# Analysis of the NIST database towards the composition of vulnerabilities in attack scenarios

Virginia N. L. Franqueira and Maurice van Keulen

University of Twente
Enschede, The Netherlands
{franqueirav,m.vankeulen}@ewi.utwente.nl

**Abstract.** The composition of vulnerabilities in attack scenarios has been traditionally performed based on detailed pre- and post-conditions. Although very precise, this approach is dependent on human analysis, is time consuming, and not at all scalable. We investigate the NIST National Vulnerability Database (NVD) with three goals: (i) understand the associations among vulnerability attributes related to impact, exploitability, privilege, type of vulnerability and clues derived from plaintext descriptions, (ii) validate our initial composition model which is based on required access and resulting effect, and (iii) investigate the maturity of XML database technology for performing statistical analyses like this *directly* on the XML data. In this report, we analyse 27,273 vulnerability entries (CVE [1]) from the NVD. Using only nominal information, we are able to e.g. identify clusters in the class of vulnerabilities with no privilege which represent 52% of the entries.

**Keywords:** network vulnerabilities, attack scenarios, CVE, CVSS, XQuery, XML database

## 1 Introduction

The combination of the elements vulnerability, reachability, and threat gives rise to the risk of an attacker exploiting attack scenarios to reach targets, such as valuable assets of an organization.

Vulnerabilities [1] are uncovered every day. This happens in part because COTS (Commercial Off-The-Shelf) software components are not bug-free, and worse, not even COTS producers know how vulnerable their products are [2]. Hence, vulnerabilities are discovered and corrected (i.e. patched) along the COTS lifecycle. Apart from that, hackers have access to more and more resources such as sophisticated tools (often available on the Internet) and hardware. As a consequence, the chance of vulnerabilities occurring in a network and of attackers finding a way to exploit them increases.

[1] vulnerability [1] is "a mistake in software that can be directly used by a hacker to gain access to a system or network".

Reachability became a demand, and organisations nowadays face the challenge of providing employees with access "all the time, from everywhere". Thus, organisations' networks can never be isolated from the outside World. Additionally, it is common-sense that "a computer network is only as secure as its weakest node" [3, Page 22]. Therefore, although valuable hosts are usually protected by sometimes several filtering barriers, reaching the less protected may allow attackers to reach the more secure and valuable hosts.

Threat arises from a wide spectrum of attackers' motivation: profit, fame and sabotage [4,5]. Considering this diversity, it seems reasonable to assume that it is likely that any organisation will be targeted by at least one class of attackers. As a consequence, threat becomes a reality and can only be avoided by means of protection.

In this report, we focus on dealing with the compositional aspect of vulnerabilities. As such, we assume that vulnerabilities have been detected by vulnerability scanning tools, like Nessus [6], which report vulnerabilities found in hosts in terms of the standard nomenclature CVE (further information in Section 3). We assume the worst-case scenario where threat is constantly present, thus, vulnerabilities and reachability among hosts are essential ingredients to be tracked in networks.

We investigate the NVD [7], maintained by the NIST [8], a repository of known vulnerabilities used by security practitioners worldwide. Our aim is to understand the database having in mind the following requirements for the vulnerability composition model:

- It should be based on generic rules,
- it should be viable to automation,
- it should be scalable,
- it should allow a binary decision for the ability of an attacker to *move* from a compromised host to exploit a vulnerability in another reachable host, and
- it should be relevant in practice in terms of coverage of its elements on the NVD.

### 1.1 Contribution

The contribution of this report is three-fold. First, we contribute by increasing knowledge about the NVD, which is one of the most comprehensive (publicly available) databases of vulnerabilities used in practice and reported by academia (e.g. [9,10,11]). Second, we contribute by validating our composition model, which is based on other researchers' work [10,12], in terms of coverage of the model in practice, i.e., our model against the NVD. Finally, as a side contribution, we demonstrate that XML database technology is mature enough to interactively analyse XML data like the NVD database using XQuery [13] directly on an XML database, the MonetDB/XQuery system [14] in particular.

### 1.2 Report organisation

After discussing related work in Section 2, we first provide in Section 3 a brief introduction about three NIST initiatives relevant for the remainder of the report.

The report is divided into two parts and is organized very much as a design problem guided by the requirements given in the previous section. Part I deals with the composition of vulnerabilities. We present an initial model for vulnerabilities in Section 4, i.e. we propose a set of criteria for classifying vulnerabilities in large scale. In Section 5, we define rules for compromise i.e. for escalation of privilege and for *moving* from one host to another. Part II deals with the processing and analysis of the data source, i.e. the NIST/NVD, mainly focusing on finding associations between the database attributes and on the distribution of CVEs among classes. This stage is reported in Sections 6 and 7. Section 7.4 summarizes what has been uncovered from the analysis. Finally, in Section 8, we reflect the findings from the database analysis over the composition model, presented in Section 5, and draw overall conclusions in Section 9. Along Sections 4 to 7 we highlight some hypotheses to be validated with the NVD and draw some conclusions based on evidence found.

## 2 Related work

As detailed in Section 1.2, this report can be split in two parts, and consequently its related work can also be split in two: one refers to the composition of vulnerabilities and is related to the research areas of Attack Graph and IDS (Intrusion Detection System) alert correlation. The other refers to knowledge discovery in vulnerability databases.

Not many commercial tools provide some sort of correlation among network vulnerabilities. Skybox and Amenaza provide such tools although it is not clear which rules for vulnerability composition they use [15,16]. In the literature, the main approach used to compose vulnerabilities and alerts in attack scenarios relies on detailed sets of pre- and post-conditions. These conditions are analyzed manually and usually involve details about network and host configurations as well as attackers' resources and capabilities which must hold as pre-conditions, and resulting network and attacker states after an attack step has happened as post-conditions. This approach is labor intensive and, therefore, often not scalable [16]. The next three examples from the Attack Graph community illustrate this approach.

Sheyner and Wing [17] model atomic attacks using intruders' pre-conditions such as his level of privilege in both source and target hosts, and network preconditions such as specific services running on the target host and reachability from source to target hosts through specific TCP/UDP ports. Intruders' and network effects refer to privilege acquired by the intruder on the target machine and changes in services running, for example. They compose attack scenarios using a symbolic model checker. Each state transition corresponds to an atomic

attack where pre-conditions in the source state are satisfied and post-conditions in the target state hold.

Phillips and Swiler [18] model attack templates based on information related to nodes, representing states of an attack, and edges, representing actions, events and/or conditions that must hold for the attack to happen. The initial state of the network is retrieved from a detailed configuration file. The initial capabilities of the attacker are obtained from an attacker profile which can distinguish e.g. novice from expert attackers, used to calculate attack probability, and can enumerate attackers' capabilities in terms of e.g. possession of automated toolkit or sniffer. Nodes require several sets of information such as user level, machine (variable which can be instantiated by one or a group of machines), vulnerabilities, capabilities (these last two can overwrite the configuration file and the attacker profile), and state which indicates stepwise progress of the attack. Attack scenarios are composed when edge conditions are satisfied permitting the transition from one attack state to the next.

Ou et al. [9] model exploit rules using a logic approach based on Datalog, a subset of Prolog. These rules represent attack steps, e.g. [9, Page 5] "fullControl(P,H,UserPriv) indicates that principal P can execute arbitrary code with privilege UserPriv on machine H", which are specified in the form of if-then rules. These rules can be included as the if-part of other rules, composing then attack scenarios to be evaluated by the logical engine. They reported to have 80 exploit rules in their MulVAL framework, in addition to rules which specify host and network configuration, principal and policies. In more recent work [19], they apply their approach to a logical attack graph which uses derived nodes, derived facts and primitive facts.

Several other frameworks follow the detailed pre- and post-conditions approach to compose attack scenarios [17,18,9,20,21,22,23,24,25,26,27,28]. For yet further works, refer to Lippmann and Ingols's review [16].

An alternative approach to compose attack steps relies on simplified pre- and post-conditions. Lippmann et al. [10,12] use a straightforward way to model attacker actions and vulnerabilities. The latter are classified based on a set of two localities (remote and local) and a set of four effects (administrator, user, other, DoS[2]). The former are determined by the vulnerability model and reachability between hosts. Their vulnerability database is populated by information retrieved from Nessus scanning tool [6] and from publicly available databases [7]. Our approach for the composition of vulnerabilities in attack scenarios is derived from theirs. We study the NVD [7] to gain insight, check some hypotheses, and validate our model. E.g. Lippmann et al. [12, Page 11] mention that "the administrator privilege category can be indicated by phrases including *execute arbitrary code*"; this was not confirmed by our analysis of the NVD. Additionally, they consider in their models an effect "other"[10, Page 5] interpreted as related to confidentiality and integrity loss. The NVD also contains a privilege "other" and our analysis revealed the impact configurations related to it.

