

Accepted Manuscript

Title: Analysis of update delays in Signature-based Network Intrusion Detection Systems

Authors: Hugo Gascon, Agustin Orfila, Jorge Blasco



PII: S0167-4048(11)00110-6

DOI: [10.1016/j.cose.2011.08.010](https://doi.org/10.1016/j.cose.2011.08.010)

Reference: COSE 533

To appear in: *Computers & Security*

Received Date: 17 December 2010

Revised Date: 17 July 2011

Accepted Date: 28 August 2011

Please cite this article as: Gascon H, Orfila A, Blasco J. Analysis of update delays in Signature-based Network Intrusion Detection Systems, *Computers & Security* (2011), doi: 10.1016/j.cose.2011.08.010

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Analysis of update delays in Signature-based Network Intrusion Detection Systems

Hugo Gascon*, Agustin Orfila, Jorge Blasco

*Department of Computer Science, Carlos III University of Madrid, Leganes, Madrid,
28911, Spain*

Abstract

Network Intrusion Detection Systems (NIDS) play a fundamental role on security policy deployment and help organizations in protecting their assets from network attacks. Signature-based NIDS rely on a set of known patterns to match malicious traffic. Accordingly, they are unable to detect a specific attack until a specific signature for the corresponding vulnerability is created, tested, released and deployed. Although vital, the delay in the updating process of these systems has not been studied in depth. This paper presents a comprehensive statistical analysis of this delay in relation to the vulnerability disclosure time, the updates of vulnerability detection systems (VDS), the software patching releases and the publication of exploits. The widely deployed NIDS *Snort* and its detection signatures release dates have been used. Results show that signature updates are typically available later than software patching releases. Moreover, *Snort* rules are generally released within the first 100 days from the vulnerability disclosure and most of the times exploits and the corresponding NIDS rules are published with little difference. Implications of these results are drawn in the context of security policy definition. This study can be easily kept up to date due to the methodology used.

Keywords: Intrusion detection, vulnerability, signature update, exploit, patch, NIDS, VDS, *Snort*, *Nessus*

*Corresponding author

Email addresses: hgascon@gmail.com (Hugo Gascon), adiaz@inf.uc3m.es
(Agustin Orfila), jbalis@inf.uc3m.es (Jorge Blasco)

1. Introduction

It is well known that software development process is by far not perfect. The failure to follow secure coding practices along the lack of adequate and effective tools for the testing phase of the software life-cycle can lead to uncontrolled failures in running systems. In many occasions, these errors can be used by malicious users to modify the expected behavior of the original code, thus surpassing the limitations imposed by the programmer for their own benefit. Programming errors turn then into security vulnerabilities. The risk of these vulnerabilities being remotely exploited has dramatically increased over the last years due to the great development of communication networks. Former romantic hackers have been replaced by a crowd of economically driven attackers, whose efforts in breaking into systems only focus on achieving some sort of revenue. Therefore, the release of patches by software vendors as soon as new vulnerabilities are discovered is critical to ensure the availability of resources and to avoid loss of data integrity or information disclosure. Nevertheless, the inherent difficulties of the patch development process and the incapacity or unwillingness shown by vendors to release on time solutions to minimize system exposure, have triggered the security problem of *windows of vulnerability*, namely, the period of time vulnerabilities are disclosed but unpatched.

Vulnerability Detection Systems (VDS) are software tools used to discover vulnerable network services at risk of being exploited. The information obtained is managed by security administrators who should be willing to take actions to mitigate this risk while software updates are released. In spite of that, and not in few cases, the deployment of new patches in large network infrastructures involves such an effort that services are kept vulnerable for long periods of time.

Network Intrusion Detection Systems (NIDS) are introduced as a solution to monitor and detect attacks on vulnerable services. Although intrusion detection has become an extensive and promising research field, where anomaly detection techniques have been developed to deal with unknown vulnerabilities, misuse detection approaches based on signatures –rules written from intrusion trails– are the current standard in real scenarios. Commercial NIDS have evolved into Network Intrusion Prevention Systems (NIPS), which are capable of blocking ongoing detected attacks and introduce non-signature detection capabilities based on heuristics and behaviour analysis. Nevertheless, administrators do not go beyond the use of vendor-recommended signatures

1
2
3
4
5
6
7
8
9 in approximately 65% of new deployments. Neither are blocking capabilities
10 used in more than 25% of deployments and approximately only 10% of
11 enterprises make an advanced use of detection engines, developing custom
12 signatures and using anomaly detection techniques in order to identify un-
13 known attacks [35]. The main reason is the high number of false alarms that
14 anomaly detectors present, which can cause undesired traffic blocking and
15 increase the difficulty to keep the normal behavior of a system up to date.
16 As a consequence, every commercial network prevention system deployed in a
17 corporative environment is to a large extent based on the signature detection
18 paradigm and their performance rely thus, on the development of detection
19 rules by security researchers. As this task requires considerable effort and
20 extensive previous testing to avoid false alarms and inconsistencies, perfor-
21 mance of signature-based NIDS depends not only on high detection ratios,
22 but also on the time it takes developers to release a new detection rule when
23 a new vulnerability is disclosed. Nowadays, every corporate security program
24 takes into account the need of an intrusion detection system to increase vis-
25 ibility of events in networks but, not in few cases, the mere fact of deploying
26 the system causes network administrators to become overconfident about the
27 level of protection. If new detection rules are not released on time and the
28 security perception is strongly based on the NIDS performance, the risk of
29 missing a successful attack highly increases.

30
31
32
33
34
35
36 Some research have been conducted in measuring and comparing the
37 patch development process of vendors [18, 31, 23] and numerous studies have
38 examined different approaches for evaluating NIDS effectiveness [33, 19, 28,
39 29, 15]. However, vulnerable time windows caused by delays in the updating
40 process of signature-based NIDS have not been yet explored and quantified
41 as a performance metric. The goal of this research is to fill this gap and apply
42 a formal methodology to evaluate a widely deployed open source signature-
43 based NIDS (i.e. *Snort* [6]) by means of measuring the *update delays* of
44 its detection rules, namely, the time interval between the release of a signa-
45 ture and the related security event. Accordingly, a time-span is statistically
46 modeled first from the existing delay between vulnerability disclosures and
47 specific rule releases. Then, the release of software patches is compared to
48 NIDS updates. Following, the confrontation is done against the updates of
49 a popular vulnerability scanner (i.e. *Nessus* [4]) and finally a comparison is
50 made between the publication of exploits and the corresponding NIDS rules
51 in order to measure the corresponding NIDS *update delay*. This comprehen-
52 sive work allows us to draw some conclusions, such as answering the question
53
54
55
56
57
58

of how useful signature-based NIDS can be to mitigate risks. Figure 1 depicts a general arrangement of the mentioned events. The relationships between their occurrence dates are quantified and statistically estimated in this work.

