Software Vulnerability vs. Critical Infrastructure - a Case Study of Antivirus Software

Juhani Eronen*, Kati Karjalainen, Rauli Puuperä
Erno Kuusela, Kimmo Halunen, Marko Laakso, Juha Röning
Oulu University Secure Programming Group
Department of Electrical and Information Engineering
P.O. Box 4500
90014 University of Oulu
Email: ouspg@ee.oulu.fi
*Finnish Communications Regulatory Authority FICORA
P.O. Box 313
00181 Helsinki
Email: juhani.eronen@ficora.fi

Abstract

During the last decade, the realisation of how vulnerable critical infrastructures are due to their interdependencies has hit home with more gravity than ever. The abundance of vulnerabilities in the software that is widely used in critical systems could have escalating consequences. In this paper, we used the PROTOS MATINE model to systematically examine the scope of software systems used in critical infrastructure. Dependency analysis methods indicated antivirus software as a critical subject to study, as its use is mandated and as it processes data from malicious sources. We determined that antivirus software is by nature susceptible to various risks and has exhibited significant vulnerability, but the issue is neither widely recognized nor reported. Awareness on the drawbacks of AV software should be spread among the planners of the critical infrastructures. Due to inherent risks, the suitability of antivirus software in critical systems should be reconsidered on a system-by-system basis.

Keywords: *Vulnerabilities, critical infrastructure, dependency analysis, antivirus software*

1 Introduction

According to NATO, critical infrastructure is defined as "those facilities, services and information systems which are so vital that their incapacity or destruction would have a debilitating impact on public and governmental security, economy, public health and safety and the effective functioning of the government" [32]. The need for critical infrastructure protection (CIP) has become paramount in recent years with the advent of new asymmetric threats, both physical and cyber.

While the physical risks have been manifested by the threats of natural disasters and terrorism, awareness of cyber risks has also been increased by cases of cascading failures in electrical networks and cases of premeditated damage by disgruntled employees. Warning signs have been raised by authorities on attacks against the supervisory control and data acquisition (SCADA) systems controlling critical systems [41]. However, there is much more to the cyber risks than SCADA.

Computerisation and ubiquitous network connectivity have been leading trends in the services of society during the last few decades. Systems comprising the critical infrastructure are no exception. Previously, control systems of critical services have been custom-designed software and hardware systems situated in dedicated networks. For reasons of synergy, efficiency and increased functionality, commercial off the shelf (COTS) hardware, networks and operating systems have frequently superseded the conventional wisdom in building critical systems. For similar reasons, control systems are increasingly interconnected with production and office networks and even the Internet.

Besides from offering a number of self-evident benefits, the transition of control systems into the realm of traditional computing predisposes critical systems to a number of risks, e.g. the growth of system complexity increases its failure

modes, and interconnections constitute vectors for attacking the system. The major risks that have been identified in CIP research are generally the loss of a major asset, or a cascading failure of multiple assets due to their interdependencies. These interdependencies are generally classified as physical, geographical, logical or cyber [35]. The cyber interdependencies are among the most complicated and varied of these interdependencies [4].

In previous research, we have created the PROTOS MATINE model [19] for deciphering technical dependencies of the critical infrastructure. The model includes reviews of specifications and other technical facts, as well as expert interviews to capture the tacit knowledge regarding the deployment and use of systems. Media and market share analyses enable prioritising of further study. Visualisation is used to present the results of the study in a quickly understandable and concise manner.

In this paper, we employ the PROTOS MATINE model to extend upon a study on a type of cyber dependencies reported in our earlier paper [2]. We analyse the cyber dependencies on multiple levels, including software vulnerability and interdependency due to factors such as data propagation and shared protocols, code, and libraries. In recent years, the use of antivirus (AV) software, whose goal is to keep malicious programs (malware) at bay, has become a widely adopted procedure among critical infrastructure systems. We selected AV software as our target of study, and explore some of their dependencies, as well as their vulnerability history. AV software vulnerabilities are not in general reported by the media, even though the number of AV vulnerabilities has expanded rapidly in recent years [40]. Although the overall vulnerability numbers seem to have decreased, the future progression of AV vulnerabilities is unpredictable.

As a part of our previous research, we scrutinised the vulnerability of AV software with a robustness test set [34]. We used a responsible vulnerability co-ordination process to ensure that any vulnerabilities found would be fixed by the vendors whose products they affect. The results of our robustness testing indicate that there is still much work to be done to counter the mounting complexity of current software. While bugs were found and eliminated, new ones continue to emerge at a constant rate. In this paper, we follow up the results of the vulnerability co-ordination process, and investigate whether affected AV vendors had actually fixed the vulnerabilities reported to them.

The current status of the AV use of software is a complex phenomenon. AV software does not automatically increase security, but may be a source of unnecessary risk, especially for information infrastructure. In addition to added complexity, dependency and vulnerability, there are issues related to vulnerability disclosure and reliability of AV software.

Recent studies on the efficiency of AV software raise concern about their effectiveness in the current threat landscape. In a test by the security company Team Cymru, only 37% of 1,066 pieces of current malware were detected by a sample of 32 antivirus software [17]. Another recent study by the AV vendor Panda Security observed the infection rate of unprotected systems at 33%, and that of protected systems at 23% [30]. The observed low detection rates combined with a mere ten percent point reduction in infection risk might not warrant for the usage of software with inherent risks.

Despite the stated problems, AV software is at present commonly considered as a basic element of safe computer use. For example, FICORA (Finnish Communications Regulatory Authority) recommends that AV software should be installed on computer systems in order to protect them from malware. HIPAA [25] and Sarbanes-Oxley Act, (SOX) [39] have extended these security requirements to laws. The same conception of security produced by AV software is popularized by security policies, user education and media. There is a considerable lack of controversial opinions in all of these areas.

The current security paradigm is the main reason for problems in the context of AV software use. Although AV software may be a necessity to fight off specific malware threats, its de facto and de jure use should be reconsidered in critical infrastructure systems. In many cases, the use of AV software may expose the system to unnecessary vulnerabilities and cause needless dependencies.

The next section presents background on the context of antivirus vulnerabilities in critical infrastructures, as well as an explanation of the PROTOS MATINE model and supplementary models for dependency analysis. The third section presents the results gathered by the dependency analysis and the ensued robustness tests. The paper concludes with observations on the results and on areas of future improvement.

2 Background

In this section, we present a definition for vulnerability, and list the unique aspects of AV software with respect to vulnerabilities. Next, we define our concept on different levels of technical dependency, and how it can be used to augment dependency analysis in traditional CIP perspective. This is followed by an explanation of the PROTOS MATINE model, as well as a short review of complementary dependency analysis methods.

2.1 Vulnerabilities and Antivirus Software

All software contains bugs due to various factors, such as inherent difficulty in translating the requirements to code, complexity of the requirements or the underlying system, immature programming practices and methods [21, 6]. An old maxim of the quality control industry states that the number of flaws in a system is generally proportionate to the complexity of the system. This can be restated as "the number of flaws in a system is roughly proportionate to the extent of its functionality". All modern COTS software systems are very complex, which raises the number of their bugs to towering amounts. Bugs with security implications are called vulnerabilities.

Although AV software is commonly thought to increase security, it is produced by the same programming processes, which can result in insecure programs in general. As a rule, all software is breakable [6]. By definition, AV software must process potentially malicious input in a wide variety of data formats. As parsing different protocols and formats have historically been proven error-prone, AV software is particularly susceptible to programming errors.

Current malware employs a variety of methods to thwart detection, such as packing, polymorphism, obfuscation, anti-analysis and anti-unpacking. Some malicious code has even been reported to exploit vulnerabilities in analysis software. The defence methods force AV systems to employ emulation, deobfuscation, unpacking, and other functions in order to successfully detect malware. As these operations are very sophisticated and handle potentially malicious input, the error-sensitivity of AV software is further highlighted.

Many AV software share the same integral scanning engine or engines [8, 46]. The scanning engine, responsible for identifying malicious files using signature databases, is the main component of an AV software. Homogeneity facilitates the design process of malware, as it is relatively quick to test the malware in development with all of the most common AV software [44]. This may be reflected in the recently observed low detection rates of current malware [17, 30].

AV software population is quite homogeneous, which in itself is a warning sign: it enables the spread of malware [5] that targets against dominant AV products. The market is dominated by a few leading vendors and using more than one AV program at a time is usually impossible [23], which may be fortunate since each vulnerable AV program would add to the attack surface by exposing more code to exploitation attempts. AV software requires high access rights in order to monitor the system, which makes them attractive attack vectors for systems compromise.

2.2 Dependencies and Critical Infrastructure in the Antivirus Vulnerability Context

During the last decade, the importance of critical infrastructure has been realised more acutely than ever. Dependencies between different infrastructures have been recognised as a major cause for escalating consequences for errors in point components. In critical infrastructure, dependencies can be identified on multiple levels, including technology, functions, people, processes and location. Failures of infrastructure components have been identified to induce immediate or delayed problems or failures in dependent components, which may in turn lead to cascading failures. Electricity and energy in general are prime examples of this behaviour, as practically all other infrastructure elements depend on these. Thus, different dependency tracking models have increasingly been taken into use in the context of critical infrastructure. A good rundown of these models is the CRN International CIIP Handbook, which presents national policy approaches to critical information infrastructure protection and the methods and models used to assess the vulnerability and security of these structures [26].