[2] Denial of Services

Knowledge discovery in vulnerability databases has been approached mainly for reasoning about timing, such as the following three examples. Arora et al. [29] empirically analysed the impact on the tendency of: (i) vendors in releasing patches, and (ii) attackers in exploiting vulnerabilities, based on data from an older version of the NVD (called ICAT Metabase) and other databases like the Bugtraq [30]. Frei et al. [11] examined the NVD and the osvdb [31] with the aim of finding trends and quantifying the performance of security (in global terms) by means of the gap between vulnerability discovery and patch release. Beattie et al. [32] used the NVD database to gather data about patch release and subsequent revisions due to faulty patches to reason about the practical problem: "when to patch?". Apart from timing, however, only a few researchers studied associations between attributes from vulnerability databases. Tierney [33] used ICAT and detected e.g. that "low severity items [vulnerabilities] are associated with loss of confidentiality". He also applied the mined rules to classify new example vulnerabilities. Although relevant, he did not invest in understanding of interesting attributes like privilege. Schumacher et al. [34] did not provide results which could increase understanding of vulnerability databases. Lippmann[10] used the NVD and vulnerabilities reported by the Nessus scanning tool[6] to manually classify locality and effect of a training set which was later used to classify vulnerabilities reported by Nessus.

## 3 NIST initiatives: NVD, CVE and CVSS

The NIST [8], sponsored by the U.S. Department of Homeland Security, has launched several initiatives towards interoperability of information related to computer security vulnerabilities. Three of such initiatives are the NVD, CVE and CVSS.

The NVD [7] is a comprehensive repository of vulnerabilities freely available to the public. It is in its second version and currently in XML format. It collects vulnerabilities in COTS (Commercial Off-the-Shelf) software since 1999.

CVE [1] (Common Vulnerabilities and Exposures [3]) is a naming scheme assigned to known vulnerabilities identified by the community. It is managed by Mitre (www.mitre.org) and is adopted by many CVE-compatible security products and services avoiding the problem of several identifiers for the same vulnerability. CVE identifiers (CVEs for short) are assigned by a CVE Editorial Board and have the format: CVE-yyyy-xxxx, where yyyy is the year when the vulnerability was discovered and xxxx is a sequential number.

The CVSS (Common Vulnerability Scoring System) [4] is an effort to provide an universal vendor-independent score of known vulnerabilities. The CVSS scoring system is in its second version since launched in 2004 and has already been

[3] exposure [1] is "a system configuration issue or a mistake in software that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or network".

[4] The CVSS is maintained by FIRST (Forum of Incident Response and Security Teams - www.first.org).

adopted by generic security bulletins such as Bugtraq [30] and osvdb [31], by vendor-specific bulletins such as from IBM, HP and Cisco, by scanning tools (see e.g. [15] for a list of tools which report in terms of CVE and CVSS) , and by intrusion detection and prevention tools.

The CVSS score, a decimal number on a scale 0.0-10.0, is composed of three metric groups: base, temporal, and environmental. The "base metric group" quantifies the intrinsic characteristics of a vulnerability in terms of two subscores: (i) *exploitability_subscore*, composed of access vector "AV" (type of access required in terms of Local, Adjacent network or Network), access complexity "AC" (level of specialisation required for access in terms of High, Medium or Low), and authentication "Au" (number of authentication instances required once the target has been accessed in terms of None, Single or Multiple), and (ii) potential *impact_subscore* to confidentiality (C), integrity (I) and availability (A) which the exploitation of the vulnerability can cause in terms of None, Partial or Complete.

Experts (from NIST) analyze each CVE and assign qualitative values for each of those attributes. Then, the CVSS system calculates the subscores. The configuration "Network" for access, "Low" for complexity and "None" for authentication returns the highest exploitability subscore, i.e. 10.0. Similarly, "Complete" impact for Confidentiality, Integrity, and Availability returns the highest impact subscore, also 10.0.

Each CVE reports: (i) the base score, (ii) the exploitability and impact subscores and (iii) the base vector from which the base score has been derived. For example, CVE-1999-0196 has base vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N), Exploitability subscore: 10.0 and Impact subscore: 2.9. The other metric groups, i.e. temporal and environmental, are context-dependent and therefore are not included in the database. Refer to the CVSS Complete Guide [35] for the base equation and for details about the transformation of attributes from qualitative to quantitative.

In the next section, we detail our initial vulnerability and composition models.

## 4 Vulnerability model

Our vulnerability model comprises two criteria and is based on the vulnerability classification provided by Lippmann et al. [10,12]. Similar to them, we represent vulnerabilities [5] in terms of "access-to-effect". Our attacker model is based on the worst-case scenario: attackers know about the existence of all CVEs present in the network and will exploit them successfully, benefiting from their effect, if access is available.

- **access**: access required to exploit a CVE
  - *network* means a CVE in host $h1$ can be exploited from another reachable host $h2$

[5] terms "vulnerability" and "CVE" are used interchangeably

  - *local* means a CVE in host $h1$ can be exploited only from host $h1$
- **effect**: effect resulting from exploiting a CVE
  - *user* means an attacker gains user access to the vulnerable host
  - *admin* means an attacker gains privileged access (e.g. root in Unix-based hosts and administrator in Windows-based hosts) to the vulnerable host
  - *runCode* means an attacker gains the ability to execute arbitrary code on the vulnerable host
  - *obtainCred* means an attacker gains the ability to obtain credentials for the vulnerable host
  - *DoS* means an attacker gains the ability to make the vulnerable host unavailable for its legitimate users

Apart from access, *credentials* may also be required to exploit a CVE. If credentials are required to fully exploit a CVE, it means that this vulnerability should be preceded by a CVE of the type {network,local}-to-obtainCred [6].

**Hyphothesis 1**: The set of CVEs which allow obtaining credentials is significant (i.e. at least 5% of CVEs) and can be detected automatically using clues from their descriptions.

Other prerequisites such as software installed, and operating system running on a host with respective versions are not part of the model, since we assume they have already been taken into account on the vulnerability scanning process, by e.g. Nessus [6] tool.

In the next section, we describe our composition model for attack scenarios.

## 5 Composition model

We consider that a random attacker can build attack scenarios by compromising one host after another, until a target is reached. Therefore, an attacker can *move* from a host $h1$ to a host $h2$ if (i) $h2$ is reachable from $h1$, (ii) $h1$ is compromised, according to the rules of compromise enumerated in Section 5.1, and (iii) there is at least one CVE of the following type in $h2$: network-to-{user,admin,DoS,runCode,obtainCred}.

### 5.1 Rules of compromise

A host is considered compromised if an attacker reached "admin" privilege in a host. An attacker $A$ can comprise a host through one of the following rules of compromise.

Rule 1 $A$ exploited a network-to-{admin,runCode,DoS} CVE.

Rule 2 $A$ exploited a network-to-user CVE, then a local-to-{admin,runCode,DoS} CVE which does not require credentials.

Rule 3 $A$ exploited a network-to-user, then a local-to-obtainCred followed by a local-to-admin CVE which requires credentials.

[6] Curly brackets indicate alternatives.

Rule 4 *A* exploited a network-to-obtainCred followed by a local-to-admin CVE which requires credentials.

**Hyphothesis 2**: Exploiting a runCode or DoS CVE implies on "admin" level of privilege in at least 90% of the cases.

## 6 Processing the NVD

The NVD database[7] contains 27,909 CVE entries scattered among XML files corresponding to vulnerabilities created between 1999 and 2007. We skipped the entries from the database that were marked "** REJECT **" and those that were not yet analysed (407 entries had just a description). The final number of vulnerability entries used in this research is 27,273.

The XML files were loaded into the XML database. Because this research was also used as a case study for demonstrating the maturity of XML database technology for performing statistical analyses like this, *directly* on the XML data, we processed the XML data in two ways:

- **Main process**
  The data was converted to CSV (Comma-Separated Value) using XQuery. This conversion applied the transformations described in Table 1. For analysis we used Python scripts [36] and the Weka mining tool [37].
- **Case study process**
  To mimick the process, a similar transformation query was constructed to produce not CSV, but an XML document containing the same attributes, transformations and selections, which was then queried.