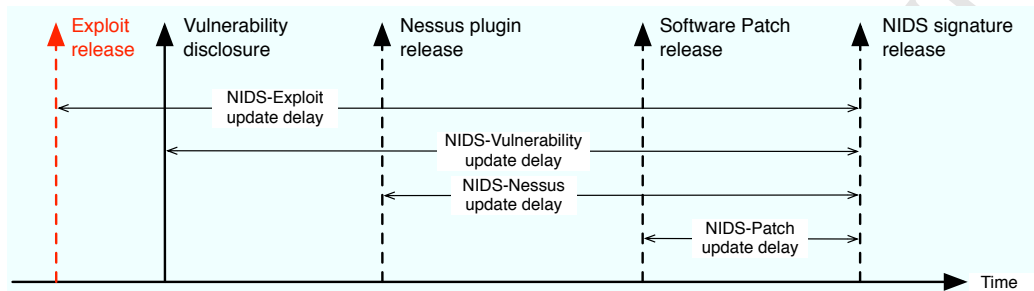


Figure 1: Timeline of security events considered as part of this study. These events do not necessarily occur in this order but let us depict their time relationships and the estimated *update delays* in a graphical and self-explanatory manner.

The rest of this paper is organized as follows. In Section 2 we establish the research context of this study and describe the related work in the field of NIDS evaluation. The goal pursued, the analyzed variables and the methodology followed to obtain valuable data is exposed in Section 3. Formal modeling of this data and the obtained numerical results are presented and discussed in Section 4. We gather the conclusions regarding the performance of the NIDS under study in Section 5. Finally, future work is introduced in Section 6.

2. Related Work

In 2002, Lippmann and Webster presented one of the first attempts to analyze the interaction between software patches, VDS and signature-based NIDS [22]. They introduced concepts such as “window of vulnerability” and “window of visibility”, namely the time interval when a compromised system can be detected by an IDS. Their work concludes that software patches, used to prevent vulnerabilities from being exploited, are available before or simultaneously with NIDS signatures. Thus, signature-based NIDS would be useless if patches were installed as soon as they become available. They also state that on large networks where it is impractical to eliminate all known

1
2
3
4
5
6
7
8
9 vulnerabilities, signature-based NIDS are still useful. Moreover, they point
10 out that on such networks information from VDS can be used to prioritize
11 the large numbers of extraneous alerts caused by failed attacks and normal
12 background traffic. Unfortunately, their investigation lacks statistical signif-
13 icance due to the fact that only eight vulnerabilities and their corresponding
14 timelines were analyzed.

15
16 The relationship between VDS and NIDS has been widely explored. Net-
17 work context monitoring can be integrated in commercial and open source
18 NIDS to reduce the number of false positives. The potential of correlating
19 *Snort* signatures, *Nessus* scripts and vulnerability databases, such as Bug-
20 traq [1], has been studied as well as to incorporate network context in detec-
21 tion signatures [25]. The conclusion reached was that format differences in
22 vulnerability databases and reference information hindered an efficient cor-
23 relation of security events between NIDS and VDS. However, their analysis
24 is based on the state of these elements at the time of the study (i.e. 2005).
25 They overlooked the dynamic behaviour of the VDS, NIDS and vulnerabil-
26 ity databases, namely, the continuous update of their rules, scripts or new
27 entries, in the case of databases. Equally important, target-based intrusion
28 detection systems based on joint operation of a VDS and a NIDS are still
29 being designed. In [16], the use of independent sets of rules for each system
30 provided by the original vendor is suggested. Although a new method for
31 rule creation based on queries is proposed for the combination of both sys-
32 tems, independent processes lead to the development of their respective set
33 of rules. As a consequence, their updating times mismatch, possibly resulting
34 in inconsistent correlations. It is important to note that all these solutions
35 trying to add network context information to the intrusion detection process
36 may decrease false positive rate and ease the NIDS management. Neverthe-
37 less, these proposals miss an statistical analysis of the time dependency of
38 their updating processes. No correlation can be made if the NIDS detects an
39 attack to a vulnerability that the VDS is not yet able to discover.

40
41 Many authors, as [27, 11], have tried to characterize the vulnerability
42 life-cycle and the most adequate policies to follow in disclosure. Neverthe-
43 less it remains a controversial field of research. As stated in [13], none of the
44 analyzed disclosure practices, immediate public, full vendor, or hybrid, is
45 optimal every time. Some authors [18, 31, 23] have measured the patch de-
46 velopment process in order to estimate the security risk arising from patching
47 policies. However, no comparison has been made with any intrusion detection
48 system. The work presented in [34] explores a quantitative characterization
49
50
51
52
53
54
55
56
57
58

1
2
3
4
5
6
7
8
9 of the vulnerability cycle based on several vulnerability related events such
10 as disclosure, patching and exploit creation time. The Open Source Vulner-
11 ability Database (OSVD) [5] is used as data source to calculate the time
12 intervals between events. Vulnerability disclosure and patch release events
13 are statistically characterized, but no information about NIDS updates is
14 provided. We use a similar approach in this paper to statistically character-
15 ize the update time response of signature-based NIDS. Frei and Tellenbach
16 [17] quantify the statistical distribution of exploit availability timing, before
17 and after vulnerability is disclosed. Patching availability time distribution
18 is also described, concluding about the trend towards increasing vulnerability
19 disclosures and zero-day exploits. In [21], the analysis of the evolution over
20 time of exploitable vulnerabilities suggests both that the proportion of high
21 and medium severity vulnerabilities has not changed during the last decade
22 and also that many developers are still ignoring security basics.

23
24
25
26
27 To improve the detection capability of *Snort*, the generalization of the
28 conditions and parameters used by detection rules has been proved useful [10].
29 In [20], a passive scanner is proposed, which is able to gather network packets
30 and build a topology of devices and services being active in the network.
31 Other research [15, 14, 29] have addressed the difficulty of evaluating NIDS.
32 Although many evaluation frameworks are being developed in very different
33 manners, most efforts focused on developing a method for analyzing IDS
34 effectiveness are based in improving detection ratios while reducing the false
35 alarm ratio or to generate better data sets to automatically test systems,
36 as in [26]. In very few cases the updating process of signature-based NIDS
37 is characterized or considered as a quality measure. In [32] the problem of
38 rule package update and the need to stop the engine to upload a new set of
39 rules is addressed. Some developments like the one presented in this paper
40 could be use to update the detection engine with one released rule at a time
41 without the handicap of long vulnerable windows due to the updating method
42 of packaged rules. In [12] some drawbacks of misuse based approaches are
43 examined, as the need of regularly updating a knowledge base in order to add
44 new intrusion scenarios, work that must be performed by experts or system
45 designers.
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

3. Goal and Methodology

First, this section presents the goal pursued by this research, then exposes the different variables that have been used as reference for calculations and finally establishes the experimental setup designed in order to gather correlated data from different sources.

