A Case for Protocol Dependency

Juhani Eronen, Marko Laakso
University of Oulu, Computer Engineering Laboratory
Linnanmaa BOX 4500, FIN-90014 University of Oulu, Finland
ouspg@ee.oulu.fi

Abstract

Vulnerabilities infest information technology. There is a lack of tools in risk assessment for understanding the impact that the disclosed vulnerabilities have on the critical information infrastructures. To address this need, this work derives a new dimension of dependency from practical vulnerability work, namely that of protocol dependency. Classic technology dependency views were reviewed, a chain of systematic vulnerability disclosures was followed as a case study and analysis revealed evidence of protocol dependency. Extrapolating from the experiences of a complex case, this new dependency dimension can be modelled. The model will benefit from going beyond a narrow technical view.

Keywords: critical infrastructure protection, information systems, risk assessment, security assessment, protocol dependency, interdependency, vulnerability, robustness

1. Introduction

The critical infrastructures have been penetrated by information systems. The very basis that we depend on has become technologically entangled.

> "The protection of critical infrastructures such as telecommunications, energy, financial services, health care, public services, and transportation ... not only exhibit strong interdependence but are also increasingly relying on information systems for their operation." [12]

Vulnerabilities infest information technology. The number of information technology vulnerabilities tracked by the information security watch-dog CERT/CC has increased from a hefty 1090 occurrences in year 2000 to over 1200 in the first quarter of 2005 alone. Incidents where these vulnerabilities have been abused have become so frequent that this watchdog has lost track of them [5].

The rise of vulnerabilities is new to traditional industries that have only quite recently become dependent on the information infrastructure or information technology in general [18]. Vulnerabilities manifest in new threats that generate risks for industries which protect their assets and operations with the help of risk management. However, vulnerabilities cannot be mitigated efficiently without first understanding the dependencies involved [15].

Technological dependency has been investigated before in various studies and programs (see for example [8], [25], [2], [15], [19] and [18].) The effects of dependency cannot be negated with the help of mere technological solutions which would only increase the complexity of the system, and would therefore further add to the effects. Instead, efficient risk management of dependent technologies would require multifaceted analysis methods for different aspects of technological dependency. [9] [15] The science of dependencies is relatively immature and many dependencies are not yet understood or even uncovered [25].

There is a lack of tools in risk assessment for understanding the impact that the disclosed vulnerabilities have on the critical information infrastructures. How to determine the impact of a disclosure of a vulnerability in a product, in certain types of products or in a more abstract concept such as a protocol or implementations of a certain protocol? There is no easy way to gather answers to the following questions:

1. If a product is affected, are similar products affected?
2. If a product is affected, are seemingly unrelated or different products affected?
3. If a vulnerability is found in one networking context, e.g. the Internet, does it affect a different context, such as the telephone networks?
4. If a vulnerability affects desktop computing, are appliances with embedded software in danger?

One approach for finding the answers is to explore protocols; languages shared by the information systems for communication. Purpose of this work is to derive a new dimension of dependency from practical vulnerability work,

namely that of protocol dependency. It is realised when protocols within a single protocol family or even between protocol families have a connection. The impact area of vulnerabilities in a shared component is greatly expanded due to protocol dependency. This may lead to faults that can have a significant effect on an infrastructure.

This paper first introduces classic views on the dependencies of critical infrastructures, and the methods used to manage related risks. The gravity of vulnerabilities for critical infrastructures is demonstrated. A method for systematic vulnerability assessment is discussed, and different levels of vulnerability are analysed and classified. Resulting cases on protocol dependencies and protocol views are presented along with analysis on causes and effects of the dependencies. Finally, the discussion expands to a modelling method that goes beyond narrow technical views.