### 6.1 Data set attributes

Table 1 contains the list of attributes we have in the data set. Column "NVD source" describes the XML tags from the original NVD XML database, and columns "description" and "preprocessing" describe the content of the attributes and further transformations applied to them[8].

## 7 Analysing the NVD

We use the following methodology to extract knowledge and derive conclusions from the NVD.

- Analysis of single attributes using the complete CSV data set (see Section 7.1).
- Analysis of relations between attributes, driven by the composition model presented in Section 5, using the complete CSV data set (see Section 7.2).
- Analysis of specific aspects using reduced data sets (see Section 7.3).

[7] We downloaded the XML files on 08/11/2007.

[8] Except by the exploitability (cvss_exploit) and impact (cvss_impact) subscores, all attributes used were nominal.

| att. names | NVD source | description | preprocessing & content |
|---|---|---|---|
| name | entry → name | CVE identifier | none |
| cvss_AC | entry → CVSS_vector | access complexity | no processing; possible values: L (low), M (medium) or H (high) (refer to Section 3 for information about the CVSS vector, i.e. attributes and possible content.) |
| cvss_AV | entry → CVSS_vector | access vector | no processing; possible values: N (network), L (local) or A (adjacent network) |
| cvss_Au | entry → CVSS_vector | authentication required | no processing; possible values: N (none), S (single) or M (multiple) |
| cvss_C | entry → CVSS_vector | impact on confidentiality | no processing; possible values: N (none), P (partial) or C (complete) |
| cvss_I | entry → CVSS_vector | impact on integrity | no processing; same values as for cvss_C |
| cvss_A | entry → CVSS_vector | impact on availability | no processing; same values as for cvss_C |
| cvss_exploit | entry → → CVSS_exploit_subscore | exploitability subscore | no processing; possible values from 0.0 to 10.0 |
| cvss_impact | entry → → CVSS_impact_subscore | impact subscore | no processing; same values as for cvss_exploit |
| privilege | entry → loss_types → sec_prot | privilege acquired by exploiting the vulnerability | can contain a single privilege such as "admin", "user", "other", can be empty (i.e. have no privilege assigned), or can contain multiple privileges. In this last case, a unique privilege was derived as follows: "other,admin" replaced by "admin"; "other,user" replaced by "user"; "user,admin" replaced by "admin"; "other,user,admin" replaced by "admin" |
| type_vuln | entry → vuln_types | type of vulnerability | can contain single types such as "input", "access", "design", "config", "race", "env", "exception", "other", can be empty, or can contain multiple types; in this last case, we replaced the list of types by "multiple" |
| runCode | entry → desc → descript | possibility of running code | "y" if the description contains the following key expressions: 'execute arbitrary code' or 'execute arbitrary programs'; "no" otherwise |
| obtainCred | entry → desc → descript | possibility of obtaining credentials for authentication | "y" if the description contains the following key expressions: 'intercept transmission', 'intercept communication', 'obtain plaintext', 'obtain cleartext', 'read network traffic', 'unencrypted' or 'sniff'; "no" otherwise |
| gainAdmin | entry → desc → descript | possibility to gain admin level of access | "y" if the description contains the following key expressions: 'gain root' or 'gain access to root'; "no" otherwise |
| DoS | entry → desc → descript | possibility to cause a denial of service in the host | "y" if the description contains the following key expression: 'denial of service'; "no" otherwise |

**Fig. 1.** CSV data set attributes

```
import module namespace mstat = "..." at ".../modstat.xq";
let $cves := doc("nvdcve.xml")//cvss
return <result>{ mstat:value-distribution($cves,"cvss_AV")
                ,mstat:value-distribution($cves,"cvss_AC")
                ,mstat:value-distribution($cves,"cvss_Au")
                ,mstat:value-distribution($cves,"privilege")
       }</result>
```

**Fig. 2.** Example XQuery reproducing Table 4(a) directly from NVD XML data

Separately, the same evaluation of the data set was performed using XQuery queries directly on the data in the XML database. For each table in this report, a query was written to reproduce the numbers in the table (see Figure 2 for an example). We tried to generalize on the kinds of analyses done by constructing generic domain-independent functions. For example, the query in Figure 2 uses `mstat:value-distribution`, which calculates the distribution of values for a given child of a sequence of elements. We evaluate our experiences with using XML database technology in this way by looking at the sizes of the queries, whether the queries run in interactive time on an ordinary PC[9], and the mistakes made along the way.

Table 3 contains the impact configurations found among the data set along the analysis process. Refer to Table 1 for attribute names and possible contents. For example, cvss_C=C means complete impact on confidentiality.

| **CIA impact** | **NVD attributes configuration** |
|---|---|
| Complete CIA | cvss_C=C and cvss_I=C and cvss_A=C |
| Partial CIA | cvss_C=P and cvss_I=P and cvss_A=P |
| No CIA | cvss_C=N and cvss_I=N and cvss_A=N |
| Only-C | (cvss_C=C or cvss_C=P) and cvss_I=N and cvss_A=N |
| Only-I | cvss_C=N and (cvss_I=C or cvss_I=P) and cvss_A=N |
| Only-A | cvss_C=N and cvss_I=N and (cvss_A=C or cvss_A=P) |
| C & I | (cvss_C=C or cvss_C=P) and (cvss_I=C or cvss_I=P) and cvss_A=N |
| A & C | (cvss_C=C or cvss_C=P) and cvss_I=N and (cvss_A=C or cvss_A=P) |
| A & I | cvss_C=N and (cvss_I=C or cvss_I=P) and (cvss_A=C or cvss_A=P) |

**Fig. 3.** Configurations of impact on CIA

### 7.1 Analysis of single attributes

The total number of CVEs in the data set used in this section is **27,273**. Table 4(a) shows the distribution of CVEs in terms of access required for their exploitation. It demonstrates that the vast majority of CVEs require local access to the vulnerable host.

[9] We used a laptop with an Intel Pentium M (1.86GHz) and 1GB of main memory. The version of MonetDB/XQuery used is 0.22.

| exploitability | | #CVEs | |
|---|---|---|---|
| access | network | 22885 | 83.9% |
| | local | 4329 | 15.9% |
| | adjacent network | 59 | 0.2% |
| complexity | low | 20762 | 76.1% |
| | medium | 4824 | 17.7% |
| | high | 1687 | 6.2% |
| authentication | none | 26480 | 97.1% |
| | single | 787 | 2.9% |
| | multiple | 6 | 0.02% |
| **privilege** | | **#CVEs** | |
| | admin | 3462 | 12.7% |
| | user | 2233 | 8.2% |
| | other | 7292 | 26.7% |
| | no privilege | 14286 | 52.4% |

(a) Exploitability and privilege

| impact | | #CVEs | |
|---|---|---|---|
| confidentiality | none | 8168 | 29.9% |
| | partial | 14646 | 53.7% |
| | complete | 4459 | 16.3% |
| integrity | none | 7344 | 26.9% |
| | partial | 15635 | 57.3% |
| | complete | 4294 | 15.7% |
| availability | none | 7890 | 28.9% |
| | partial | 14429 | 52.9% |
| | complete | 4954 | 18.2% |
| **clues** | | **#CVEs** | |
| gainAdmin | yes | 406 | 1.5% |
| | no | 26867 | 98.5% |
| runCode | yes | 3839 | 14.1% |
| | no | 23434 | 85.9% |
| obtainCred | yes | 131 | 0.5% |
| | no | 27142 | 99.5% |
| DoS | yes | 4933 | 18.1% |
| | no | 22340 | 81.9% |

(b) Impact and clues from description

**Fig. 4.** Distribution of CVEs by single attributes

Figure 5 illustrates the initial distribution of CVEs based on privilege acquired by exploiting them, according to Table 4(a). This distribution is interesting because it demonstrates that almost 80% of the CVEs (privilege "other" and no privilege) are not clearly classified in terms of resulting privilege for the attacker. This is a good opportunity for further investigation.