3.1. Goal

The performance of intrusion detection systems is usually measured in terms of the trade-off between the detection rate and the false alarm rate. However, it also depends on how fast the detection engine is able to respond to new threats. Regarding signature-based NIDS, this is the time it takes the security researchers to release new rules when new vulnerabilities are disclosed. The aim of this work is to evaluate and characterize this time response in a widely deployed open source NIDS (i.e. *Snort*) using the concept of *update delays*. These delays provide information about the NIDS response against vulnerability disclosure and exploit release, and let us evaluate the utility of the NIDS in comparison to the patching development process and VDS plugin release.

3.2. Analyzed Variables

Four different variables have been defined to evaluate the update response of signature-based NIDS, namely, the vulnerability disclosure time, the software patch release time, the vulnerability scanner update time and the exploit release time. These variables allow to estimate and compare the NIDS *update delay* from different perspectives.

- *Vulnerability Disclosure Time*

In order to characterize the *update delay* of the NIDS rules from vulnerability disclosure, the time when vulnerabilities are made public has to be established. Two independent public and commonly referenced databases have been used for this purpose:

- *National Vulnerability Database (NVD) [3]*

The NVD comprises the *Common Vulnerabilities and Exposures* identifiers (i.e. CVEs) and intends to be a public vulnerability index where the correlation between products, vulnerabilities and

1
2
3
4
5
6
7
8
9 their severity is defined. Every CVE includes the date it was incor-
10 porated to the database. As CVEs are widely used as a reference
11 by the security research community, this date can be established as
12 the vulnerability disclosure time. A CVE definition also includes a
13 *Common Vulnerability Scoring System* (CVSS) index. CVSS is a
14 standard in vulnerability severity evaluation. In this work, CVSS
15 allows us to relate NIDS *update delay* to the related vulnerability
16 risk.
17
18
19

20
21 – *Bugtraq Mailing List* [1]

22 Bugtraq is a moderated mailing list based on the *full disclosure*
23 philosophy, whose goal is to facilitate the discussion between re-
24 searchers and the publication of new vulnerabilities and related
25 information. Each vulnerability is associated with a Bugtraq iden-
26 tifier and, in most definitions, the related CVE is provided in order
27 to correlate both databases. The publication date of the identifier
28 is established as the vulnerability disclosure time.
29
30
31

32
33 • *Software Patch Release Time*

34 We use the software patch release date to quantify the utility of the
35 NIDS versus the patching process. The *Open Source Vulnerability*
36 *Database* (OSVDB) [5] gathers a large collection of vulnerability defi-
37 nitions that are continuously updated. Each vulnerability has an ID,
38 a description, a classification and available external references. If the
39 release date of the solution to the vulnerability is included in the defi-
40 nition, it is used as the patch release date, which will be compared with
41 the associated Snort detection rule if it is also included. As stated in
42 [34], the OSVDB is one of the most complete public sources of infor-
43 mation about vulnerabilities and allows us to establish the time when
44 security patches are made available by vendors. The Snort rule release
45 date is fetched from *Snort* website [6].
46
47
48
49

50
51 • *Vulnerability Scanner Update Time*

52 The utility of the NIDS when used in combination with other tools is
53 addressed by the analysis of the *update delay* between the vulnerability
54 scanner plugins and the corresponding NIDS rules. The *Nessus* vulner-
55 ability scanner has been considered as the reference VDS as it is widely
56
57
58

used by security administrators to audit and verify the existence of vulnerable network services. Each vulnerability is detected by a plugin, whose release date is used to calculate the NIDS *update delay*.

- *Exploit Release Time*

The exploit release date has been used to characterize the update time response of the NIDS in relation to the publication of new exploits. Although not for every entry, vulnerability definitions in the *Open Source Vulnerability Database* include the date when an exploit that can take advantage of certain vulnerable software is made public.

3.3. *Experimental setup*

The *Snort* intrusion detection system is a signature-based NIDS able to capture and analyze traffic in IP networks in real time. It was first developed and released by M. Roesch in 1998 [30] and has become a standard in open source intrusion detection. Although self-written rules can be added by anyone, the default set of rules of the engine is developed by the *Snort* Vulnerability Response Team (VRT). All rules are made available within packages released with no periodicity. Paying subscribers can access these packages as soon as they are released while registered users must wait 30 days after their initial release to obtain them free of charge. The *Snort* VRT classifies rules as *low*, *medium* or *high* depending on their risk level. In this paper, every available type of rule is used to compute delays and no qualitative distinction is made between them. In packages, rules are also categorised as *updated* or *new*. *Updated* rules are modified in order to detect new attacks to known vulnerabilities while *new* rules are released after a new vulnerability is disclosed. Information about an attack to a known vulnerability can be included as part of an existing vulnerability definition or result in a new CVE or Bugtraq ID. Thus, it is not possible to correlate an updated rule to a specific vulnerability as no further information is provided by *Snort*. In order to ensure that a rule can be correlated with a specific vulnerability, only *new* rules are considered. The rules this study is based on, are new detection rules released between November 6th, 2007 and August 25th, 2010, whose description is available at *Snort* website [6].

We have built several *Python* scripts in order to gather information about a significant number of signatures and vulnerabilities. Figure 2 presents a diagram of the entire setup. Information retrieval scripts, which can be run at any time to update the analyzed data, allow us to obtain information

from the different sources. The OSVDB vulnerability definitions include several references to external databases and security tools. These are used to establish the exact *Snort* rule, CVE, Bugtraq ID and Nessus plugin ID that is related to the same vulnerability. The OSVDB provides itself some methods to be exported to a local database, thus correlations between *Snort* IDs and other references are obtained through the appropriate queries. Our results takes into account all data included in this database up to September 15th, 2010.

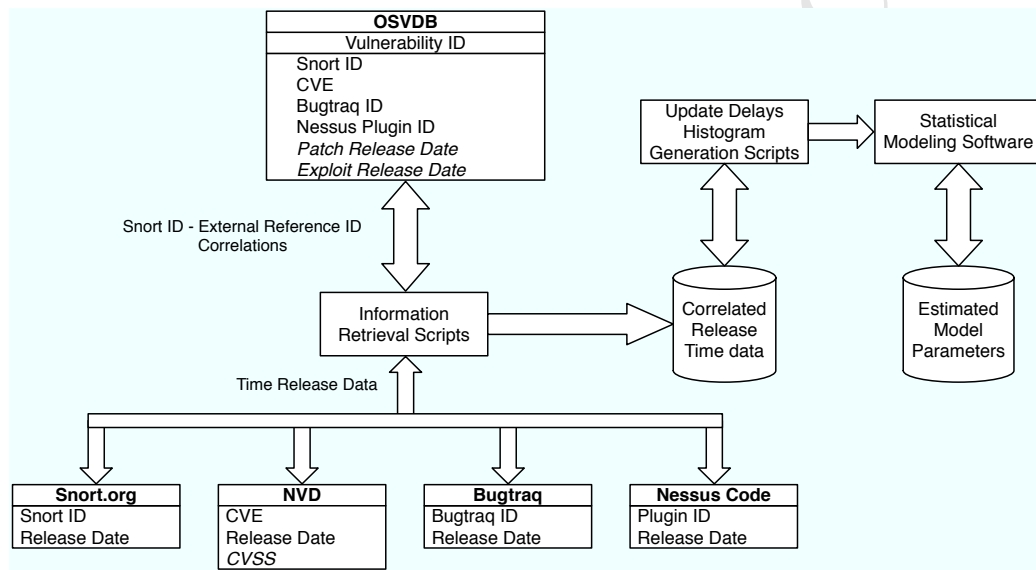


Figure 2: Complete experimental setup. Information retrieval scripts gather and correlate data from different sources. Correlated data is used to compute delays and their cumulative distributions. Finally, statistical models are adjusted using statistical software [2].