In this paper, a dependency is defined as a linkage between entities or common metadata among them. Dependencies are discovered by forming descriptive metadata and links from given information and then analysing common features and differences of this semantic data. As an example in the critical infrastructure context, the dependency of a communications network on electricity could be portrayed as a link between a power plant and a cell phone tower, whereas their location in the same building could be described with location metadata containing GPS coordinates. The concepts of links and metadata can be considered equivalent to RDF [48] triplets with nodes and literals, as defined by the W3C semantic networks initiative. Similarly, the dependency graphs essentially form a semantic network. However, many of the concepts of semantic networks, such as data types and strict ontologies, have not been identified as beneficial for the rapid analysis of dependencies in previous research [21]. Thus, the approach to semantic data used in the scope of this paper is that of lightweight tagging and folksonomies rather than the stricter and more formal semantic approach.

The case presented in this paper is that the dependency of critical infrastructure components on the robustness of AV software may induce risks to those components, which may lead to wider infrastructure-level risks through cascading failures. Measuring the threat that software constitutes is quite impossible without understanding it in a detailed level. Identifying the technical dependency of software may serve as a decent first aid for this purpose, while bestowing multiple benefits such as increased understanding of the ac-

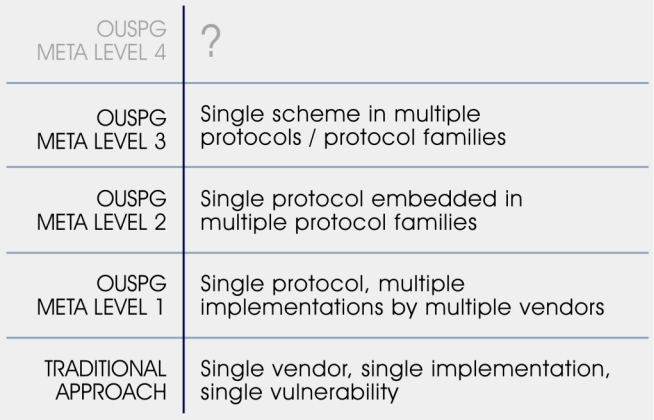

Figure 1. OUSPG metalevels

tual reason for and the scope of different kinds of failures.

Technical dependencies span multiple levels of abstraction, and thus, need to be examined in an iterative fashion. First, the boundaries of different software systems and their interfaces are enumerated, contributing to the basic understanding of the composition of the system. Communication via interfaces is performed by protocols, and thus, the used communication protocols need to be identified. Analysis on the data flows of these protocols sheds light on the propagation of data among systems, and possible attack vectors. Once the critical avenues of attack are attained, the analysis can be prioritised on the code that handles them. As most current systems are modular, this analysis can be broken down further in examinations of libraries and software subsystems that handle distinct inputs, use cases, and so forth. The data gathered by this method can be used for discerning the impact of vulnerabilities in system components of different granularity.

The concept of meta levels (see Figure 1 on page 4 is applicable to any context with inherent dependencies. Meta level is an attribute of a vulnerability, which describes its level of abstraction as well as its scope. Information on the structure of different systems and their relations highlights elements, which are highly connected or common between multiple systems. Vulnerabilities in these elements are typically of a higher meta level, as they can result in epidemic failures due to their wide implementation base, or cascading effects due to the failure of a high number of dependent elements [19].

Meta level zero describes the case where a vulnerability only affects a single implementation (a software version). Meta level one vulnerabilities affect a whole class of systems (all software that implements a certain interface). Meta level two vulnerabilities affect a super-system consisting of multiple classes of systems (all software having any interface that includes a certain subsystem). Meta level three affects an element that is used for widely disparate purposes, perhaps by a great number of systems (all systems that use a certain notation, encoding, or other function) [19].

In our previous paper [2], we have given some preliminary results of our research and a brief explanation of the methods used in this research. Our research was focused on the file formats that different AV software handles, as they form a common public interface. Side-by-side comparison of the exposure of AV software to file format vulnerabilities is not straightforward, as the support for different formats varies considerably among AV software.

Uncovering dependencies in the handling of archive file formats may be difficult due to a number of implementation details. A file format implemented in two software products can cause similar but unrelated problems in them, which could constitute a dependency false positive. Files of some archive formats may embody files in other archive formats, which may lead to the use of different algorithms in different parsing implementations of these files. Analysis of cases such as these is difficult, as specifications or source codes for commercial AV software are not available.

2.3 The PROTOS MATINE Model

The research method is based on an earlier OUSPG (Oulu University Secure Programming Group) project, PROTOS MATINE. The project focused on the interdependencies of network protocols and produced the PROTOS MATINE model [19] (see Figure 2) and the semantic tool Graphingwiki [20], which are now put into use in the context of AV vulnerabilities. The model presents an iterative method for rapidly gaining an insight into a field of study.

The PROTOS MATINE model was originally developed to illustrate protocol dependencies in critical infrastructure from multiple angles. Understanding protocol dependencies has been seen beneficial for the assessment of the wider technical dependencies of infrastructure, and the impact vulnerabilities would have on it. The method was developed to collect protocol specific data, which is spread out in multiple sources, e.g. newspapers, mailing lists, technical documents, protocol specifications and experts, who have tacit knowledge of protocol usage. During the development of the model, we noticed that tracking only one or two sources gives a biased picture of protocol's history, usage and prevalence, and by combining several data gathering methods, the accumulated data coincides better with the real situation.

The different data sources, such as specifications, literature, media and experts, work towards a common goal - understanding a technological subject on multiple levels. These levels include contents and structure of the subject, its history as well as projected future, its use cases and areas

of usage and its environment and relations to other subjects. With this kind of knowledge, the weight of a subject can be determined in the desired context, such as a system, a network, a corporation or a sector of the critical infrastructure. As an example, various data gathering methods were used to perform analyses of the effects of vulnerabilities found in parsers of the prevalent ASN.1 notation [21, 19]. The analyses were conducted with heavy emphasis on systems used in critical infrastructures, and resulted in a number of test suites to test the robustness of different protocol implementations [19].

The results of the PROTOS MATINE model are presented by visualisations that aim to portray different aspects of the protocol. Visualisations were also used as a communication method between researchers, managers and other operatives. The first views that we created, depicted the protocol's specification history (protocol view), its technical linkage (technological view), and its usage scenarios or general usage in different sectors of society (organisational view). These views were constructed using various data sources and methods: experts (interviews), public attention (media follow-up), protocol definitions (standards, technical specifications) and the prevalence of protocol implementations (historical data, usage environments). The main views were adapted according to a specific target group or usage scenario. However, we quickly discovered that more versatile and automatically generated views were needed.

We started to develop Graphingwiki, a semantic wiki tool that enables the deepened analysis of the Wiki data, by augmenting it with semantic data in a simple, practical and easily usable manner. Graphingwiki can be used to automatically present the semantic data as tables and to visualise it as graphs. These visualisations are used to clarify the resulting body of knowledge so that only the essential information for a usage scenario is displayed [21]. The key aspects of the workflow are automated gathering of baseline data, augmenting the data by experts and manual data gathering, and generating automated visualisations.

AV software was selected as our target because such software has an extensive attack surface due to a wide variety of file formats it must handle, they are run with high privileges, and their usage is mandated in many cases. Vulnerable AV software would be tempting attack vectors for systems compromise. We wanted to visualise AV vulnerabilities and, with the help of the dependency graphs, find out if there are any linkages between file formats, AV vendors and software vulnerabilities. Preliminary results helped us to focus PROTOS Genome robustness test set on archive formats and offered a context for the vulnerability co-ordination process.

In the context of AV software, vulnerability databases represent the main data sources of the PROTOS MATINE model. Media tracking and review of the market situation were performed in the year 2006 and the following results were also represented in the previous paper [2]. Media tracking and review of the market situation lay out the priorities of later data gathering and the relative importance of different AV software. Expert interviews and publicly available specifications were only used to discern the usage of archive formats.

The semantic information on AV vulnerabilities, for example impact type and file format, was gathered from the U.S. National Vulnerability Database (NVD) [31]. NVD's descriptors of vulnerabilities are categorised and presented a in specified standard format. Additional information was gathered as a media follow-up, which was focused to national level. The media follow-up consisted of regular observation of Digitoday Finland [16], commercial news database focusing on IT sector, throughout the year 2006. News considering AV issues was classified and analysed with content analysis. The focus of media follow-up was on how the AV software and vendors are presented in the media.

2.4 Previous Work

Dependency analysis methods span disparate fields, such as graph theory, social network analysis, computing and natural language processing. Some methods of relevant fields are examined in the light of the PROTOS MATINE model, and their suitability for use in the context of antivirus software is evaluated.