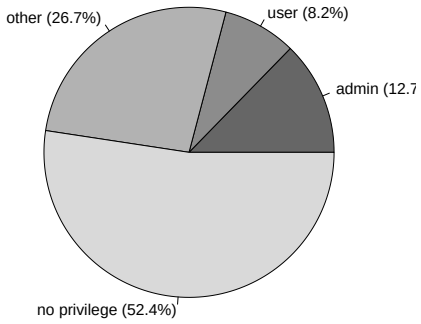



**Fig. 5.** Initial distribution of CVEs by privilege

Table 4(a) also shows the distribution of complexity level for exploiting CVEs and the distribution of authentication required for exploiting them. Results for the former demonstrate that only 6.2% of CVEs involve a "high" level of complexity to be exploited. Results for the latter demonstrate that 97.1% of CVEs require no credentials to be exploited. This is more or less confirmed by the fact that 93.8% of CVEs involve "low" or "medium" levels of complexity since the need for authentication, at least in part, increases the level of complexity for exploitation. Table 4(b) shows the number of CVEs in terms of level of impact caused on the exploited host. What seems curious about this table is the similarity among the numbers.

**Hyphothesis 3**: Impact on C, I and A usually match. It means e.g. that when a vulnerability causes *partial* impact of confidentiality, it also causes *par-*

| impact | admin | user | other | no privilege |
|---|---|---|---|---|
| only-C | 0 | 0 | 0 | 3232 |
| only-I | 0 | 0 | 0 | 3931 |
| only-A | 0 | 0 | 0 | 3965 |
| complete CIA | 3462 | 0 | 0 | 787 |
| unknown CIA configuration | 0 | 2233 | 7292 | 2371 |
| total | 3462 | 2233 | 7292 | 14286 |

**Fig. 6.** Distribution of CVE by privilege against type of impact

*tial* impact on integrity and availability.

Table 4(b) also shows some effects resulting from the exploitation of CVEs we would like to get insights about. First the table demonstrates that there are 406 CVEs of the type "gainAdmin". However, we have seen in Table 4(a) that 3462 CVE result in "admin" privilege. A next step is to investigate if the "gainAdmin" set is a subset of the "admin" set or not. If confirmed, this last set can be simply ignored. If not confirmed, the "admin" set can be enlarged with elements from the "gainAdmin" set. We perform this analysis in the next section.

**Conclusion 1**: Hypothesis 1 is not confirmed in practice since only 0.5% (Table 4(b)) of CVEs allow obtaining credentials, according to clues derived from their descriptions.

We examine attributes together in the next section.

### 7.2 Analysis of relations between attributes

The complete data set, containing 27,273 CVE, is now used for the analysis of multiple attributes. Table 6 shows the distribution of CVEs in terms of privilege against type of impact.

This distribution leads us to the following remarks and to Conclusions 2 and 3.

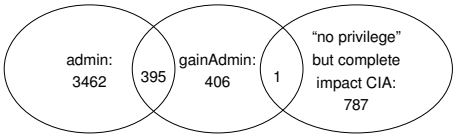
- All CVEs classified with privilege "admin" have complete impact on CIA.
- The set of CVEs with no privilege can be split into four subsets containing specific impact configurations: only-C, only-I, only-A, and complete CIA (refer to Table 3).

**Conclusion 2**: The set of CVEs with no privilege but complete CIA impact (787 CVEs) can be reclassified with privilege "admin".

**Conclusion 3**: The set of CVEs with no privilege contains three significant (i.e. at least 5% of CVEs) subsets: only-C, only-I and only-A.

| impact | admin | user | other | no privilege |
|---|---|---|---|---|
| only-C | 0 | 0 | 0 | 1 |
| only-I | 0 | 0 | 0 | 2 |
| only-A | 0 | 0 | 0 | 1 |
| complete CIA | 395 | 0 | 0 | 1 |

(a) CVEs with gainAdmin clue




(b) Sets of admin-related CVEs

**Fig. 7.** Figuring out reclassification of "admin"

If we consider Tables 4(b), 6 and 7(a), we realise there are two overlaps between CVE which contain clues of root privilege (gainAdmin = y) with (i) CVEs classified as "admin" and with (ii) CVEs with no privilege but which result in complete CIA impact, as illustrated in Figure 7(b). Thus, from a total of 406 CVEs with gainAdmin, 395 overlap with the former set and 1 overlaps with the latter set, resulting in 10 CVEs $(406 - 395 - 1)$ which can be reclassified as "admin". These are outliers because they do not imply complete CIA impact[10].

The resulting distribution of CVEs in terms of privilege is shown in Figure 8.

**Conclusion 4**: There are 10 CVEs which can also be reclassified with privilege "admin" based on clues derived from their descriptions.

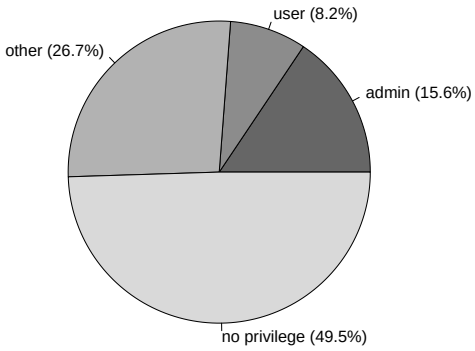



**Fig. 8.** Distribution of CVEs after "admin" reclassification

Based on conclusions 2 and 4, we reclassified 797 CVEs as "admin". Most of them had no privilege.

We proceed with the analysis by looking at the distribution of CVEs in terms of resulting privilege and impact against access required for their exploitation (see Table 9).

The following remarks can be drawn from this table: (i) 23.7% of CVEs can be classified under the following access-to-effect: network-to-admin (9.8%), network-to-user (6.8%), local-to-admin (5.8%) and local-to-user (1.3%), while 76.3%, including privilege "other" and no privilege, remains with unclear effect; (ii) no conclusions can be drawn for the access "adjacent network", since it covers several sets of privilege and impact. It will be investigated in the next section. However, it is interesting to

[10] Outliers: CVE-1999-0132, CVE-1999-0133, CVE-1999-0333, CVE-1999-0706, CVE-2000-0163, CVE-2000-0224, CVE-2000-0901, CVE-2003-0261, CVE-2005-3269, and CVE-2005-3784.

| privilege | impact | network | local | adjacent network | total |
|---|---|---|---|---|---|
| reclassified admin | | 2676 | 1574 | 9 | 4259 |
| user | | 1857 | 366 | 6 | 2229 |
| other | | 6724 | 558 | 8 | 7290 |
| no privilege | only-C | 2725 | 502 | 4 | 3231 |
| | only-I | 3517 | 411 | 1 | 3929 |
| | only-A | 3387 | 553 | 24 | 3964 |
| | unknown CIA configurations | 1999 | 365 | 7 | 2371 |
| total | | 22885 | 4329 | 59 | 27273 |

**Fig. 9.** Privilege and type of impact against access required to exploit CVEs

observe that 40.7% of these are related to CVEs with only-A impact. This issue is examined in the next section.

We looked next at all classes of CVEs identified so far (with reclassified admin) from the perspective of runCode and DoS clues gathered from CVEs' descriptions, as defined in Table 1. Table 12 shows the resulting distribution.

The total of CVEs which causes DoS in Table 12 matches the total presented in Table 4(b). However, there is a difference of one CVE between these two tables in respect to runCode. It refers to the outlier CVE-2007-4060 which does not match any of the classes identified, since it has impact cvss_C=P, cvss_I=P and cvss_A=C. As expected intuitively, DoS does impact heavily on availability, thus 3695 out of 3964 (93.2%) of only-A CVEs cause DoS. Intriguing are the 112 CVEs classified as runCode but which cause only-A impact, shouldn't they be classified as DoS? In fact, as shown by the last column of Table 12, there is an overlap of 98 CVEs between CVEs which cause only-A impact but belong to both DoS and runCode sets. It indicates that these CVEs cause DoS by (probably automatic) code execution, except for 0.4% of the cases, i.e. 14 out of 3709 ($= 3695 + 112 - 98$). We refer, from now on, to both sets of Dos and only-A interchangeably.

| impact | user | other | no privilege | total |
|---|---|---|---|---|
| partial CIA | 2229 | 7290 | 1340 | 10859 |
| no CIA | 0 | 0 | 59 | 59 |
| A & I | 0 | 0 | 213 | 213 |
| A & C | 0 | 0 | 88 | 88 |
| C & I | 0 | 0 | 668 | 668 |
| total | 2229 | 7290 | 2368 | 2368 |

**Fig. 11.** Understanding isolated CVEs with partial CIA impact

CVEs which allow to run code, however, impact mostly on all three aspects of security, i.e. on C, I and A. Roughly, "runCode" either impact CIA completely (33.8%) or partially (60.5%). Excluding CVEs with admin privilege, 91.4% (2324 out of 2542) of CVEs which result in runCode cause partial CIA impact. It suggests that the set of runCode CVEs is a subset of the partial CIA set, with a representation of 21% (2324 out of 10859).