NIDS rule release dates are obtained from *Snort* advisories site. This source does not provide any method to access information about rules and release dates in a formal manner, thereby it is necessary to parse available web data as long as it remains structured. Previous information to November 2007 is not structured and is, therefore, not parseable. Release dates of CVE and Bugtraq IDs are gathered from their respective web databases. Information regarding all vulnerabilities is well structured and can be easily retrieved. In addition, *Nessus* plugins release dates have been obtained straight from their code using a script built for that purpose. Software patches and ex-

Simultaneously Included References	OSVDB Vulnerability Definitions
<i>Snort ID</i> & CVE	10280 vulnerabilities
<i>Snort ID</i> & Bugtraq ID	7575 vulnerabilities
<i>Snort ID</i> & Software Patch	4729 vulnerabilities
<i>Snort ID</i> & Nessus Plugin	37839 vulnerabilities
<i>Snort ID</i> & Exploit	4130 vulnerabilities

Table 1: Each value represents the available number of vulnerabilities in the OSVDB that include an associated *Snort ID* and another reference. The total number of gathered dates is lower than available correlations in OSVDB as not all *Snort ID* rule release dates are available from *Snort* site.

exploits release dates have been obtained from the OSVDB. Table 1 shows a summary of the vulnerability definitions available in the OSVDB including an associated *Snort ID* and other external reference.

Once that release time data from all sources is properly correlated, a script is used to calculate the cumulative distributions of the analyzed variables. These histograms are the input of the statistical modelling software used to estimate probability distributions of the *update delays* and their numerical parameters.

Figures 3 and 4 present the pseudocode of the information retrieval scripts that have been used to obtain relevant information from the different data sources. Figure 5 presents the pseudocode of the *update delay* histogram generation script. Correlated release time data is used to compute delays and calculate the cumulative probability distributions. All scripts are available from: <http://www.lab.inf.uc3m.es/~adiaz/NIDSupdatedelays.zip>.

```

10 WHILE OSVDB.vulnerabilityID NOT null
11   READ OSVDB.vulnerabilityID
12   IF OSVDB.vulnerabilityID.snortID NOT null AND OSVDB.vulnerabilityID.extReferenceID NOT null
13     READ OSVDB.vulnerabilityID.snortID
14     READ OSVDB.vulnerabilityID.extReferenceID
15     READ snortSite.snortID.releaseDate
16     READ externalDatabase.extReferenceID.releaseDate
17     CorrelatedTimeData = (snortID, snortIDReleaseDate, extReferenceID, extRefIDReleaseDate)
18     WRITE CorrelatedTimeData

```

Figure 3: Pseudocode of information retrieval scripts for CVE, Bugtraq and *Nessus* release data.

```

25 WHILE OSVDB.vulnerabilityID NOT null
26   READ OSVDB.vulnerabilityID
27   IF OSVDB.vulnerabilityID.snortID NOT null AND OSVDB.vulnerabilityID.exploitDate NOT null
28     READ OSVDB.vulnerabilityID.snortID
29     READ OSVDB.vulnerabilityID.exploitDate
30     READ snortSite.snortID.releaseDate
31     CorrelatedTimeData = (snortID, snortIDReleaseDate, exploitReleaseDate)
32     WRITE CorrelatedTimeData

```

Figure 4: Pseudocode of information retrieval scripts for Software patches and exploit release data.

```

40 WHILE CorrelatedTimeData NOT null
41   READ CorrelatedTimeData
42   delayID = snortID.releaseDate - externalReferenceID.releaseDate
43   delayIDVolume = COUNT delayID
44   WRITE (delayID,delayIDVolume)

```

Figure 5: Pseudocode of *update delay* histogram generation scripts.

4. Results & Data Modeling

In this Section, the *update delays* calculated from each of the analyzed variables are statically modeled and numerical results are presented. The measures directly obtained by the scripts have been depicted in several histograms allowing us to estimate the best fitting probability density function and its parameters using the Kolmogórov-Smirnov test [24]. Table 2 shows how measures are calculated. The t_{item} –with $item \in \{CVE, Bugtraq, patch, plugin, exploit, snort\}$ – indicates the release date of the corresponding *item*. According to these definitions of *update delays*, a negative delay implies that the *Snort* rule is released before the corresponding *item* is made public. Hence, positive delays are measured when the *Snort* rule is released after the event of reference. The higher this delay becomes, the higher is also the risk that a successful attack to a network service goes undetected.

Measure	Definition (days)
NIDS-CVE <i>update delay</i>	$t_{snort} - t_{CVE}$
NIDS-Bugtraq <i>update delay</i>	$t_{snort} - t_{bugtraq}$
NIDS-Patches <i>update delay</i>	$t_{snort} - t_{patch}$
NIDS-Nessus <i>update delay</i>	$t_{snort} - t_{plugin}$
NIDS-Exploits <i>update delay</i>	$t_{snort} - t_{exploit}$

Table 2: Definitions of *Snort* NIDS *update delay* in relation to related security events.

4.1. NIDS Update Delay from Vulnerability Disclosure

Figure 6 represents two estimations of the probability distribution of the *Snort* rule updating process. In both cases, when CVEs and Bugtraq IDs are established as time references for the disclosure of new vulnerabilities, it can be stated that most detection rules are released within the first 100 days. In order to thoroughly characterize both distributions, Figures 7 and 8 show the specific parameters for each estimation.

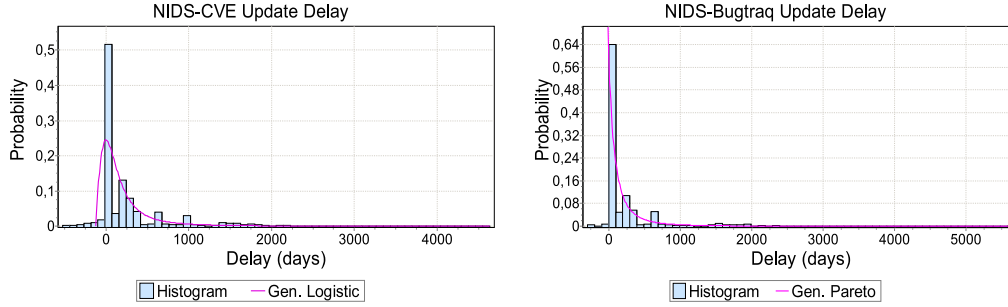


Figure 6: Probability distributions of *Snort update delay* vs. CVE and Bugtraq IDs publication dates. A positive delay reflects that a *Snort* rule is released after the related reference is published in the corresponding database.