In graph theory, dependencies are naturally defined by the links between nodes in the graph. The links usually do not have other attributes than their direction and possibly a numeric value signifying the strength of the link. Graph theory analyses graphs with measures such as cliques, connectedness and centrality [15]. Social network analysis is a closely related field that studies specifically graphs representing relations among people. The basic realisation behind social networks is that weak ties in social networks are more significant that stronger ones. Many sophisticated analysis methods have been developed in this field [37]. However, efficient use of these approaches requires research on which analysis methods and aspects of graphs are the most relevant in the desired context.

Conceptual graphs extend the basic graph model by introducing attributes to the dependencies, which are defined as links between dependent and antecedent [13]. The conceptual graphs model attempts to form a generalised ontology of dependencies, i.e. the set of attributes that apply to every dependency regardless of context: sensitivity, stability, need, importance, strength and impact. The model is very versatile, but its rigorous definition of dependency may represent a hindrance rather than an aid in the context of rapid knowledge discovery. It is also noteworthy that the conceptual graphs model considers only the attributes

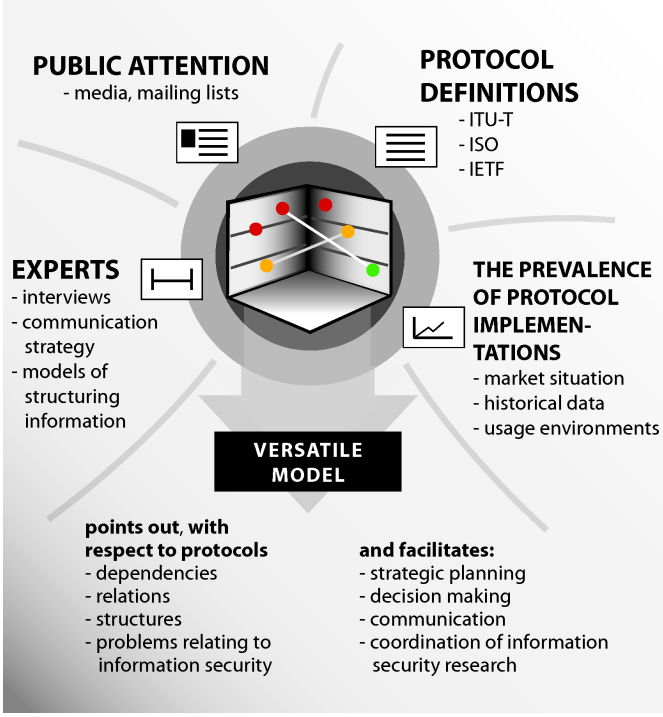

Figure 2. PROTOS MATINE model

of nodes as a source of dependencies, which may limit its usefulness.

There has been a growing interest towards dependency analysis in recent years in critical infrastructure management. The papers on the subject range from studies on the effect of a single event on one part of critical infrastructure [27] to critiques of policies adopted by a whole nation state [24]. We will describe some of the analysis methods that we consider relevant for the purposes of this paper. A more thorough review on the state of the art in critical infrastructure protection can be found in [4].

The Critical Infrastructure Modelling System (CIMS) is a system and a method for modelling dependencies in critical infrastructures, and simulating related events regarding its components. The dependency types used as the systems include physical, informational, geospatial, procedural and societal dependencies. Critical infrastructure systems are modelled by graphs, where dependencies are manifested through linkage or proximity of the nodes [18]. The CIMS software visualises graphs as 3D visualisations that can be layered on, e.g. satellite images or maps. Software-aided support for what-if scenario building is mentioned as a subject for further research.

The use of intelligent software agents to integrate, model and simulate infrastructure components has been suggested in a paper by Tolone *et al.* [45]. The system proposed in this paper is in many ways quite similar to CIMS, and it also includes 3D visualisation and simulations on critical infrastructure failures. The simulations include what-if, goal-driven, probabilistic, and discovery based analyses based on events, i.e. agent state changes. The paper does not define the dependency types used in the simulation, however, simply stating that dependencies vary based on the context of the system.

Both of the systems described in the previous paragraphs, as many other systems used to model dependencies of the critical infrastructure, for that matter, are largely constrained into the physical setting and thus unusable in the AV context. For example, the use of 3D models of may work very well in the physical context, but are unnecessary or even inapplicable to many other contexts. However, many of the ideas used in the models, such as automated scenario building and node proximity as dependency, could provide benefits to use cases such as AV. Similarly, studies that include a temporal dimension to dependencies and fault propagation could be useful, e.g. in modelling attacks to vulnerable systems [36]. We have not yet observed the need for context-varied dependencies in our research, and thus, only find the agent-based approach interesting in an academic perspective.

Graph theoretical methods used in the context of critical infrastructures mostly focus on the availability, reachability or quality of service aspects of graph portraying the infrastructure. Analysis of graph properties such as topol-

ogy and node proximity can be used to aid fault simulation [38]. Fuzzy numbers can be employed to represent incomplete information about the resiliency and other properties of the nodes [14]. Complexity analysis of the graph, in its turns, can provide insight into how efficiently the graph can be traversed in the event of failures [49]. Finally, social network analysis has proven to find critical nodes in the infrastructure graph with simple centrality, degree and variance calculations [7].

All of the above methods have the same goal, namely, to present the dependencies in a highly visual and thus more human readable form than mere documents and spreadsheets. With the PROTOS MATINE method, we aim at this very same goal, and to this end, we have used the Graphingwiki visualisation tool. The nature of the PROTOS MATINE method means that in order to be able to make visualisations with diverse types of information and multiple levels of abstraction, the visualisations need to be quite simple. Many of the methods do not have support for multiple levels of abstraction, whereas with Graphingwiki, we can present visualisations from minute details (such as single vulnerability and its impact) to greater schemes (dependencies between protocols or even dependencies in critical infrastructure). Graphingwiki works well in this context, because there are essentially no restrictions on the type of data that can be represented.

It should be noted that none of the other methods mentioned here have been used in the context of AV software. To our knowledge, there are no other studies on the dependencies in AV software.

3 Results

In this section, we present our review of the historic vulnerability data regarding AV software, and the results of related vulnerability co-ordination work.

3.1 Analysis of AV Vulnerability Data

This section contains the data in numbers and shares and presents the picture gathered from the media during our research. We use the SCAP [42] set of standards (including CVE, CPE, and CVSS) to measure gathered vulnerability data. The gathered data provided insight into the problematic areas of AV software, and guided the development of a test suite to exercise their robustness. The methods used in the coordinating the fixing process are described, as well as the results of the coordination.

In our previous analysis [2], AV vulnerability data was gathered manually from the U.S. NVD database [31]. In this paper, we parsed the data from NVD in XML format and uploaded all entries containing the words 'virus' or 'malware' to Graphingwiki with the help of automated scripts, which were used to ease laborious data gathering process and minimise errors and loss of data in data collection process.

As explained later in greater detail, we combined the NVD data with data from the SecurityFocus database [43]. In this case, the main function of the scripts was to convert the freeform vulnerability descriptions to structural data. The data from different sources can be seen to represent different expert opinions on the vulnerability. Our approach was not to combine these opinions in any way, though different opinions can be formed into a single dependency by considering the combination of various edges between two nodes as a dependency. Currently, the views to the data are generated automatically, but it is the analyst's task to decide on the most appropriate data points for his purposes. Algorithmic or other formal methods to form dependency views could be implemented as custom plugins. Our initial experiences indicate that gathering and comparing data from different vulnerability databases in this manner is a promising, yet largely neglected research area.

As the data gathered for this paper is more systematic, uniform and wider in scope, direct comparison to our previous analysis is not meaningful. The gathered data is represented in the following formats: CVE [10] enumerates unique vulnerabilities, CVSS [11] measures vulnerability severity, CPE [9] enumerates products affected, and finally, CWE Common Weakness Enumeration [12] provides a listing of weakness types.

The total number of vulnerabilities was 346, and included vulnerabilities in the products of practically all known antivirus vendors. The data spanned from the year 1998, although the bulk of the vulnerabilities were from the years 2004-2008, with a noteworthy peak in the year 2005. As can be seen in Figure 3), the number of AV vulnerabilities has expanded rapidly through these years. As measured by their CVSS scores, the average severity of all the gathered AV vulnerabilities is 6.56. This means that antivirus vulnerabilities are generally quite severe, as the NVD database considers vulnerabilities with a CVSS score equal or greater than 7.0 to have a high severity. Further, the severities of vulnerabilities have been in a slight rise during these years, as can be seen in Figure 4. By combining these two statistics, it is clear that after the year 2004 there has been a significant increase of vulnerabilities with high or medium severities.