**Conclusion 5**: Among CVEs with privilege "user", "other" and without privilege, there are two sets of CVEs: DoS and runCode. The former and the set of only-A impact CVEs will be addressed interchageably because of high degree of overlap between them. The latter is a subset of the partial CIA set.

**Conclusion 6**: Hypothesis 2 is not confirmed since only 33.7% (1296/3839) of runCode CVEs and 7.3% (359/4933) of DoS CVEs result in "admin" privilege.

A natural next step is to narrow down the investigation to understand which CIA configurations there are in the set of CVEs with privilege "other", "user" and without privilege (curiosity triggered by Table 6). Weka [37] visualization facilities provided clues for this next step, as illustrated in Figure 10. This figure shows that, although CVEs without privilege have varied impact on confidentiality (C) 10(a), integrity (I) 10(b) and availability (A) 10(c), demonstrated by the fact that the first column of these three plots have three segments corresponding to C (complete), N (none), P (partial), CVEs with privilege "other" and "user" have *only* partial impact on C, I and A. Table 11 confirms that, and shows the complete set of CIA impact configurations present in the set of CVEs with no privilege. It leads us to Conclusions 7 and 8.

**Conclusion 7**: The set of "user" and "other" is composed of CVEs that *always* result in partial CIA impact. There is a subset of CVEs with no privilege which also causes partial CIA impact, and another which causes no CIA impact. Adding to this, we have seen a significant set of CVEs resulting in complete CIA impact. Hence, hypothesis 3 is confirmed since, for 55.8% of CVEs, confidentiality, integrity and availability match.

From this conclusion, a natural question to ask is: "which criteria distinguish CVEs classified as "user", "other", and the subset of CVEs without privilege but with partial CIA impact?". This will be investigated in the next section. Furthermore, the group of 59 CVEs which result in no impact at all on CIA is puzzling. We also examine this in the next section.

**Conclusion 8**: Apart from 3 outliers, the set of CVEs with no privilege (13,495) contains the following impact subsets: (from Table 9 and 11): (i) only-C, (ii) only-I, (iii) only-A, (iv) partial CIA, (v) no CIA, (vi) A & I, (vii) A & C, and (viii) C & I (i.e. it contains 8 out of 9 subsets identified during the whole analysis process, see Table 3).

The outliers are CVE-2007-3266 has impact: A=P C=C I=P, CVE-2007-3949 has impact: A=C C=P I=P, and CVE-2007-4060 has impact: A=C C=P I=P.

### 7.3 Analysis of specific aspects

In this section, we zoom into the data set to try understanding CVEs with no impact, CVEs with "adjacent network" access, and to find insights which could distinguish CVEs with partial CIA impact among privilege in terms of exploitability and type of vulnerability.

We learnt, by manual analyzes of the set of CVEs with no CIA impact, that:

| privilege | impact | DoS | runCode | overlap |
|---|---|---|---|---|
| re-class. admin | | 359 | 1296 | 206 |
| user | partial CIA | 332 | 1292 | 278 |
| other | partial CIA | 295 | 834 | 218 |
| no privilege | partial CIA | 113 | 198 | 38 |
| | only-C | 6 | 3 | 0 |
| | only-I | 9 | 49 | 1 |
| | only-A | 3695 | 112 | 98 |
| | A & I | 58 | 32 | 16 |
| | A & C | 63 | 4 | 3 |
| | C & I | 1 | 17 | 0 |
| | no CIA | 2 | 1 | 1 |
| total | | 4933 | 3838 | 859 |

**Fig. 12.** Distribution of CVEs in terms of DoS and runCode

- 69.5% refer to warning, e.g. "The Telnet service is running", "A component service related to NIS is running", "Anonymous FTP is enabled", "A hacker utility, backdoor or Trojan Horse is installed" (they are all classified with "network" access and complexity "low")
- 22% refer to CVEs with unknown impact and access vector, e.g CVE-2005-222
- 8.5% refer to misclassified CVEs, e.g. CVE-2007-4526: "...allows local users to obtain sensitive information...", CVE-2007-3801: "...allows remote attackers to cause Denial of Services..."

As a remark, it is interesting to notice that, although warning CVEs are all classified with complexity "low", they involve in reality no complexity at all.

Further investigation of the set of CVEs with "adjacent network" access revealed that they refer to vulnerabilities related to e.g. wireless network using technology like Bluetooth, and printer drivers. Additionally, 90% (53 out of 59) of them cause some degree of availability impact (considering the partial CIA, complete CIA and only-A sets), although no more than half contain DoS clues in their description.

We then isolated the set of CVEs that result in partial CIA impact, i.e. **10,859** CVEs from privilege "user", "other" and without privilege for further analysis (refer to Table 11). We used Weka [37] to find associations among the nominal attributes: access (cvss_AV), complexity (cvss_AC), authentication (cvss_Au), confidentiality (cvss_C), integrity (cvss_I), availability (cvss_A), privilege, vulnerability type (vuln_type) (see Table 1 for more details).

The Predictive Apriori algorithm returned 100 association rules among these attributes. Figure 13 shows the first 30 rules with higher accuracy rate (acc). Analysing these associations we see that, in fact, the rules relate exploitability (i.e. attributes cvss_AV, cvss_AC, cvss_Au), privilege, and type of vulnerability, since impact on C, I and A are the same (=partial) among all CVEs isolated.

The associations predicted can be summarized in terms of exploitability subscores, by applying the exploitability attributes to the CVSS scoring system [35], while maintaining impact attributes constant (see Table 1):

- Exploitability 10.0, 8.6, and 4.9 correspond to cvss_AV=N, cvss_AC=L,M,H, cvss_Au=N; 86.8% of CVEs have this exploitability configuration.
- Exploitability 3.9, 3.4, and 1.9 correspond to cvss_AV=L, cvss_AC=L,M,H, cvss_Au=N; 10.3% of CVEs have this exploitability configuration.
- Exploitability 8.0, 6.8, and 3.9 correspond to cvss_AV=N, cvss_AC=L,M,H, cvss_Au=S; 2.4% [11] of CVEs have this exploitability configuration.

Figure 14(a) shows the distribution of the isolated CVEs in terms of exploitability. No new visible insights towards distinguishing the isolated CVEs per privilege could be identified from this analysis. Nevertheless, the plot confirms that the majority of CVEs are easily exploitable, since they can be exploited remotely (through network access), involves low complexity and requires no authentication.

Figure 14(b) shows the distribution of the isolated CVEs in terms of types of vulnerability [12]. It makes evident that the type "input", i.e. buffer overflow and boundary condition errors, dominates by far the isolated CVEs. In fact, 82.8% of the total number of CVEs with partial CIA impact result from "input" (66.5%), "access" (5.2%) and "design" (11.1%) flaws.

The insights obtained in terms of exploitability and types of vulnerability among the isolated CVEs triggered our curiosity towards what happens among CVEs with only-C, only-I and only-A impact. Table 15 shows the results and suggests Conclusions 9 and 10.

**Conclusion 9**: While exploiting only-C and only-A CVEs typically (90%+) involve low complexity, exploiting only-I CVEs involve low complexity in only 33.3% but high complex in 57.5% of the cases.

**Conclusion 10**: Similar to what happens with CVEs with partial CIA impact, 70.4% of only-I CVEs relate to "input" flaws. However, the same does not happen with only-C and only-A CVEs, where only 33% and 32.5% respectively, relate to "input" flaws.

The knowledge acquired troughout the analysis is summarized in the next section.

### 7.4 Summary of insights from the NVD analysis

A summary of conclusions reachead along the analysis of the NVD repository are enumerated next.

[11] Included here 15 out of 1048 CVEs (1.4%) with exploitability 3.9 corresponding to cvss_AV=N, cvss_AC=H, cvss_Au=N.

[12] We grouped in *oth.* the following: multiple types of vulnerability for the CVE, type equal to "other" and empty type.