Parameter	Value
κ	0.50988
σ	115.41
μ	100.26

$$f_{CVE}(x) = \begin{cases} \frac{(1+\kappa z)^{-1-1/\kappa}}{\sigma(1+(1+\kappa z)^{-1/\kappa})^2} & \kappa \neq 0 \\ \frac{\exp(-z)}{\sigma(1+\exp(-z))^2} & \kappa = 0 \end{cases}$$

Figure 7: Parameters and probability density function of the *generalized logistic distribution*, the best fitting distribution to the *update delay* between *Snort* rules and CVEs according to Kolmogórov-Smirnov test.

Parameter	Value
κ	0.54174
σ	110.79
μ	-30.173

$$f_{Bugtraq}(x) = \begin{cases} \frac{1}{\sigma} \left(1 + \kappa \frac{x-\mu}{\sigma}\right)^{-1-1/\kappa} & \kappa \neq 0 \\ \frac{1}{\sigma} \exp\left(-\frac{x-\mu}{\sigma}\right) & \kappa = 0 \end{cases}$$

Figure 8: Parameters and probability density function of the *generalized Pareto distribution*, the best fitting distribution to the *update delay* between *Snort* rules and Bugtraq IDs according to Kolmogórov-Smirnov test.

4.2. NIDS Update Delay from Software Updates

Figure 9 represents the probability distribution estimation of the time interval between the release of the *Snort* rule and the release of the corresponding security patch capable of fixing the related vulnerable service. The obtained distribution, centered around positive delay values, suggests that the response of the NIDS updating process is slightly slower than vendors patching release. In order to thoroughly characterize the distribution, the specific parameters for the estimation are presented.

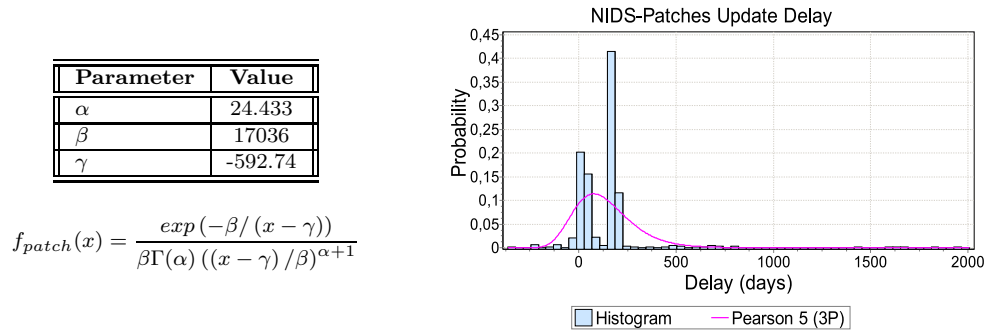


Figure 9: Estimated probability distribution of *Snort* update delay vs. patch release. Parameters and probability density function of the *three-parameters Pearson 5 distribution*, the best fitting distribution to this measure according to Kolmogórov-Smirnov test.

4.3. NIDS Update Delay from Vulnerability Scanner

Figure 10 represents the probability distribution estimation of the time interval between the *Snort* rule update, which is able to detect an attack to a vulnerable service, and the release of *Nessus* plugin able to detect the corresponding vulnerable service. The obtained distribution, centered around positive delay values, suggests that the response of the NIDS rule update process is typically slower than the release of new *Nessus* plugins when a new vulnerability is disclosed. In order to thoroughly characterize the distribution, the specific parameters for the estimation are presented.

Parameter	Value
α	9.3016
β	4728.4
γ	-356.16

$$f_{Nessus}(x) = \frac{\exp(-\beta/(x-\gamma))}{\beta\Gamma(\alpha)((x-\gamma)/\beta)^{\alpha+1}}$$

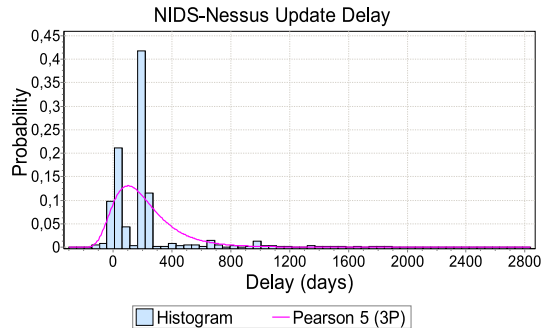


Figure 10: Estimated probability distribution of *Snort update delay* vs. *Nessus* plugins release. Parameters and probability density function of the *three-parameters Pearson 5 distribution*, the best fitting distribution to this measure according to Kolmogórov-Smirnov test.

4.4. NIDS Update Delay from Exploit release

Figure 11 represents the probability distribution estimation of the time interval between the release of a functional exploit that take advantage of a vulnerability and the publication of the *Snort* rule capable of detecting such specific attack. The obtained distribution is centered around positive delay values and increasing probabilities around 0 days. It suggests that the exploit development is based on the information gathered from the disclosed vulnerability while the NIDS rule is created in order to detect an specific exploit after it has been made public. Therefore, the response of the NIDS update process seems slightly slower than attackers exploit development. In order to thoroughly characterize the distribution, the specific parameters for the estimation are presented.

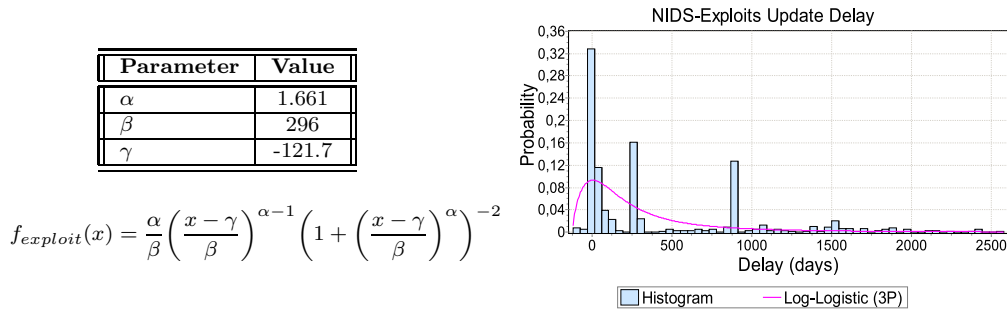


Figure 11: *Snort update delay* vs. exploit publication dates estimated probability distribution. Parameters and probability density function of the *three-parameter log-logistic distribution*, the best fitting distribution to this measure according Kolmogórov-Smirnov.