Most of the vulnerabilities (276 out of 346) were exploitable remotely according to NVD. This shows that, as we speculated, the AV software has difficulties in robust handling of the data it inspects. With antivirus software, the type remotely exploitable means that data can be sent to the system e.g. via email, after which the AV system must inspect it. NVD classified most of the vulnerabilities as having a low access complexity, which means that ex-

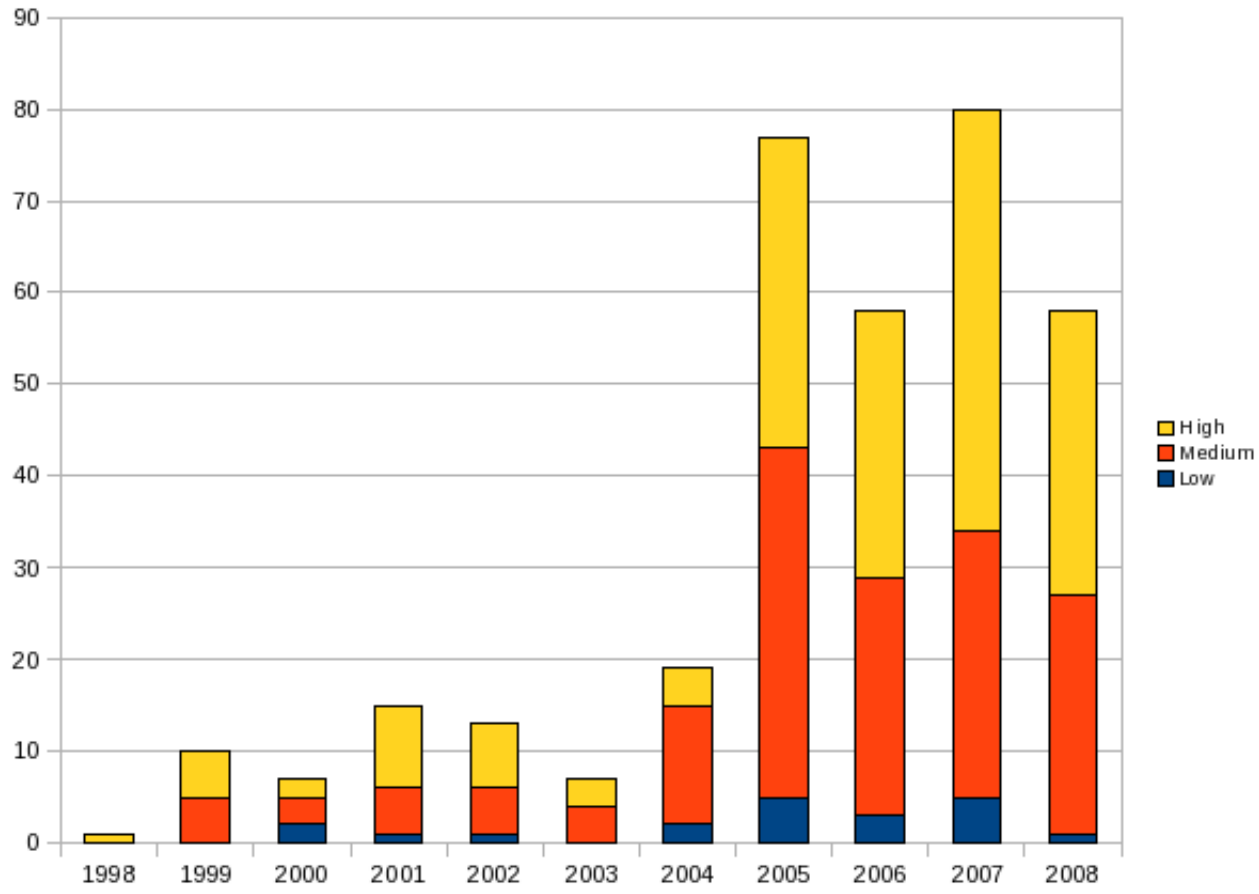

Figure 3. AV vulnerabilities of different severity per year

ploiting the vulnerabilities is not considered to be a difficult task, which further emphasises their severity.

From our data, we noted that archive formats are associated with a large portion of the vulnerabilities (see Figure 5). The most frequent archive formats were RAR, CAB and ZIP. Vulnerabilities in parsing file formats are often trivial to exploit, as well as relatively easy to discover by the means of black-box testing, i.e. fuzzing. This hypothesis can be ascertained by examining the type information of the vulnerabilities.

The NVD vulnerability database presents vulnerability types with CWE identifiers. Only about a fifth of the vulnerabilities we gathered had error type information. Therefore, we used vulnerability type information from the Securityfocus vulnerability database, which has type information about 262, or 76% of the vulnerabilities. The Securityfocus databases use an undocumented vulnerability taxonomy, which according to observations closely resembles the widely used Aslam taxonomy [3] [28].

Our previous analysis showed that the most common error type in AV software is design error. An analysis with more data indicates that boundary condition errors (60 vulnerabilities) and failure to handle exceptional conditions (46 vulnerabilities) are as prevalent as design error (59 vulnerabilities). The yearly observation depicted in Figure 6 indicates that the observed peak in the year 2005 correlates with a similar peak with the vulnerability type failure to handle exceptional conditions. Many of the vulnerabilities of this type were due to problems in parsing.

The observations prompted research in PROTOS GENOME -project, where malformed archive files were generated to test the robustness of AV software. The results of this research are reported in [34]. Our analysis suggests that the biggest factors for the peak in AV vulnerabilities in the year 2005 were related to different archive file formats, mainly RAR and ZIP. The results of PROTOS GENOME archive test set affected in turn the number of vulnerabilities in the year 2008.

The media follow-up was performed in 2006 and gained results of the analysis were also presented in a previous paper [2]. The media analysis resource consisted of 92 news items. The results can be seen in Table 7.

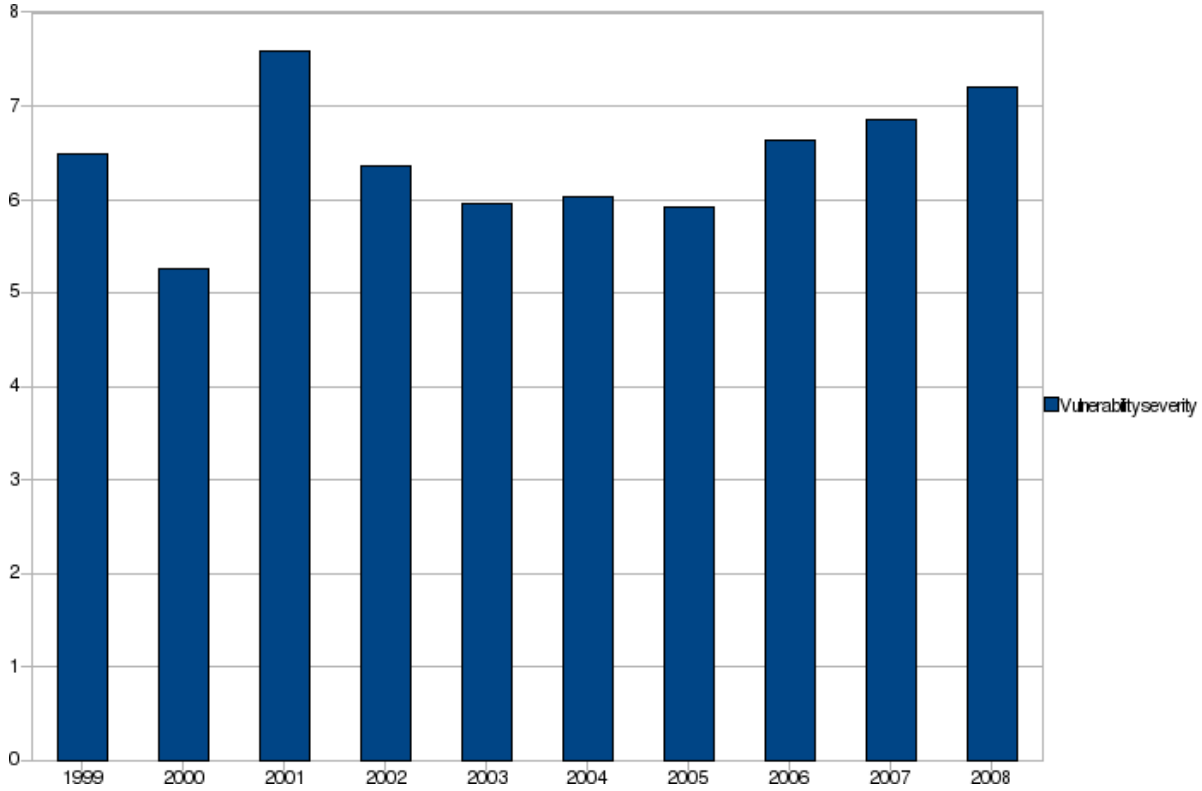

Figure 4. Average yearly severity of AV vulnerabilities

In general, AV software is presented in the news in a very positive light as continuously developing industry, which provides better solutions and increased security. The discussion of more negative issues is neglected. As our analysis of vulnerability data indicated, AV software have remarkable amount of vulnerabilities that can have wide-ranging effects. However, from the results of the media analysis, it can be noted that only 11.9% of all the collected AV-related newspaper articles dealt with AV software vulnerabilities and malfunction. We think that awareness of AV software vulnerabilities and their impact is not at the appropriate level and the usage of antivirus software should be considered more carefully in critical systems.