1. Six predominant classes of CVEs were identified, some have explicit effect on impact, i.e. partial CIA, only-I and only-C and others have implicit effect on impact, i.e. admin, DoS and runCode.
2. Admin level of privilege imply in complete CIA impact. However, 10 outliers, i.e. with no complete CIA impact, were identified which explicitely contained in their description the expression "gain root".
3. Two significant sets of DoS and runCode were identified among CVEs with "user", "other" and with no privilege. However, we found that they overlap by 9.2%. This overlap is explained by the fact that the ability to run code is a way used for automated DoS attacks. Nevertheless, there is a clear cut between these sets: DoS imply mostly on only-A while runCode is more generic and results mostly on either complete CIA or partial CIA impact.
4. A significant group of only-I impact was detected among CVEs with no privilege. Almost 60% of CVEs in this set involve high complexity for exploitation and 70% are exploited by "input" attacks, i.e. buffer overflow and boundary errors.
5. An also significant set of only-C CVEs was detected among CVEs with no privilege. These CVEs are typically easily exploitable, i.e. require network access, low complexity and no authentication, and $\frac{1}{3}$ of them derive from "input" flaws and amost another $\frac{1}{3}$ derive from "design" flaws.
6. 100% of CVEs with privilege "user" and "other" cause partial CIA impact. The set of CVEs with no privilege also contains a subset of partial CIA impact. We found that 86.8% of these require network access and no authentication although involve a varied levels of complexity for exploitation. However, although easily exploitabable and the fact that 82.8% of these CVEs are related to input, access and design flaws, no conclusive factor was detected to distinguish this set in terms of privilege.
7. Other clusters without significant representation were also detected among CVEs with no privilege. They are: no CIA impact, A & C, A & I and C & I (refer to Table 3).
8. The set of CVEs which cause no CIA impact are mostly (69%) composed of warnings. They either refer to services running which allow attackers to get context information such as day/time, valid users and environment variables, or to services which can establish, if enabled, a relationship of *trust* between two hosts, e.g. SSH, Telnet or FTP connections.

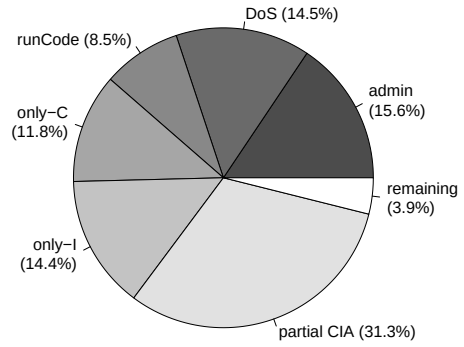
Based on these conclusions, two slightly different views of CVEs can be derived: an effect-view and an impact-view. The former is a mixture of implicit and explicit effects. The latter is a purely impact-driven view. Figure 16 illustrates both views (sets which represent less than 5% of CVEs were included in the pies as "remaining").

In the next section, we reflect both views on our initial vulnerability and composition models presented in Sections 4 and 5.

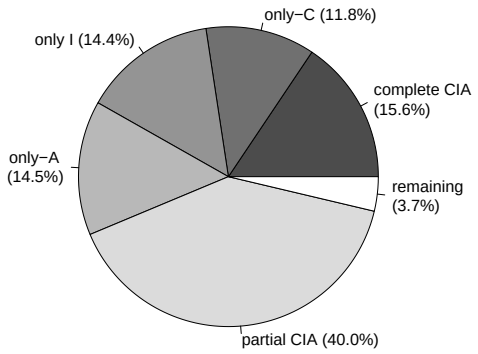
## 8 Evaluation

We evaluate in this section, the effect-view and the impact-view of CVEs over our initial vulnerability and composition models, presented in Sections 4 and 5. Moreover, we evaluate the experiences with using XML database technology for performing the analyses.

Since no significant high-level differences between CVEs which result in partial CIA impact were spotted, we use the effect "user", in the effect view, to refer to the partial CIA set of CVEs.




(a) Effect view




(b) Impact view

**Fig. 16.** Final distribution of CVEs

In our initial composition model we considered only movement between hosts once the source host was fully compromised, i.e. an attacker had reached "admin" level on it. However, warning CVEs (among CVEs with no CIA impact) have raised the flag of services which establish connections of *trust* between hosts. In this case, if the attacker obtains credentials for a host $h1$, such as an user account password, he can potentially access a host $h2$, via the connection, as the user and then exploit, for example, any local-to-{admin, runCode,DoS,only-C,only-I, partialCIA} CVE. Hence, although per Conclusion 1, only 0.5% of CVEs contain clues in their descriptions which indicate that they allow attackers to obtain credentials, the effect *obtainCred* will be maintained in our reviewed model, not associated with CVEs which require credentials but with warning CVEs.

In the next two subsections, we present the rules of compromise updated, according to both views.

### 8.1 Reviewed rules of compromise

The effect and impact views do not differ concerning the composition model as the considerations for *moving* from a host to another apply for both. An attacker can *move* from host $h1$ to host $h2$ if:

- (i) $h2$ is reachable from $h1$ via a *trusted* [13] service/interface, (ii) $h1$ is compromised in terms of network-to-obtainCred, and (iii) there is at least one vulnerability of the type local-to-{admin,runCode,DoS,only-C,only-I} in $h2$
- (i) $h2$ is reachable from $h1$ via a *non-trusted* service/interface, (ii) $h1$ is fully compromised according to the rules of compromise, and (iii) at least one vulnerability of the type network-to-{admin,runCode,DoS,only-C,only-I} exists in $h2$

Next we review the rules for the effect-view, considering an attacker $A$.

Rule 1 $A$ exploited a network-to-admin CVE.
Rule 2 $A$ exploited one or several network-to-{user,runCode,DoS,only-C,only-I} CVEs, followed by a local-to-admin CVE.

Similarly, the rules of compromise for the impact-view are:

Rule 1 $A$ exploited a network-to-completeCIA CVE.
Rule 2 $A$ exploited one or several network-to-{partialCIA,only-C,only-I,only-A} CVEs, followed by a local-to-completeCIA CVE.

The advantage of the impact-view over the effect-view is that the effect-view has implicit impacts which might not be always straightforward. For example, it is not evident that the effects runCode and "user" imply partial CIA impact, as we have revealed in our analysis.

### 8.2 Evaluation of experiences with using XML database technology for performing the analyses

To reproduce all tables 4(a) to 16, we defined four generic functions of in total about 100 lines of XQuery.[14] One function determines the distribution of values in a given child of a sequence of elements. Another function determines correlations between two given children of a sequence of elements. We also defined generalizations of these that accept an XML fragment containing a class specification. This is used, for example, to convey what "partial/complete C" means in terms of conditions on attributes in the data set.Using these functions, the queries need on average a mere 22 lines and all execute in interactive time (within a few seconds). Another observation that can be made is that these queries are very simple in nature and all follow the same design pattern.

[13] We consider as trusted service/interface the type of connections which are considered as warning CVEs.

[14] The source of the queries can be downloaded from http://wwwhome.cs.utwente.nl/~keulen/pub/nvd_cvss_xquery.zip.

We should note that we did transform the original XML files to one XML document structured as a sequence of entries with each attribute of Table 1 as child element. This proved useful as it simplified the queries. Especially the encoding of several attributes in one string for CVSS_vector in the original NVD XML files is unsuitable for analysis and it is advisable to split it. The transformation query contains 75 lines of XQuery.

We observed an important qualitative characteristic favoring the use of XML database technology. As common in analytical research, some scripts (Python) were developed for the main process to aid in the analysis. In the XML database case study there was no need for that as all manipulations could be expressed in XQuery. Scripting, however, involves manipulation of the data on the syntax level essentially some parsing and generation of new data. Being able to stay in the XML domain in the case study appeared to be less error prone, especially regarding low-level syntactical mistakes with row and field separators and quoting. Such low-level errors are inconceivable when using XML technology, because it properly handles all syntactical issues, hence protecting the user against such low-level errors. Moreover, such low-level mistakes may go unnoticed due to the inherent aggregation in the analysis.

We expect that the scalability properties of XML database technology [14] allows similar analyses on considerable larger data sets than involved in this case study. We intend to investigate this expectation. Furthermore, association rule mining as in Figure 13 was not among the aims for this case study as it requires special algorithms (Predictive Apriori). We also plan to investigate the feasibility of enhancing MonetDB/XQuery with support for such algorithms. Enhancing *relational* databases in this way is a known approach, see e.g. [38].