4.5. Discussion

Statistical values from every estimated distribution are shown in Table 3. This let us draw conclusions about the NIDS performance against vulnerability disclosure, software patch releases, VDS updates and exploit publication timing. The *Standard Error* column provides a quality index of the statistical results.

Measure	Sample Size	Average	Median (50%)	Standard Error
NIDS-CVE <i>update delay</i>	3032 delays	236.66 days	27 days	7.8336 days
NIDS-Bugtraq <i>update delay</i>	2270 delays	211.6 days	19 days	9.1755 days
NIDS-Patches <i>update delay</i>	1730 delays	138.35 days	160 days	5.173 days
NIDS- <i>Nessus update delay</i>	15038 delays	221.68 days	216 days	2.3064 days
NIDS-Exploits <i>update delay</i>	658 delays	402.53 days	145.5 days	21.537 days

Table 3: Statistical information of the NIDS rule *update delay* distribution for each analyzed measure. All displayed values are in days excluding the sample size, namely the total number of delays used for the estimation.

The column labeled as *Sample Size* shows the total number of delays that have been used to generate histograms for each measure. The total number of *Snort* rules release dates retrieved from *Snort.org* is 4411. In most cases, a detection rule refers to a single vulnerability and only one correlation with a single reference is possible. A single delay value is thus calculated. Nevertheless, in a few occasions, a single rule can refer to two or more vulnerabilities, VDS plugins, patches or exploits. In this case, several

delays can be calculated. Sample size of *Nessus* plugins measure counts up to 15038 when only 4411 *Snort* rules have been used for the computation. As mentioned, several *Nessus* plugins refer to the same *Snort* rule. Therefore, one *Snort* rule can detect attacks to several vulnerabilities that are discovered by different *Nessus* plugins and, according to this, the rule presents several independent delays.

The column labeled as *Average* shows the average time in days that it takes researchers to release a rule after the corresponding reference is made public. The NIDS rules present the highest average *update delay* on exploit publications. Therefore, if CVE and Bugtraq release dates are considered as the vulnerability disclosure date, it can be stated that most exploits are developed before the analyzed vulnerability becomes public. More specifically, Table 4 shows the percentage of NIDS rules that are released faster than the corresponding reference. The ratio of negative, null and positive *update delays* is shown for each case.

Measure	Negative Delays	Null Delays	Positive Delays
NIDS-CVE <i>update delay</i>	6.46%	6.4%	87.13%
NIDS-Bugtraq <i>update delay</i>	1.14%	2%	96.85%
NIDS-Patches <i>update delay</i>	4.57%	2%	93.43%
NIDS- <i>Nessus</i> <i>update delay</i>	3.26%	0.56%	96.17%
NIDS-Exploits <i>update delay</i>	3.49%	0.61%	95.89%

Table 4: Percentage of negative, null and positive delays for each measure.

Median values in Table 3 are the center of the distribution. The median becomes significant in cases like these where very high delay values have a strong influence on the average. When measuring the time interval between the disclosure of a vulnerability and the publication of the associated rule, we have observed that some specific rule updates are released very late, specially when a new form of attack is discovered for an already known old vulnerability. Table 5 shows an example of a detection rule with an extremely high *update delay*. The vulnerability identified as OSVDB ID 1166 was disclosed in 1999 according to the creation date of both related CVE and Bugtraq ID. It refers to a remote input validation error in the Microsoft NT Local Security Authority (LSA), causing a denial of service (DoS). A detection rule for the attack using UDP packets was shortly released. In 2009, the same vulnerability was openly proved to be exploitable by TCP packets and a new detection rule was released. As a result, a potentially successful attack has

not been detected by the NIDS during almost 10 years. This extreme high delays values tend to shift the average of the *update delay* distribution, thus the 50th percentile seems more significative and allows us to gain more information in order to build security metrics. Nevertheless, we consider that the occurrence of this high delays should arouse serious concerns, not only in NIDS developer teams but also in the research community.

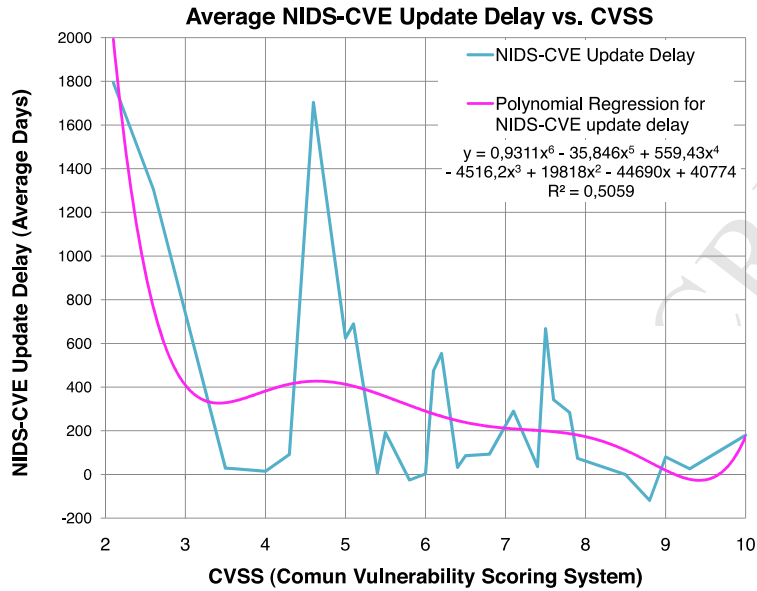
OSVDB ID	Disclosure Date	Snort ID	Rule release date	update delay
1166	1999-12-16	529	Unknown	Unknown
1166	1999-12-16	15448	2009-04-08	3401 days

Snort ID	Description
529	NETBIOS DCERPC NCADG-IP-UDP srsvnc NetrShareEnum null policy handle attempt
15448	NETBIOS DCERPC NCACN-IP-TCP srsvnc NetrShareEnum null policy handle attempt

Table 5: Example of *Snort* rule released with a extremely high *update delay*.

The obtained NIDS *update delays* have been correlated to the corresponding CVSS index in order to characterize how the NIDS update process is influenced by the severity of disclosed vulnerabilities. This index categorizes severity from 1 to 10, being 10 the most severe case. Figure 12 shows averaged NIDS *update delays* from CVE release for vulnerabilities with the same CVSS. NIDS-CVE Update Delay has been modeled using a 6th order polynomial regression in order to avoid overadjustment to data but obtaining a coefficient of determination $R^2 = 0.5059$. Figure 12 shows how the *update delay* decreases for higher risk vulnerabilities.

Therefore, when a new vulnerability is disclosed and a CVE is created, the associated CVSS index can be used by security officers to estimate the expected average *update delay* of the deployed NIDS. The estimated regression used in Figure 12 and formalised by its coefficients in Table 6 will provide a numerical delay in days that can be taken into account when following a strategy to protect vulnerable systems. Table 7 shows the estimated values from sample data that have been used to derive the statistical model.