3.2 Using the PROTOS MATINE Model in AV Vulnerability Disclosure

Effective responsible vulnerability disclosure requires that data about vulnerabilities is presented to all affected vendors to enable them to repair the found vulnerabilities in their software. In the case of the archive format tests, the scope of potentially affected software is colossal. We used the PROTOS MATINE model to form a technical view on the usage of archive formats. The best sources of information for constructing the view in a rapid fashion were expert interviews and sources of formalised data on software. Archive formats have an extensive history, both in specification and implementation, which makes them a tedious subject of study from the literature standpoint.

The first data source we employed was the APT package management system [1], which we used to identify software that used popular archive handling libraries. We visualised this data in a technical view (Figure 8), which was augmented with the help of expert interviews. The view shed light on the scope of the potential problems - as we quickly saw, the usage of archive formats ranged from basic operating system and network functionality to applications.

The technical view was further enhanced using the NVD vulnerability database as a data source. We searched the CVE entries with the help of automated scripts for mentions of the archive formats comprising the archive test set. We filtered the gathered CVE entries by hand to remove the vulnerabilities which were not actually related to archive formats. We divided the vulnerable products gathered in this manner in groups based on their type. We gave the resulting list of products and categories to experts, who supplemented it with products of similar function.

The resulting view presented a clear direction to the vulnerability coordination process, which was performed in two phases. In the first phase, we contacted a small num-