## 9 Conclusion

We analysed the NVD repository with three aims. The first aim was to understand associations between attributes. This analysis has allowed us to revert an initial state where almost 80% of CVEs (with privilege “other” or with no privilege), had unclear effect to a state where we know which clusters they contain. Additionally, we derived from the analysis two views of CVEs related to effect: one called ”effect view” which is a mixture of explicit effect and impact, and one called ”impact view” which is purely impact-driven. The impact-view appears better towards quantitative analysis of impact expected by attack scenarios. The second aim was to gain insights about our initial vulnerability and composition models. In this sense we were able to check hypothesis, to verify the coverage of the models and, in the end, to review them. Furthermore, in respect to our third aim, we can conclude that XML database technology is mature enough to carry out such research by writing XQuery queries, which already proved to be less error prone. As future work, we plan to incorporate insights from the reviewed composition model into our approach to learn attack scenarios from network graphs [39].

### Acknowledgements

Research supported by the research program Sentinels (www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organisation for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

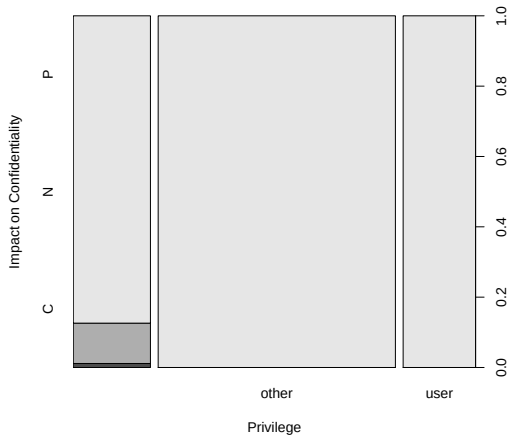
Thankful to dr. Pascal van Eck for review of this report.

## References

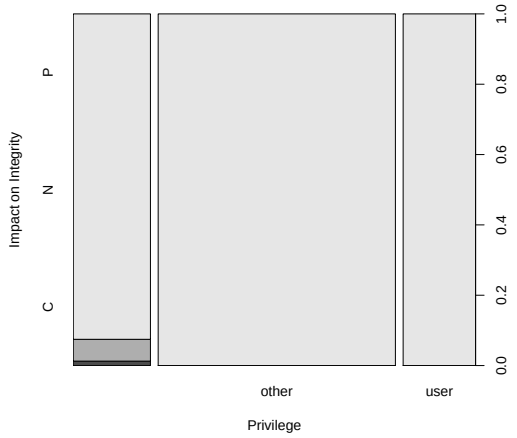
1. CVE: (Common Vulnerabilities and Exposures dictionary) http://cve.mitre.org.
2. Anderson, R., Moore, T.: Information Security Economics - and Beyond. In: CRYPTO. (2007) 68–91
3. Red-Hat: Red hat enterprise linux 4.5.0: Security guide. Red Hat, Inc. (2007) http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/en-US/Security_Guide/.
4. Leeson, P.T., Coyne, C.J.: The Economics of Computer Hacking. Journal of Law, Economics and Policy **1**(2) (2006) 511–532
5. Keeney, M., Kowalski, E., Cappelli, D., Moore, A., Shimeall, T., Rogers, S.: Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors (2005) U.S. Secret Service and CERT Coordination Center.
6. Security, T.N.: (The Nessus Security Scanner) www.nessus.org.
7. NVD: (National Vulnerability Database v2) http://nvd.nist.gov.
8. NIST: (National Institute of Standards and Technology) http://www.nist.gov.
9. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: a logic-based network security analyzer. In: SSYM'05: Proc. of the 14th Conf. on USENIX Security Symposium, Berkeley, CA, USA, USENIX Association (2005)
10. Lippmann, R., Ingols, K., Scott, C., Piwowarski, K., Kratkiewicz, K., Artz, M., Cunningham, R.: Validating and Restoring Defense in Depth Using Attack Graphs. In: MILCOM 2006: Military Communications Conference. (2006) 1–10
11. Frei, S., May, M., Fiedler, U., Plattner, B.: Large-Scale Vulnerability Analysis. In: LSAD '06: Proc. of the 2006 SIGCOMM workshop on Large-scale attack defense, New York, NY, USA (2006) 131–138
12. Lippmann, R.P., Ingols, K.W., Scott, C., Piwowarski, K., Kratkiewicz, K.J., Artz, M., Cunningham, R.K.: Evaluating and Strengthening Enterprise Network Security Using Attack Graphs. Technical Report ESC-TR-2005-064, Masschusetts Institute of Technology, Lincoln Laboratory, Masschusetts, USA (2005)
13. XQuery 1.0: An XML Query Language. W3C (2007) http://www.w3.org/TR/xquery.
14. Boncz, P., Grust, T., van Keulen, M., Manegold, S., Rittinger, J., Teubner, J.: MonetDB/XQuery: a fast XQuery processor powered by a relational engine. In: Proc. of the 2006 ACM SIGMOD Int'l conf. on Management of data, Chicago, IL, USA. (2006) 479–490
15. Welberg, S.M.: Vulnerability management tools for COTS software - A comparison. Technical Report TR-CTIT-08-15, Centre for Telematics and Information Technology, University of Twente, Enschede (2008) http://eprints.eemcs.utwente.nl/12034/.
16. Lippmann, R.P., Ingols, K.W.: An Annotated Review of Past Papers on Attack Graphs. Technical Report ESC-TR-2005-054, Masschusetts Institute of Technology, Lincoln Laboratory, Masschusetts, USA (2005)
17. Sheyner, O., Wing, J.: Tools for Generating and Analyzing Attack Graphs. In: In Proc. of Workshop on Formal Methods for Components and Objects. LNCS 3188, Germany, Springer-Verlag (2004) 344–371

18. Phillips, C., Swiler, L.P.: A Graph-Based System for Network-Vulnerability Analysis. In: NSPW '98: Proc. 1998 workshop on New Security Paradigms, New York, NY, USA, ACM Press (1998) 71–79
19. Ou, X., Boyer, W.F., McQueen, M.A.: A Scalable Approach to Attack Graph Generation. In: CCS '06: Proc. of the 13th ACM Conf. on Computer and Communications Security, New York, NY, USA, ACM (2006) 336–345 people.cis.ksu.edu/~xou/publications/ccs06.pdf.
20. S. Jajodia, S. Noel, B.O.: Topological Analysis of Network Attack Vulnerability. In: Managing Cyber Threats: Issues, Approaches and Challenges. Springer (2005)
21. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. In: NSPW'00: Proc. of the 2000 Workshop on New Security Paradigms, New York, NY, USA, ACM (2000) 31–38
22. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: SP'02: Proc. of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (2002) 202–215
23. Dawkins, J., Hale, J.: A Systematic Approach to Multi-Stage Network Attack Analysis. In: IWIA '04: Proc. of the 2nd IEEE Int. Information Assurance Workshop, Washington, DC, USA, IEEE Computer Society (2004) 48–56
24. Ning, P., Xu, D.: Learning Attack Strategies from Intrusion Alerts. In: CCS'03: Proc. of the 10th ACM Conf. on Computer and Communications Security, New York, NY, USA, ACM (2003) 200–209
25. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: CCS '02: Proceedings of the 9th ACM conference on Computer and communications security, New York, NY, USA, ACM (2002) 217–224
26. Kotenko, I., Stepashkin, M.: Attack Graph Based Evaluation of Network Security. In: CMS'06: Proc. of the 10th IFIP TC-6 TC-11 Int. Conf. on Communications and Multimedia Security. LNCS 4237, Germany, Springer-Verlag (2006) 216–227
27. Kumar, S.: Classification and Detection of Computer Intrusions. PhD thesis, Purdue University, Department of Computer Sciences (1995)
28. Kruegel, C., Valeur, F., Vigna, G.: Intrusion Detection and Correlation - Challenges and Solutions. Volume 14 of Advances in Information Security. Springer Verlag, New York, NY, USA (2005)
29. Arora, A., Krishnan, R., Nandkumar, A., Telang, R., Yang, Y.: Impact of Vulnerability Disclosure and Patch Availability - An Empirical Analysis. In: WEIS'04: Third Workshop on the Economics of Information Security. (2004)
30. Corporation, S.S.: (Bugtraq maling list) www.securityfocus.com.
31. osvdb: (Open Source Vulnerability Database) www.osvdb.org.
32. Beattie, S., Arnold, S., Cowan, C., Wagle, P., Wright, C.: Timing the Application of Security Patches for Optimal Uptime. In: LISA '02: Proc. of the 16th USENIX conf. on System administration, Berkeley, CA, USA (2002) 233–242
33. Tierney, S.: Knowledge Discovery in Cyber Vulnerability Databases. Master of science, Computing and Software Systems, University of Washington (2005) http://www.tacoma.washington.edu/tech/docs/research/gradresearch/STierney.pdf.
34. Schumacher, M., Haul, C., Hurler, M., Buchmann, A.: Data Mining in Vulnerability Databases. Darmstadt University of Technology (2000) 12 p., http://www.ito.tu-darmstadt.de/publs/pdf/sdb-dfn-cert-eng.pdf.
35. Mell, P., Scarfone, K., Romanosky, S.: A Complete Guide to the Common Vulnerability Scoring System, version 2.0. Published by FIRST - Forum of Incident Response and Security Teams (2007) http://www.first.org/cvss/cvss-guide.pdf.
36. Python: (Programming language) http://www.python.org/.