1.1. Risk management of dependencies

> "The essence of risk management lies in maximising the areas where we have some control over the outcome while minimising the areas where we have absolutely no control over the outcome and the linkage between effect and cause is hidden from us."
>
> Peter L Bernstein, 'Against the Gods, The Remarkable Story of Risk'

Risk management is used in practically all current organisations and enterprises to protect their assets and ensure their continued operation. The use of risk management as a decision-making tool is recommended throughout the organisation, from senior management to the administration of individual devices. Risk is thought of as the function of the likelihood of a threat source displaying a potential vulnerability, and the resulting impact of the adverse effect. [27] [28]

A crucial part of risk management is the risk assessment phase, in which the relevant processes and systems are identified, their threats and vulnerabilities are anatomised along with the corresponding likelihoods. From these factors risks are determined with the help of impact analysis. When the risks are known the process can continue with the development of mitigation strategies for the relevant risks. [27]

Critical information infrastructures present several challenges for efficient risk assessment. Finding relevant functions requires that infrastructures must first be dissected to a group of critical sectors. In some cases the analysis is taken further, and critical elements are identified within the sectors [8]. The sectors and elements must be evaluated in the proper context to distinguish the couplings among them, and thus among the infrastructures. These couplings are called interdependencies if the relationship between affected infrastructures is bidirectional, and dependencies if it is unidirectional [25].

The dependencies and interdependencies can be seen to embody multiple dimensions such as their environment, feedback mechanisms or failure types, and the degree and type of the relation strongly influence the operating characteristics of the affected infrastructures [28]. A vulnerability in linked infrastructures can cause failures due to a common cause, cascading failures, or even escalating failures among the affected infrastructures [25].

Current information infrastructures consist of several interconnected infrastructures that expand over countries and continents. The efficiency of communication networks has prompted their use even in the most implausible places. The composed infrastructures exhibit significant complexity and nonlinear dynamics due to the variety of interconnected elements. A single failure in one part of the infrastructure can cascade throughout the network over a varying time-span and finally cause catastrophic failures in the infrastructure. The origin of the failure might not be evident due to the complexity of the interconnections. Thus, the study of dependencies is essential for mitigating risks caused by the vulnerabilities of an infrastructure. [10]

There have been some analyses on the interdependencies of different critical infrastructure sectors and their technical and managerial layers. Some analyses have gone as far as identifying critical information technology components and their interdependencies with other components, or using historical data to model the dynamic behaviour of an interconnected system [2] [28].

However, risk analysis of a single IT component is challenging due to the very nature of information technology. IT suffers from the very same continuous evolution and modification that has made it so widely used in the first place. Changes in the environment, components, architecture, and procedures create new risks and demand continuous reassessment of the prevalent risk assessment [19] [18]. The current analytic methods for reaching a more holistic view of the risks and interdependencies of IT systems fall short of what would be required while vulnerabilities are prevalent among practically all IT systems [18]. The current trend of ubiquitous interconnectivity of IT systems has resulted in widespread vulnerability, where continuously increasing levels and varieties of attacks have emerged [24].

The heated market situation in the IT industry does not encourage efforts to reduce the vulnerabilities by the means of research and analysis [23], and technical solutions will not alone be sufficient to eliminate the risks created by the vulnerabilities due to their inherent complexity and vulnerabilities [25]. Therefore the research on vulnerabilities is a decisive topic in mitigating the risks related to IT systems and ultimately to critical infrastructures.

2. Method

This chapter presents the concept of vulnerability from the information system point of view. Furthermore, a systematic assessment method that has produced the necessary data for this study is introduced.

When requirements for a system are gathered into a system specification the focus is on the positive requirements, ie. on what the system should do. Some security and safety aspects may be incorporated in positive requirements, such as authentication and cryptography. However, specifying that system should use cryptography is more a design choice which attempts to fulfil confidentiality requirement, for example for data transportation. It would be more accurate to require that the information in transit is not disclosed to unauthorised party. Security aspects described in this fashion are negative (or inherent) requirements, classic example being "the system should not crash". A downside of the negative requirements is that they are hard if not impossible to systematically test for in the final system. The only realistic possibility is to prove that the negative requirements are not attained.