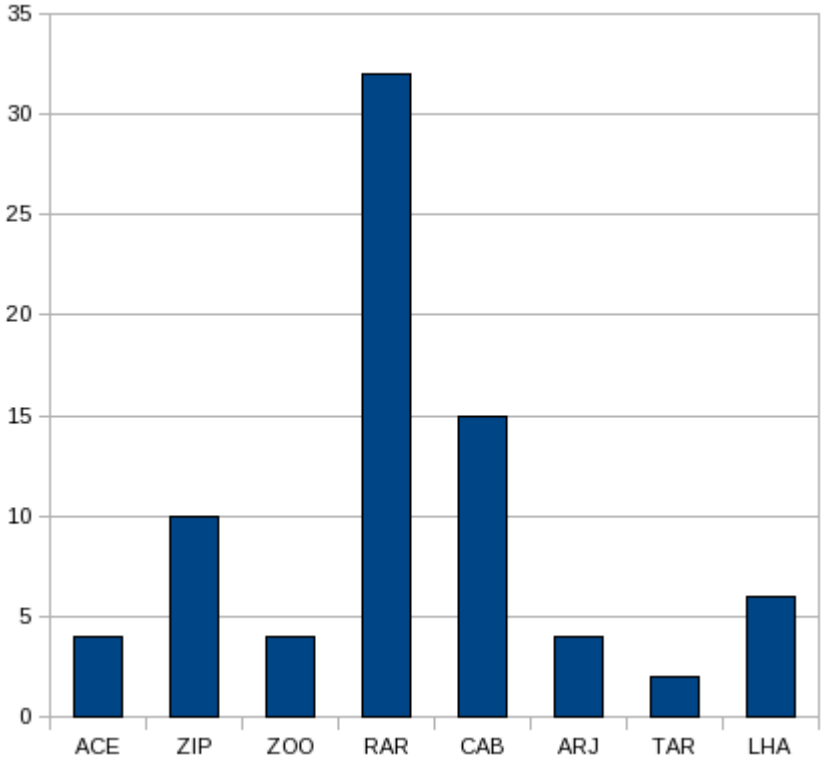

Figure 5. Archive file formats associated with AV vulnerabilities

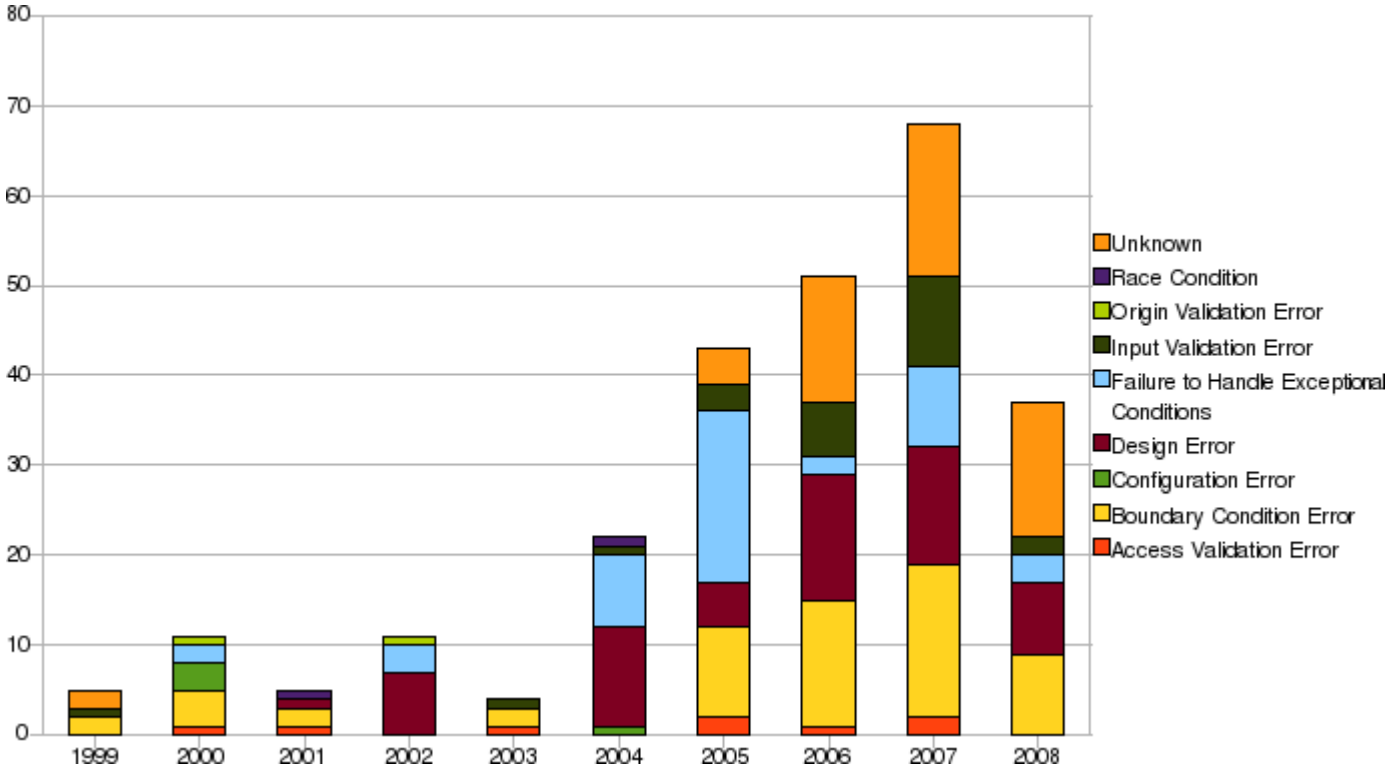

Figure 6. SecurityFocus vulnerability classification of AV vulnerabilities

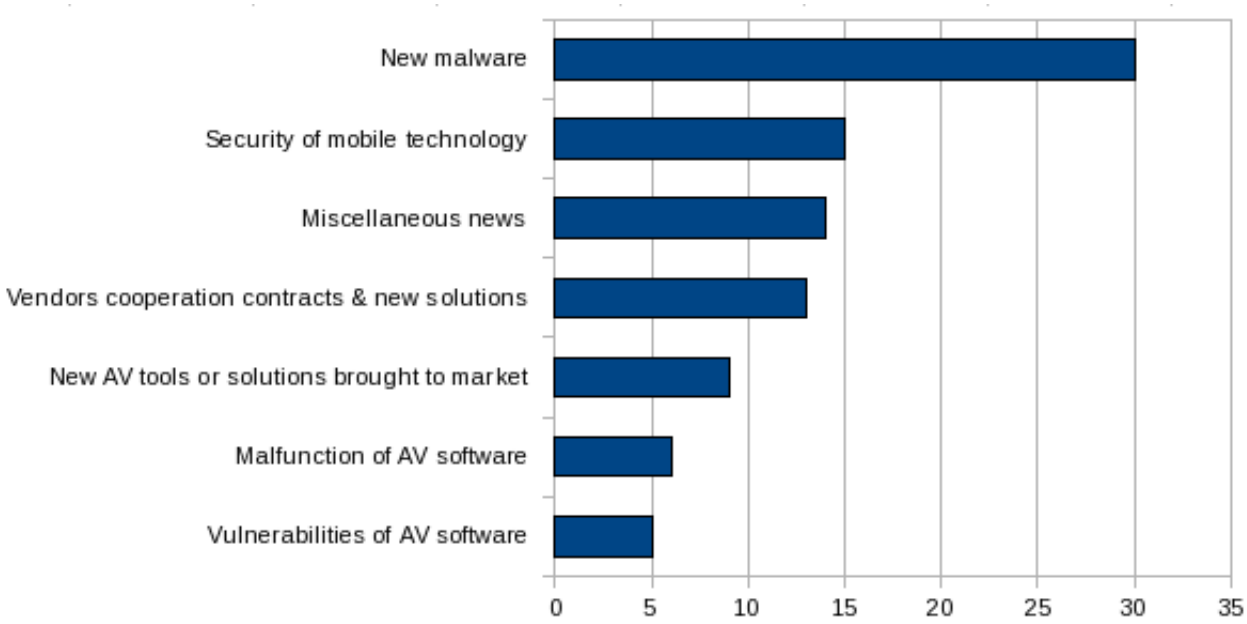

Figure 7. News topics concerning AV

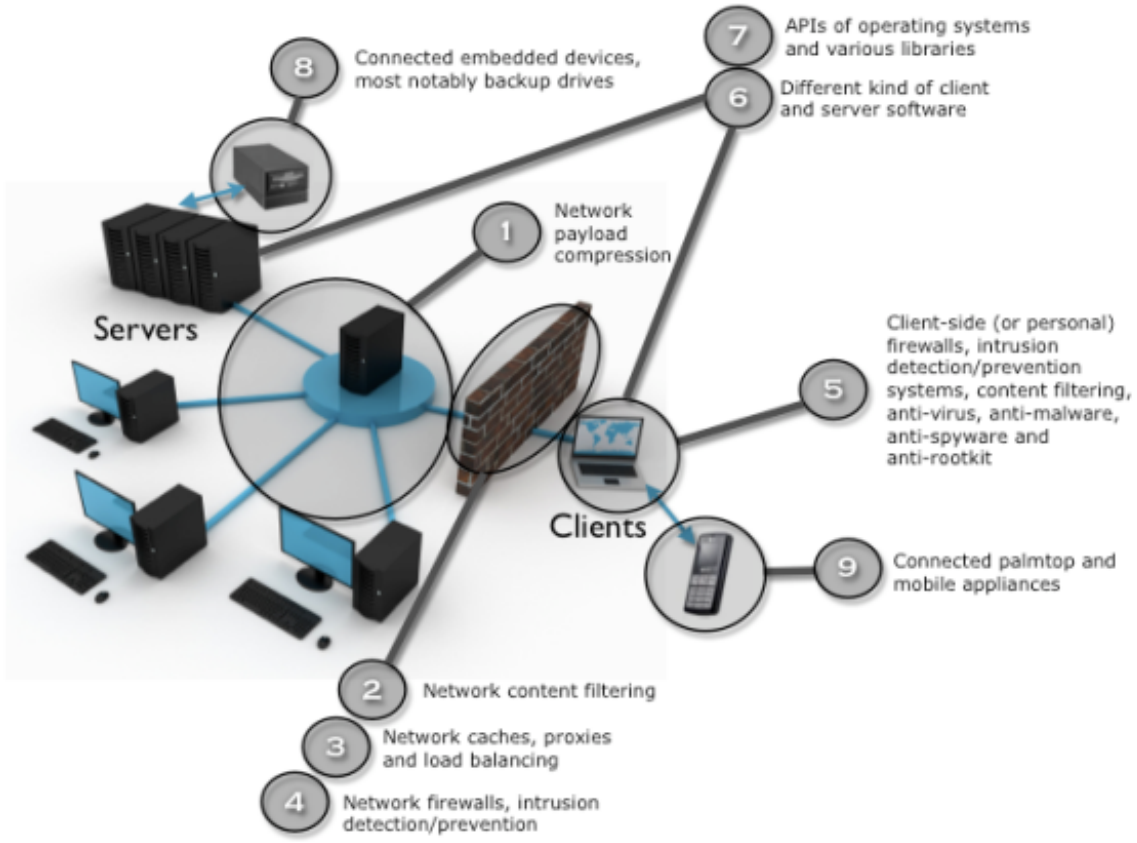

Figure 8. A simple technical view depicting the use of archive formats

ber of key vendors. The rationale for this decision was that some of these vendors have quite extensive product lines and they would require more time for testing. While the first phase was in progress, we continued to enumerate vendors and their contact addresses for the second phase. The coordination process was performed in co-operation with CERT-FI and CPNI (The UK Centre for the protection of National Infrastructure) followed the constructive disclosure process as outlined in [29]. Altogether, over a hundred vendors were contacted by CERT-FI about the test suite. Approximately 25% of the contacted vendors wanted to test their products with the test suite. The advisory stated 12 vendors vulnerable, 8 vendors not vulnerable, and the status of 30 vendors as unknown.

Published vulnerabilities represent only a small subset of the actual vulnerability of software. For various reasons, measuring the numbers of vulnerabilities is not simple [33]. Often, a bug is reported as a vulnerability only if a security conscious developer has a look at it. Bugs and vulnerabilities found internally or as a result of an audit often do not get reported. Many software vendors do not feel comfortable about sharing details about vulnerabilities in their software. It is fairly common to omit mentioning fixed vulnerabilities in change logs, or to refer to them as "reliability fixes". These factors make it difficult to measure the improvement bestowed by the archive test suite. In a more general sense, they contribute to the difficulty of making informed decisions about vulnerabilities.

Fixes were made by various vendors after the publication of the test set. This was in part due to fixes in open source products that were incorporated into commercial products and open source distributions. There is some anecdotal evidence on the fact that some vendors did not perform any testing, and that some vendors had a silent disclosure of patches to the archive set test cases.

In the test set documentation [34], we tested a sample of five antivirus software for vulnerabilities and four of them were vulnerable. We re-tested five of them for the purposes of this paper, and the same four of them were still vulnerable against the same test material.

F-Secure was the only antivirus vendor to publish updates and a security bulletin based on the archive test set at the time of publication [22], though ClamAV did publish a bulletin and an update at a later date. When the grace period before any public announcement of the danger spans months or even years, there will be vendors who issue silent fixes and move on without joining the public advisory.

4 Discussion

We set out to understand the dependency of critical infrastructure on the AV software, nature of AV software with respect to information security, security of the AV software itself both historically and presently and perception of the media and thus general public on the role of the AV software. In short, we aimed to disclose and understand any risks that such security software may pose on the critical infrastructure.

We experienced some difficulties in our data gathering efforts regarding antivirus software. Highly competitive fields do not encourage research, open standards, and open publication of data in general, and the antivirus industry is no exception. Information on the common scanning engines and possible undocumented standards used by the AV industry would provide a significant advantage to deciphering their dependencies in terms of vulnerability. As AV vendors are naturally reluctant to reveal their trade secrets, we are missing the data that would in many other cases be available via public metrics and expert interviews from the developers of products and standards. However, interviews of the actors of the critical infrastructures could provide a further insight into the criticality of applications, and hence the effects of vulnerabilities.

As the amount of public information on AV vulnerabilities leaves much to be desired, the significance of media and other public sources is emphasised. Our observation on the labour-intensiveness of the media follow-up as a data gathering method prompts attention to it as a further field of study. Still, additional sources such as social media could significantly augment the scope of public information. In this paper, we successfully used automatic data gathering methods for vulnerability databases, but employing similar methods for free-form news articles presents further challenges. The field of natural language processing has shown some promising results, methods such as support vector machines have proven useful for many tasks. It has been suggested that some dependencies could be discerned from textual structures alone. All in all, automated methods for gathering public information and refining it to more useful forms could require extensive research.

Selective aggregation of different data feeds also constitutes a promising information gathering method. As an example, package management software and the SCAP project use different nomenclature for software packages, which makes it harder to track vulnerabilities regarding Linux software packages. However, most Linux distributions provide security advisories that include SCAP compliant CVE vulnerability identifiers, which can be used to find SCAP compliant CPE software identifiers. Combining these facts from, e.g. APT popularity contest, project that attempts to map the relative popularity of Debian Linux software packages could provide insight into the vulnerability of some of the most popular Linux software. This is how using the three sources in conjunction could provide otherwise unattainable information.

The large amount of archive file format related bugs in

AV software suggests that software components used in critical infrastructure should be exposed to thorough testing – for it seems that vendors' quality assurance processes do not guarantee a sufficient level of robustness. Finding bugs in any software is not enormously difficult [47], however. As the data gathered in AV case illustrates, most of the bugs are of the same type. Whenever a bug is found, the focus is on fixing the bug, not finding its causes. There is a need for methods for understanding the bugs so that we could write programs with fewer bugs. Also, the prevalence of common vulnerability types, such as boundary condition error, in antivirus software indicates that they are largely created with programming languages that are particularly susceptible to those types of error, such as C and C++. The usage of programming languages that are inherently more secure should be examined in security-critical portions of the code, notwithstanding the possible performance penalties.

Even though the proportion of AV vulnerabilities to all vulnerabilities is diminutive, we had great difficulties in digesting the data. Although we used only one source of information on the vulnerabilities, manually trawling through the relevant vulnerabilities and analysing them was challenging. Methods for analysing the gathered information were sorely needed. Using graphs to visualise data helped in understanding the big picture, but still left plenty of room for development. For some example graphs, see Figures 9 and 10 Currently, we only analyse the graphs by visual observation. Graph theoretical and social networking analysis methods for analysing different properties of the graphs remains a future field of study.

As the summaries in the different fields of dependency analysis showed, there are still a number of dependency types we could study. Dependency through proximity and other context-related dependencies could provide benefits for analysis. The temporal dimension of dependencies is another field of further study. In the current implementation, all historical data are stored, but have not been used in analysis. Dynamic simulation of events and changes in the dependency graph would constitute an interesting line of future research.