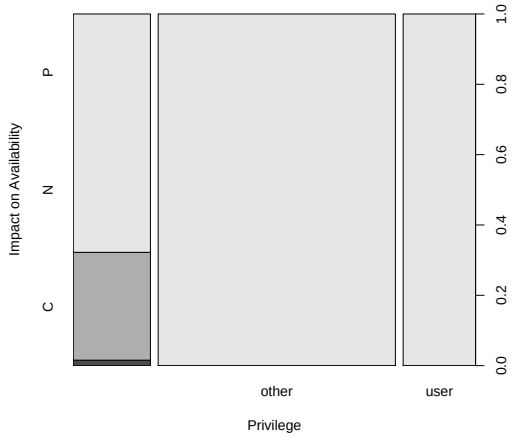
37. Weka: (Data mining software) http://www.cs.waikato.ac.nz/ml/weka/.
38. Carrasco, R., Vila, M., Aguilar, M.: A new language and architecture to obtain fuzzy global dependencies. In: Proc. of the 1st Int'l VLDB workshop on Management of Uncertain Data (MUD), Vienna, Austria. (2007) 33–47
39. Franqueira, V.N.L., Lopes, R.H.C.: Vulnerability Assessment by Learning Attack Specifications in Graphs. In: IAS'07: Proc. of the 3rd Int. Symposium on Information Assurance and Security). (2007)




(a) Privilege against impact on C




(b) Privilege against impact on I



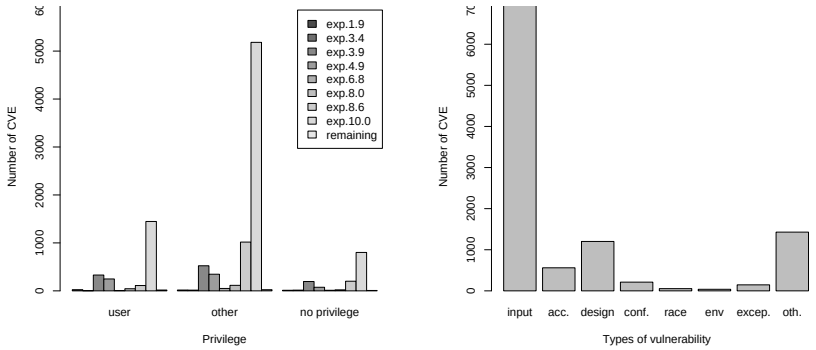

(c) Privilege against impact on A

**Fig. 10.** Plots of isolated CVEs: with no privilege (first bar), and with privileges "other" & "user"

Best rules found:

1. cvss_AC=L cvss_AV=N vuln_type=config 144 ==> cvss_Au=N 144 acc:(0.99465)
2. cvss_AC=M privilege= vuln_type=input 126 ==> cvss_AV=N 126 acc:(0.99449)
3. cvss_AC=L vuln_type=config 199 ==> cvss_Au=N 198 acc:(0.99444)
4. cvss_AC=H cvss_Au=N privilege=other vuln_type=input 276 ==> cvss_AV=N 274 acc:(0.99441)
5. cvss_AC=H privilege=user 274 ==> cvss_Au=N 272 acc:(0.99439)
6. cvss_AC=L cvss_AV=L privilege=other vuln_type=design 192 ==> cvss_Au=N 191 acc:(0.99435)
7. cvss_AV=N vuln_type=config 156 ==> cvss_Au=N 155 acc:(0.99364)
8. cvss_AC=H vuln_type=multiple 79 ==> cvss_Au=N 79 acc:(0.99348)
9. privilege= vuln_type=access 74 ==> cvss_Au=N 74 acc:(0.99327)
10. vuln_type=config 212 ==> cvss_Au=N 210 acc:(0.99294)
11. cvss_AV=L privilege= vuln_type=input 66 ==> cvss_Au=N 66 acc:(0.99285)
12. cvss_AC=H privilege=other vuln_type=input 282 ==> cvss_AV=N 279 acc:(0.99257)
13. cvss_AC=L cvss_AV=N privilege= vuln_type=multiple 56 ==> cvss_Au=N 56 acc:(0.99211)
14. cvss_AV=L vuln_type=config 55 ==> cvss_AC=L 55 acc:(0.99202)
15. cvss_AC=H vuln_type=race 49 ==> cvss_Au=N 49 acc:(0.99138)
16. cvss_AV=N privilege= vuln_type=multiple 109 ==> cvss_Au=N 108 acc:(0.9913)
17. vuln_type=env 38 ==> cvss_Au=N 38 acc:(0.98959)
18. cvss_AV=L privilege=other vuln_type=multiple 38 ==> cvss_Au=N 38 acc:(0.98959)
19. cvss_AC=L privilege= vuln_type=design 228 ==> cvss_Au=N 225 acc:(0.98895)
20. cvss_AC=L cvss_AV=L privilege=user vuln_type=access 33 ==> cvss_Au=N 33 acc:(0.98835)
21. cvss_AV=L cvss_Au=N privilege=user vuln_type=access 33 ==> cvss_AC=L 33 acc:(0.98835)
22. cvss_AC=M privilege=user vuln_type=input 82 ==> cvss_Au=N 81 acc:(0.98802)
23. cvss_AC=L cvss_AV=L privilege=user vuln_type=design 82 ==> cvss_Au=N 81 acc:(0.98802)
24. cvss_AV=L privilege= vuln_type=design 80 ==> cvss_Au=N 79 acc:(0.98766)
25. privilege= vuln_type=exception 28 ==> cvss_Au=N 28 acc:(0.98666)
26. cvss_AV=L vuln_type=multiple 73 ==> cvss_Au=N 72 acc:(0.98619)
27. cvss_AC=L privilege= vuln_type=multiple 73 ==> cvss_Au=N 72 acc:(0.98619)
28. cvss_AV=L vuln_type=exception 26 ==> cvss_Au=N 26 acc:(0.9858)
29. cvss_AV=L privilege=other vuln_type=config 26 ==> cvss_AC=L cvss_Au=N 26 acc:(0.9858)
30. cvss_AC=L cvss_AV=L privilege=other vuln_type=input 129 ==> cvss_Au=N 127 acc:(0.98567)

**Fig. 13.** First 30 rules returned by Predictive Apriori from Weka [37]




(a) Exploitability (b) Types of vulnerability

**Fig. 14.** Distribution of isolated CVEs (with partial impact on CIA)

| characteristics | | only-C | only-I | only-A |
|---|---|---|---|---|
| access | network | 2725 | 3517 | 3387 |
| | local | 502 | 411 | 553 |
| | adjacent network | 4 | 1 | 24 |
| complexity | low | 2965 | 1308 | 3574 |
| | medium | 129 | 360 | 148 |
| | high | 137 | 2261 | 242 |
| authentication | none | 3145 | 3841 | 3877 |
| | single | 86 | 88 | 86 |
| | multiple | 0 | 0 | 1 |
| type_vuln | input | 1065 | 2767 | 1290 |
| | access | 314 | 121 | 39 |
| | design | 966 | 426 | 506 |
| total | | 3231 | 3929 | 3964 |

**Fig. 15.** Exploitability & type of vulnerability for only-C, only-I and only-A