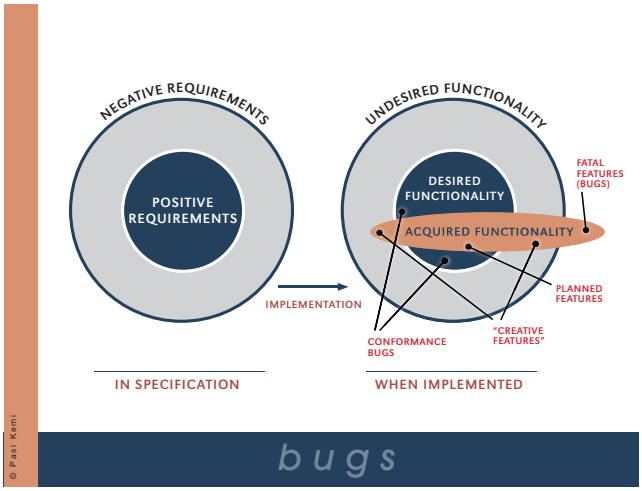

Figure 1. Plans vs. reality in implementation

Theory meets practice when the system undergoes implementation. As a result, we get more and less than we asked for. In information systems this deviation is due to a gap between typically natural language requirements and machine language of the implementation and cultural differences of people interpreting and implementing the specifications. Technical decisions, such as the choice of tools and programming language, also play a major part in the deviation. Figure 1 illustrates the complications introduced by the inherent imperfection of the implementation. At best, positive requirements result in desired features. Failure of an implementation to capture the positive requirements leads to conformance bugs, i.e. failures in conforming to the requirements. Extra functionality brought in by the implementation results in creative features, ie. features that can be used to achieve functionality that neither the requirements, designer, nor even the programmer anticipated for. As the actual functionality of the system enters the area of negative requirements, undesired features have been implemented and the security of the system has been compromised by vulnerabilities. The need to differentiate between desired and actualised functionality has been recognised in the context of critical infrastructure protection [24].

Innate vulnerabilities result when ideas are refined into concrete implementations. These vulnerabilities can be as varied as the implementations they appear in - for example. the Common Vulnerabilities and Exposures project classifies vulnerabilities in numerous continuously evolving categories [7].

Traditional vulnerability research has proceeded mostly in a very reactive fashion, addressing vulnerabilities as they are discovered in a "penetrate & patch" paradigm. An internal or external auditor finds a vulnerability in an implementation and reports it. The vendor or maintainer of the implementation can then proceed to fix it. [16] In the process, knowledge about vulnerability types has been accumulated and remedies for common vulnerabilities have become well known. Yet vendors continue to produce software that contains common vulnerabilities, which constitute a major portion of the total amount of found vulnerabilities [4].

On the stated premise, the Oulu University Secure Programming Group (OUSPG) has claimed that programming errors leading to vulnerabilities are systematic, and that many of those vulnerabilities could be eliminated by systematic testing [26]. In the PROTOS project, OUSPG set out to find several vulnerabilities from multiple implementations with systematic testing. The used approach was black-box (i.e. functional) testing of protocol implementations. [21]

Every connection of a software to its exterior takes place via an interface using a dedicated communications protocol. In effect, these protocols are used for communication between software functions, software modules, software components, software packages, or even between the software and the user. In the vulnerability testing performed during the PROTOS project, syntactical errors were inserted into protocol messages, and the messages were input to the tested implementations. The implementation was deemed to have failed the test if it exhibited vulnerable behaviour upon receiving the input. [13]

The testing was done in a systematic fashion and could be repeated and verified at any time. As many protocols are standardised and used by several implementations, the same material could be used to test a multitude of implementations using the tested protocol. Possibilities emerged

for finding a mass of vulnerabilities in several protocol implementations in a systematic fashion.