Future research also includes related studies in other types of security software. Dependency studies could direct robustness testing efforts on the modules deemed to have the most impact on critical systems. Further research into the generation of test sets could improve their effectiveness in finding vulnerabilities. The testing efforts could serve to raise the bar for the security of critical systems.

5 Conclusion

The main goal for this paper was to examine AV software vulnerabilities and the risks they bring to critical information infrastructure systems. The PROTOS MATINE model was used as a method for disentangling the untrodden field of AV vulnerabilities in a rapid, iteratively expanding fashion. This paper presents the results of our research, which focused on AV software vulnerabilities and the picture depicted by the dependencies between these vulnerabilities.

By applying the PROTOS MATINE model through the study of media follow-ups, expert interviews, specifications, market situation, historical data, public vulnerability data and usage scenarios we found out that there are implementation level security issues in AV software that not only make it ineffective against malware, but also actually open new ways to attack the system. There is a substantial amount of information about such vulnerabilities in the past. AV products are an attractive target due to mono culture and high access privileges involved.

AV software itself was discovered to share a meta level one type of dependency risk exposure through the same archive formats being implemented in all AV products. This dependency risk spans beyond AV products, and when vulnerabilities related to archive forms are disclosed, then a very large vendor base of more generic products varying from consumer devices to network infrastructure have to be considered.

We found that issues with handling archive files have been the main reason for the fast rise of AV vulnerabilities in recent years. Our observations prompted robustness testing research of archive file formats in the PROTOS GENOME project. The results and the followup results presented here demonstrate that archive file formats are still a big issue in AV software. Ten months after public availability, preceded by a year-long period of limited distributions to vendors only, 4 out of 5 tested products were still vulnerable.

AV vendors do not necessarily fix vulnerabilities uncovered by published test sets. At least some AV vendors react to disclosed security issues and improve their products, but overall there is no significant trend for better or worse. More vulnerabilities akin to the ones we observed can be found by testing in a relatively straightforward fashion.

The results from media follow-up draw a quite desolate picture from the viewpoint of equal communication. Media (and the public) mostly do not recognise or discuss the risks related to dependency on the AV products. The AV vulnerabilities are seldom reported, and there truly is a lack of open discussion and controversial opinions, e.g. on the reliability of different AV software. Media concentrates on reporting new malware and fusions of AV enterprises.

Traditionally, AV software security has been measured by its ability to detect malware. However, some recent studies, even by the AV industry itself, have shown that the efficiency of malware may be of suspect. Nevertheless, AV is widely used despite this criticism on the effectiveness of the very approach. The use of AV software in critical in-

Figure 9. Comparison of CWE and Securityfocus vulnerability types in some serious vulnerabilitie

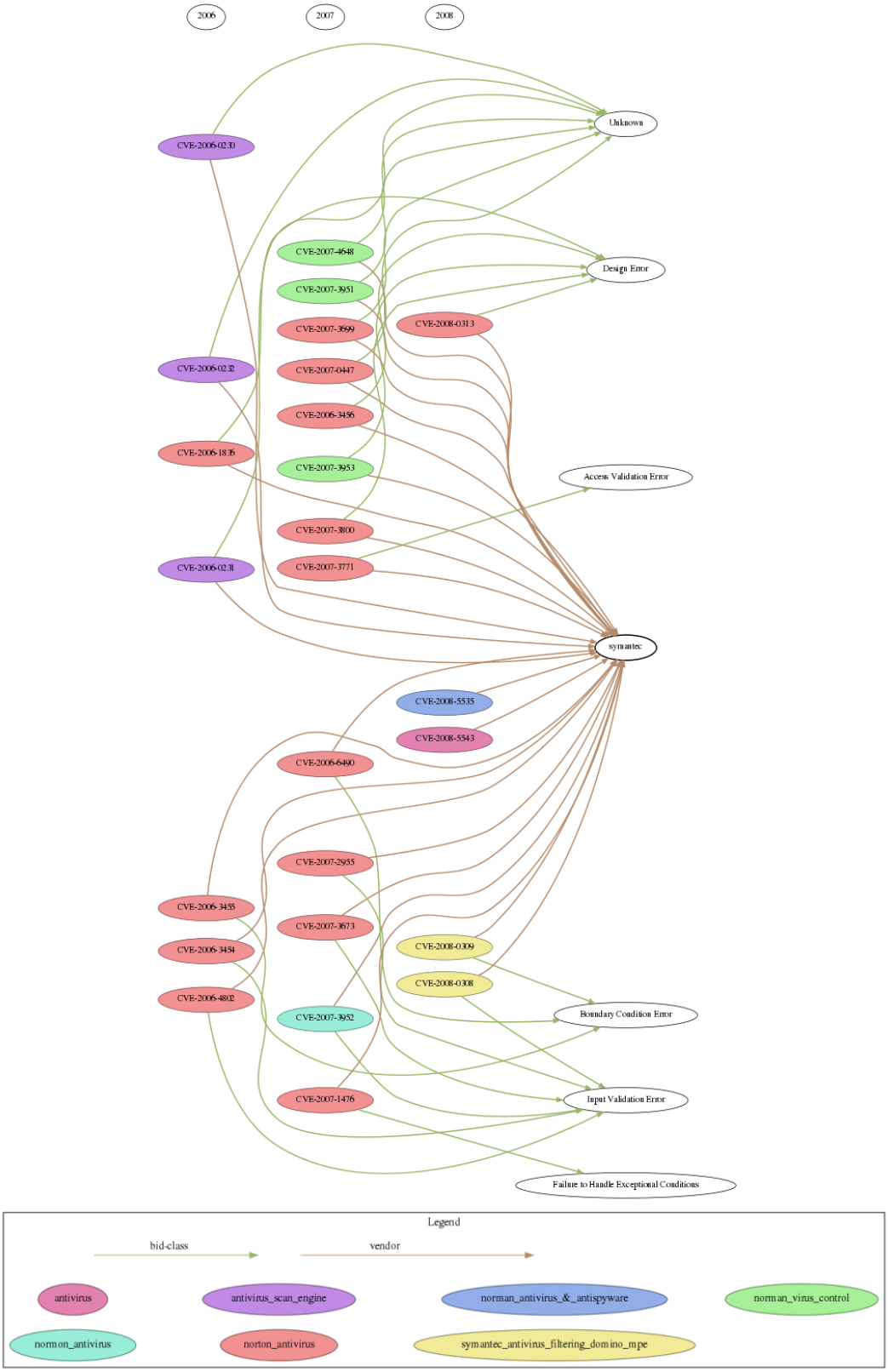

Figure 10. Some recent serious AV vulnerabilities in Symantec products

frastructure is wide spread and sometimes even mandated by laws and regulations

Our research indicates that AV software and AV vulnerabilities should be considered in the context of critical infrastructures. Firstly, awareness on the drawbacks of AV software should be spread among the planners of the critical infrastructures. Following this, the suitability of the software should be reconsidered on a system-by-system basis, along with the planning of divergent defences and defence strategies. Information on the interrelationships of different formats and products, and the vulnerability histories (track records) of the products can prove to be valuable decision-making tools in this process.

The context of AV software vulnerabilities and critical information infrastructure is still to be conceptualised by grounded theory. The graphs created in Graphingwiki could be analysed by means of graph theory to gain more insight into the dependencies. In order to do this, the semantic analysis should be further investigated. This would then provide the meaning for the mathematical results. Consistent and planned media follow-up as well as expert interviews would provide enough material for qualitative analysis. At the same time, a more thorough statistical analysis and graph theoretical approach could add more quantitative information. This future work should provide more in depth understanding of dependencies in AV software. More research should also be done in the field of automated data gathering methods since the media follow-up is laborious to perform manually. Automated data gathering would generate fewer errors, and on a large scale, it would give more reliable results.

This study on the vulnerability dependencies in AV software showed us that the PROTOS MATINE model is well suited for gathering information on a previously unknown subject, organising and analysing that information, finding points of interest for further, more focused research, and finally, extrapolating on the impact of the discovered vulnerabilities. The data gathering part of the method benefits from the use of multiple sources of information. The organising and analysing did benefit from the Graphingwiki visualisations, which pointed to the direction of archive file formats as the source of many vulnerabilities. Finally, expert interviews gave a wider perspective to the impact of these vulnerabilities and enabled the responsible vulnerability disclosure coordination effort, as we realised that these vulnerabilities could be present in various software beyond our initial target.

By applying the model, we collected antivirus prevalence, mandate and vulnerability track record data, we identified antivirus related risk factors and disclosed new information about media perception of antivirus, new vulnerabilities and handling of these vulnerabilities. In short, we disclosed and now understand better the risks that the antivirus software may pose on the critical infrastructure.

6 Acknowledgements

The authors would like to thank MATINE (Scientific Advisory Board for Defencce in Finland) and infotec Oulu for Financial support of the research. The authors express their gratitude also to Jani Kenttälä of Clarified Networks for valuable help on creating some of the pictures in this paper.

