ACM, New York, NY, USA, 2004. doi: 10.1145/1029208.1029225

[32] C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 workshop on New security paradigms*, pp. 71–79. ACM, 1998.

[33] RSA enVision. RSA enVision. `http://www.emc.com/support/rsa/eops/siem.htm`.

[34] V. Shandilya, C. B. Simmons, and S. Shiva. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014, Oct. 2014.

[35] H. Shiravi, A. Shiravi, A. Ghorbani, et al. A survey of visualization systems for network security. *Visualization and Computer Graphics, IEEE Transactions on*, 18(8):1313–1329, 2012.

[36] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. In *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pp. 73–78. IEEE, 2001.

[37] R. Skowyra, S. R. Gomez, D. Bigelow, J. Landry, and H. Okhravi. Quasar: Quantitative attack space analysis and reasoning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 68–78. ACM, 2017.

[38] G. Software. Languard network security scanner. `http://www.gfi.com/`.

[39] Tableau. Treemap Bar Charts. `https://public.tableau.com/en-us/s/blog/2013/01/new-8-treemap-bar-charts`.

[40] Tenatable. Nessus Professional Vulnerability Scanner. `https://www.tenable.com/products/nessus/nessus-professional`.

[41] The MITRE Corporation. Common Vulnerabilities and Exposures (CVE). `https://cve.mitre.org/`.

[42] The PANOPTESEC Consortium. *The Official Website of the PANOPTESEC Project, http://www.panoptesec.eu/*, 2014.

[43] VisiTrend. VisiTrend. `http://visitrend.tumblr.com/`.

[44] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, and W. Aigner. A survey of visualization systems for malware analysis. In *EG Conference on Visualization (EuroVis)-STARs*, pp. 105–125, 2015.

[45] C. Ware. *Visual Thinking: For Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[46] L. Williams, R. Lippmann, and K. Ingols. An interactive attack graph cascade and reachability display. In *VizSEC 2007*, pp. 221–236. Springer, 2008.