30 Figure 12: Average NIDS *update delay* from CVE release vs. Common Vul-
31 nerability Scoring System index.
32
33

34

NIDS-CVE Update Delay Polynomial Regression Coefficients						
0	1	2	3	4	5	6
40774	-44690	19818	-4516.2	559.43	-35.846	0.9311

35
36
37
38

39 Table 6: Estimated coefficients for the polynomial regression adjusted to the
40 non-linear relationship between CVSS and NIDS-CVE Update Delay.
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

Average NIDS-CVE Update Delay vs. CVSS			
CVSS	Avg. Update Delay	CVSS	Avg. Update Delay
2,1	1793 days	6,4	32,02 days
2,6	1305 days	6,5	86,29 days
3,5	29 days	6,8	93,12 days
3,6	27 days	7,1	289,67 days
4	14,88 days	7,4	36 days
4,3	91,88 days	7,5	667,49 days
4,6	1703,6 days	7,6	341,86 days
5	624,45 days	7,8	283,26 days
5,1	689,6 days	7,9	73,5 days
5,4	7 days	8,5	-1 days
5,5	191,67 days	8,8	-118,67 days
5,8	-25,19 days	9	80,14 days
6	2,67 days	9,3	26 days
6,1	475 days	10	179,82 days
6,2	554 days		

Table 7: NIDS update delays on average from the CVE and the corresponding CVSS obtained from the total number of vulnerabilities used as sample data.

The perception of security in corporate environments is strongly related to the NIDS performance. As stated in the ISO/IEC 27005 code of practice for information security risk management from the *International Organization for Standardization* (ISO) [8] and the *International Electrotechnical Commission* (IEC) [7], quantitative risk estimation is based on a numerical calculation according to security metrics on the asset. Signature-based NIDS metrics are typically based on detection rates and ease of management. Nevertheless, the NIDS updating process must be taken into account in combination with patching policies and other security tools. The experimental setup presented in Section 3 and the formal model derived from data in Section 4, provide a way to automatize the process of keeping numerical results up to date. Therefore, the estimated statistical distributions of these delays can be periodically updated and used to quantitatively evaluate security risk associated to NIDS in terms of its updating process.

5. Conclusions

In this paper we have characterized the time response of the *Snort* rule release process. This task has been done through the comparison of its update time versus several related security events such as vulnerability disclosures, software security patch releases, *Nessus* plugin releases and exploit publications. The time interval between each of these events and the release of

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

NIDS rule updates have been defined as different *NIDS update delays*. To the best of our knowledge, this is the first comprehensive study that statistically analyzes signature-based *NIDS update delays*. Specific conclusions are drawn from the shape of the estimated probability distributions. When the vulnerability disclosure date is established as the reference event to calculate how fast the NIDS is updated, most of the NIDS rule updates are released within the first 100 days. The positive profile of the distributions and the little number of negative *update delays* let us state that the *Snort* NIDS is certainly a reactive technology.

The performance of the NIDS updates related to the software patches have also been measured. From the obtained results, we conclude that *Snort* updates slightly slower than patches are released. Nevertheless, a non-negligible number of negative delay values have been measured, meaning that some detection rules are released before the software patch. This makes this NIDS an effective tool when it comes to alerting the administrators of attacks targeting not yet patched vulnerable services. As stated in [11] and [23], security administrators are not always able to immediately deploy software updates in every system but attacks do increase considerably with their release. Despite the time it takes researchers to release a specific detection rule, the updated NIDS can prevent vulnerable services to be compromised until new patches are completely deployed or vulnerable systems are removed from the network.

The straight conclusion obtained from the analysis of the *NIDS update delay* from *Nessus* plugin releases is that the VDS has a faster response to the disclosure of vulnerabilities. Hence, in a properly designed security policy, the VDS should be used as the primary tool to early detect vulnerable services and consequently, take actions to harden systems. Signature-based NIDS can also alert from a vulnerability when an attack targeted to a service is detected but not as early as the VDS. As a matter of fact, new *Nessus* plugins are released almost every day, unlike *Snort*, whose updates are always done through packages of rules released with no periodicity.

Although the release of NIDS rules seems slower than publication of exploits, the maximum of the distribution is around zero, meaning that most of them are made public almost at the same time. This suggests that the development process begins for both elements after the vulnerability is disclosed but in many cases, the detection rule is based on the exploit definition and not in the vulnerability. An interesting conclusion from the average *NIDS update delay* on exploits in comparison with CVE or Bugtraq *update delay*,

1
2
3
4
5
6
7
8
9 is that many exploits are developed and released before the vulnerability is
10 disclosed.

11 The relationship between NIDS *update delay* and the severity of vulner-
12 abilities has also been analyzed. CVEs and their CVSS severity index have
13 been used in order to quantify the risk of the corresponding failure. Results
14 show that when the CVSS of vulnerabilities approaches 10 (i.e. the maxi-
15 mum), the average NIDS *update delay* decreases. We consider this a positive
16 result that implies that NIDS rules researchers are aware of the potential
17 impact of vulnerable services. Therefore, detection rules protecting high risk
18 vulnerable services are released faster. According to the estimated regression
19 of the NIDS update delays in relation to the severity of the disclosed vulner-
20 ability, some implications in the context of security policy definition and de-
21 ployment have been drawn. Specifically, we provide a way to approximately
22 quantify, the NIDS *update delay* (in days) depending on the vulnerability
23 severity (CVSS). This model and the practical consequences of the results
24 presented can help network security administrators to develop more effective
25 strategies in the regular vulnerability disclosure scenario.
26
27
28
29
30
31

32 **6. Future Work**

33 Lippmann and Webster [22] tried in 2002 to determine the role of the
34 NIDS and its performance against the VDS and the patching process tim-
35 ing. Although our findings agree with some of their conclusions, the size of
36 the analyzed sample and the statistical model we present represent a quan-
37 titative and a qualitative improvement. According to our results and their
38 implications, we encourage to build a security system that gathers infor-
39 mation from the services the organization is running and checks for public
40 vulnerabilities of those services as well, reporting if there is any software or
41 NIDS update to mitigate those vulnerabilities. This kind of system, which
42 should continuously perform passive traffic analysis to identify active network
43 services, would help to reduce the time the system is exposed. This would
44 allow system administrators to take action in order to mitigate the effects
45 of possible attacks even if there is no published solution to the vulnerabil-
46 ity. Well established correlations between vulnerabilities, detection plugins
47 and signatures are needed. A proposal in this direction was presented in
48 2005 [25]. Massicotte et al. tried to correlate *Snort* detection signatures
49 with *Nessus* vulnerability detection plugins but authors were not successful
50 as the information available from different databases was insufficient at that
51
52
53
54
55
56
57
58