References

[1] *Advanced Packaging Tool* `http://en.wikipedia.org/wiki/Advanced\_Packaging\_Tool`, May 8, 2009

[2] Askola, K., Puupera, R., Pietikainen, P., Eronen, J., Laakso, M., Halunen, K., and Röning, J., *Vulnerability Dependencies in Antivirus Software*, SECURWARE 2008, The Second International Conference on Emerging Security Information, Systems and Technologies, pages 273-278, 2008

[3] Aslam T., Krsul I., and Spafford E. H., *Use of a taxonomy of security faults*, 19th NIST-NCSC National Information Systems Security Conference, pages 551-560, 1996.

[4] Bagheri, E. and Ghorbani, A. A., *The State of the Art in Critical Infrastructure Protection: A Framework for Convergence*, International Journal of Critical Infrastructures, Vol.4, no. 3, pages 215-244, 2008.

[5] Bassham, L. E. and Polk. W. T., *Threat Assessment of Malicious Code and Human Threats* (NISTIR 4939) `http://csrc.nist.gov/publications/nistir/ir4939.txt`, May 8, 2009

[6] Beizer, B. *Software Testing Techniques*, Second edition. (1990). International Thomson Computer Press. ISBN: 1-850-32880-3

[7] Chai, C-l., Liu, X., Zhang, W. J., Deters, R., Liu, D., Dyachuk, D., Tu, Y. L., and Baber, Z., *Social Network Analysis of the Vulnerabilities of Interdependent Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 3, pages 256-273, 2008.

[8] Christoderescu, M., Jha, S., Seshia, S. A., Song, D., and Bryant, R., *Semantics-Aware Malware Detection*, IEEE Symposium on Security and Privacy (S&P'05), pages 32-46.

[9] *Common Platform Enumeration* `http://cpe.mitre.org/`, May 8, 2009

[10] *Common Vulnerabilities and Exposures* http://cve.mitre.org/about/index.html, May 8, 2009

[11] *Common Vulnerability Scoring System* http://nvd.nist.gov/cvss.cfm?version=2, May 8, 2009

[12] *Common Weakness Enumeration* http://cwe.mitre.org/, May 8, 2009

[13] Cox L. and Delugah H.S., *Dependency Analysis Using Conceptual Graphs*, Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001.

[14] De Porcellinis, S., Setola, R., Panzieri, S., and Ulivi, G., *Simulation of Heterogeneous and Interdependent Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 1/2, pages 110-128, 2008.

[15] Diestel, R., *Graph Theory*, 3rd edition, Graduate Texts in Mathematics, Vol. 173, Springer-Verlag, Heidelberg, 2005.

[16] *Digitoday Finland*, http://www.digitoday.fi/, May 8, 2009

[17] Dixon, J., *How Good Is Your Network Neighborhood Watch*, http://media.techtarget.com/searchFinancialSecurity/downloads/How_Good_Is_Your_Network_Neighborhood_Watch.pdf, May 8, 2009

[18] Dudenhoeffer, D. D., Permann, M. R., and Manic, M., *CIMS: A Framework for Infrastructure Interdependency Modeling and Analysis*, Proceedings of the 2006 Winter Simulation Conference, pages 478-485.

[19] Eronen J. and Laakso M., *A Case for Protocol Dependency*, In proceedings of the First IEEE International Workshop on Critical Infrastructure Protection. Darmstadt, Germany. November 3-4, 2005.

[20] Eronen J. and Röning J., *Graphingwiki - a Semantic Wiki extension for visualising and inferring protocol dependency*, Proceedings of the First Workshop on Semantic Wikis (SemWiki2006 - From Wiki to Semantics), co-located with the 3rd Annual European Semantic Web Conference (ESWC). Budva, Montenegro, 11th - 14th June, 2006.

[21] Eronen, J., *A collaborative method for assessing the dependencies of critical information infrastructures* M.Sc. (Tech) Thesis for the Department of Electrical and Information Engineering at University of Oulu. URL: http://www.ee.oulu.fi/research/ouspg/protos/sota/matine/method-thesis/di.pdf, May 8, 2009

[22] *F-Secure Security Advisory FSC-2008-2*, http://www.f-secure.com/en_EMEA/support/security-advisory/fsc-2008-2.html, May 15, 2009.

[23] Hicks, B., *Network Anti-Virus Market Trends*, Faulkner Information Services, 2005.

[24] Hills, A. *Insidious Environments: Creeping Dependencies and Urban Vulnerabilities*, Journal of Contingencies and Crisis Management, Vol. 13, No. 1, pages 12-20, 2005.

[25] *HIPAA* http://www.cms.hhs.gov/HIPAAGenInfo/Downloads/HIPAALaw.pdf, May 8, 2009

[26] *International CIIP Handbook 2006 (Vol. II)*, eds. Myriam Dunn, Victor Mauer; Center for Security Studies, ETH Zurich.

[27] Itzwerth R. L., MacIntyre C. R., Shah S., and Plant A. J.,*Pandemic influenza and critical infrastructure dependencies: possible impact on hospitals*, The Medical Journal of Australia, 185, pages S70-S72, 2006.

[28] Ko K., Jang I., Kang Y., Lee J., and Eom Y., *Characteristic Classification and Correlation Analysis of Source-Level Vulnerabilities in the Linux Kernel*, Lecture Notes in Computer Science, Volume 3802, pages 1149-1156, 2005.

[29] Laakso, M., Takanen, A., and Röning, J., *Introducing Constructive Vulnerability Disclosures*, The 13th FIRST Conference on Computer Security Incident Handling, 2001.

[30] *Malware infections in protected systems*, http://www.pandasoftware.jp/scan/pdf/panda_lab_research_paper.pdf, May 8, 2009

[31] *National Vulnerabilty Database*, http://nvd.nist.gov/, May 8, 2009

[32] NATO, *The Protection of Critical Infrastructure*, Committee Report, 162 CDS 07 E rev 1, 2007, http://www.nato-pa.int/Default.asp?SHORTCUT=1165, May 15, 2009

[33] Ollman G., *Counting Vulnerabilities*, http://blogs.iss.net/archive/CountingVulns.html, May 8, 2009

[34] *OUSPG PROTOS-Genome Test Suite c10-archive*, http://www.ee.oulu.fi/research/ouspg/protos/testing/c10/archive/index.html, May 8, 2009

[35] Rinaldi, S. M., Peerenboom, J. P., and Kelly, T. K., *Identifying, understanding, and analyzing critical infrastructure interdependencies*, IEEE Control Systems Magazine, Vol. 21, no. 6, pages 11-25, 2001.

[36] Robert, B., de Calan, R., and Morabito, L., *Modeling Interdependencies Among Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 4, pages 392-408, 2008.

[37] Roivainen, H-L., *Discovery of hidden social networks in software companies*, M.Sc. (Tech) Thesis for the Department of Electrical and Information Engineering at University of Oulu, 2008.

[38] Rosato, V., Issacharoff, L., Tiriticco, F., Meloni, S., De Porcellinis, S., and Setola, R. *Modelling Interdependet Infrastructures using Interactins Dynamical Models*, International Journal of Critical Infrastructures, Vol.4, no. 1/2, pages 63-79, 2008.

[39] *Sarbanes-Oxley Act* http://www.sarbanes-oxley.com/section.php?level=1\&pub_id=Sarbanes-Oxley, May 8, 2009

[40] *Secunia Vulnerability Statistics* http://www.secunia.com, May 8, 2009

[41] *Cyber Assessment Methods for SCADA Security*, http://www.oe.energy.gov/DocumentsandMedia/Cyber_Assessment_Methods_for_SCADA_Security_Mays_ISA_Paper.pdf, May 15, 2009

[42] *Security Content Automation Protocol* http://nvd.nist.gov/scap.cfm, May 8, 2009

[43] *Securityfocus vulnerability database* http://www.securityfocus.com/vulnerabilities, May 8, 2009

[44] St. Neitzel, M., *Welcome to 2007: the year of professional organized malware ...* (HISPASEC) http://blog.hispasec.com/virustotal/recursos/welcome_2007.pdf, May 8, 2009

[45] Tolone, W. J., Wilson, D., Raja A., Xiang W., Hao H., Phelps S., and Johnson E. W.,*Critical Infrastructure Integration Modeling and Simulation*, Lecture Notes in Computer Science, vol. 3073, pages 214-225, Springer Berlin Heidelberg, 2004.

[46] Turner, D., Entwisle S., Fossi M., Blackbird J., McKinney D., Conneff T., Whitehouse O., *Symantec Internet Security Threat Report vol. X*, http://www.symantec.com/business/theme.jsp?themeid=threatreport, September 2006, May 8, 2009

[47] Viide, J., Helin, A., Laakso, M., Pietikäinen, P., Seppänen, M., Halunen, K., Puuperä, R., and Röning, J., *Experiences with Model Inference Assisted Fuzzing*, Second USENIX Workshop on Offensive Technologies (WOOT'08'), 2008.

[48] World Wide Web Consortium, *Resurce Descrption Framework* http://www.w3.org/RDF/, May 8, 2009

[49] Zio, E., *From Complexity Science to Reliability Efficiency: A New Way of Looking at Complex Network Systems and Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.3, no. 3/4, pages 488-508, 2007.