1
2
3
4
5
6
7
8
9 time. Nowadays, towards an effort to improve their security tools manage-
10 ment, vendors have better structured their databases and some *open source*
11 initiatives like the OSVDB are becoming a worldwide available repository
12 of vulnerability information. In our work, we have been able, in the worst
13 case, to correlate up to 4130 vulnerability entries (in a 2 years, 9 months and
14 19 days period). Recently developed *Sourcefire RNA (Real-time Network*
15 *Awareness)* [9] commercial tool also focuses on passive network discovery
16 and targeted vulnerability assessment. However, as it is based on *Snort IDS*,
17 *update delays* are, unfortunately, exactly those described in this work.
18
19
20
21
22
23

24 References

- 25
26 [1] Bugtraq vulnerability database. 2010. <http://www.securityfocus.com/archive/1>.
27
28 [2] Easyfit distribution fitting software. 2010. <http://www.mathwave.com>.
29
30 [3] National vulnerability database. 2010. <http://nvd.nist.gov>.
31
32 [4] Nessus vulnerability scanner. 2010. <http://www.nessus.org>.
33
34 [5] Open source vulnerability database. 2010. <http://www.osvdb.org>.
35
36 [6] Snort intrusion detection system. 2010. <http://www.snort.org>.
37
38 [7] International electrotechnical commission. 2011. <http://www.iec.ch/>.
39
40 [8] International organization for standardization. 2011. <http://www.iso.org/>.
41
42 [9] Sourcefire rna (real-time network awareness). 2011. <http://www4.sourcefire.com/products/3D/rna>.
43
44 [10] Aickelin, U., Twycross, J., Roberts, T.. Rule generalisation in intrusion
45 detection systems using snort. *International Journal of Electronic Security and Digital Forensics* 2007;1:101–116.
46
47 [11] Arora, A., Nandkumar, A., Telang, R.. Does information security
48 attack frequency increase with vulnerability disclosure? *An empirical analysis*. *Information Systems Frontiers* 2006;8(5):350–362.
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

- 1
2
3
4
5
6
7
8
9 [12] Biermann, E.. A comparison of Intrusion Detection systems. *Computers*
10 & *Security* 2001;20(8):676–683.
11
- 12 [13] Cavusoglu, H., Cavusoglu, H., Raghunathan, S.. Efficiency of vulner-
13 ability disclosure mechanisms to disseminate vulnerability knowledge.
14 *IEEE Transactions on Software Engineering* 2007;33(3):171–185.
15
- 16 [14] Chen, Z., Delis, A.. A pragmatic methodology for testing intrusion
17 prevention systems. *The Computer Journal* 2008;52(4):429–460.
18
- 19 [15] Chen, Z., Liu, Q., Lin, S.. Research on Evaluation Method of Intrusion
20 Detection System. In: *Proceedings of the 2nd International Conference*
21 *on E-business and Information System Security*. IEEE; 2010. p. 1–4.
22
- 23 [16] Flowers, J.. US Patent 7,509,681: Interoperability of vulnerability and
24 intrusion detection systems. 2009.
25
- 26 [17] Frei, S., May, M., Fiedler, U.. Large-scale vulnerability analysis. In:
27 *SIGCOMM workshop on Large-scale attack defense*. 2006. p. 131–138.
28
- 29 [18] Frei, S., Tellenbach, B.. 0-Day Patch-Exposing Vendors (In)security
30 Performance. In: *BlackHat Europe*. 2008. .
31
- 32 [19] Gu, G., Fogla, P., Dagon, D., Lee, W., Škorić, B.. Measuring
33 intrusion detection capability: an information-theoretic approach. In:
34 *Proceedings of the 2006 ACM Symposium on Information*. 2006. p. 90–
35 101.
36
- 37 [20] Gula, R.J., Deraison, R.M.M., Hayton, M.T.. US Patent 7,761,918:
38 System and method for scanning a network. 2010.
39
- 40 [21] Kuhn, R., Johnson, C.. Vulnerability Trends: Measuring Progress. *IT*
41 *Professional* 2010;12(4):51–53.
42
- 43 [22] Lippmann, R., Webster, S.. The effect of identifying vulnerabilities
44 and patching software on the utility of network intrusion detection. In:
45 *Recent Advances in Intrusion Detection (RAID)*. Springer-Verlag; 2002.
46 p. 307–326.
47
- 48 [23] Liu, S., Kuhn, R., Rossman, H.. Surviving Insecure IT: Effective
49 Patch Management. *IT Professional* 2009;11(2):49–51.
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

- 1
2
3
4
5
6
7
8
9 [24] Massey, F.J.. The Kolmogorov-Smirnov test for goodness of fit. *Journal*
10 *of the American Statistical Association* 1951;46(253):pp. 68–78.
11
- 12 [25] Massicotte, F., Couture, M., Briand, L., Labiche, Y.. Context-Based
13 *Intrusion Detection Using Snort , Nessus and Bugtraq Databases*. In:
14 *Proceedings of the Third Annual Conference on Privacy, Security and*
15 *Trust*. 2005. .
16
17
- 18 [26] Massicotte, F., Gagnon, F., Labiche, Y.. Automatic evaluation of in-
19 *trusion detection systems*. In: *Proceedings of the ACSAC 06. 22nd An-*
20 *annual Computer Security Applications Conference*. IEEE; 2006. p. 361–
21 370.
22
23
- 24 [27] Nizovtsev, D., Thursby, M.. To disclose or not? An analysis of software
25 *user behavior*. *Information Economics and Policy* 2007;19(1):43–64.
26
27
- 28 [28] Orfila, A., Carbo, J., Ribagorda, A.. *Autonomous decision on in-*
29 *trusion detection with trained BDI agents*. *Computer Communications*
30 2008;31(9):1803–1813.
31
32
- 33 [29] Orfila, A., Tapiador, J.M.E., Ribagorda, A.. *Trends , problems and*
34 *misconceptions on testing Network Intrusion Detection Systems effec-*
35 *tiveness*; Nova Publishers. p. 51–62.
36
- 37 [30] Roesch, M.. *Snort - lightweight intrusion detection for networks*. In:
38 *Proceedings of the 13th USENIX conference on System administration*.
39 *Berkeley, CA, USA: USENIX Association; LISA '99; 1999*. p. 229–238.
40
41
- 42 [31] Schryen, G.. *A Comprehensive and Comparative Analysis of the Patch-*
43 *ing Behavior of Open Source and Closed Source Software Vendors*. In:
44 *Proceedings of the Fifth International Conference on IT Security Inci-*
45 *dent Management and IT Forensics*. Ieee; 2009. p. 153–168.
46
47
- 48 [32] Sun, M., Chen, T.. *US Patent Application 20,090/217,341: Method of*
49 *updating intrusion detection rules through link data packet*. 2009.
50
- 51 [33] Ulvila, J.W., E., G.J.. *A decision analysis method for evaluating*
52 *computer intrusion detection systems*. *Decision Analysis* 2004;1(1):35–
53 50.
54
55
56
57
58

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

- [34] Vache, G.. Vulnerability analysis for a quantitative security evaluation. In: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE; 2009. p. 526–534.
- [35] Young, G., Pescatore, J.. Gartner Magic Quadrant for Network Intrusion Prevention Systems. 2010.