3. Results

This chapter describes three of the test materials published by the PROTOS team. These materials produced a large quantity of vulnerability data and revealed interdependencies involved in information system vulnerabilities. These materials, designed for established Internet protocols, are listed below in order of publication:

1. Lightweight Directory Access Protocol (LDAP)
   The material covers protocol version 3
2. Simple Network Management Protocol (SNMP)
   The material covers protocol version 1
3. H.225.0
   Part of the H.323 video conferencing protocol suite
   The material covers protocol version 4

The findings of the test materials confirmed the claims stated by the PROTOS project: 80% of the products tested within the project failed due to exploitable flaws [26]. The public disclosure of the different test materials were handled by Australia's National Computer Emergency Response Team (AusCERT), the US based CERT Coordination Center (CERT/CC) and the UK National Infrastructure Security Co-Ordination Centre (NISCC). The advisory for the SNMPv1 test material alone has statements from 140 vendors [3].

After the test material for LDAP it became suspect, for the PROTOS team, that there are implementation level vulnerabilities in various ASN.1 parsers, which are prevalent in protocol implementations. Initial analysis and observations supported this. Thus, the most significant impacts caused by the test materials reached far beyond the scope of the protocols involved. E.g. the LDAP and SNMP protocols both use a syntactic notation called Abstract Syntax Notation 1 (ASN.1), more specifically its Basic Encoding Rules (BER). Syntactic errors with respect to the notation were routinely used in the corresponding test materials. During testing it became apparent that the material evoked vulnerable behaviour also on unrelated implementations that used ASN.1.

The PROTOS team compiled an internal list of core protocols to select the target protocol for the next test suite. Creation of an ASN.1 Basic Encoding Rules (BER) test suite was considered, but after some consideration an SNMP test suite with extensive ASN.1 BER encoding tests was selected.

The SNMPv1 test material attracted much attention, and other actors became aware of ASN.1 vulnerabilities and began to work on the subject. NISCC compiled a list of top ten ASN.1 protocols relevant in the CNI perspective. This served as guidance for further research, and prompted the PROTOS team to generate test material for H.225.0. [20]

This discovery enhanced attention on IT-related risks. The severeness is demonstrated by the fact that the US president was briefed on the ASN.1 vulnerabilities [6]. The impact is well described by a study on Canadian critical network infrastructures.

> "Testing by Oulu University in Finland recently exposed serious vulnerabilities in the widely-used version 1 of the Simple Network Management Protocol (SNMP) and the Light Directory Access Protocol (LDAP). The formal definition language Abstract Syntax Notation 1 (ASN1) [sic] has been implicated in both of these vulnerabilities but experts have not agreed on whether the problem lies with the Basic Encoding Rules for ASN1 [sic] or the way the rules are used in implementations. Since the Basic Encoding Rules are used very widely in protocols running on the world-wide telecommunications infrastructure, the problem has serious implications regardless of the root cause." [11]

Another issue of impact beyond the obvious was uncovered during the development of the H.225.0 test material. It was noted that H.225.0 implements a subset of ITU-T recommendation Q.931 [22]. Q.931 has been developed by ITU-T in co-operation with ATM Forum. It is used in Integrated Services Digital Network (ISDN) signalling and a related protocol User to Network Interface (UNI) is used in Asynchronous Transfer Mode (ATM) signalling. Thus the potential impact of the H.225.0 test material containing Q.931 test would be vastly larger than intended. This raised questions on the linkage of protocol specifications and prompted research on protocol dependency.

During the creation of the test suite a new vulnerability domain was discovered. Initially the PROTOS team studied Q.931 as a part of the H.323 research, and not until some studies on ISDN it was discovered that the protocols shared a common connection control protocol. Later it was discovered that ATM also shares a related control protocol - User to Network Interface.

4. Analysis

> "When components of any system are highly interdependent, there is no such thing as a local fix."
>
> Andrew Hunt and David Thomas, 'The Pragmatic Programmer'

The PROTOS team has often been queried on the impact of the vulnerabilities found with the test material. Apparent from the LDAP, SNMP and H.225.0 cases, the impact assessment is not a trivial task due to different levels of abstraction of the vulnerabilities.

The assignment of abstraction levels to vulnerabilities is a problem previously tackled by the Common Vulnerabilities and Exposures (CVE) project. CVE researched the existing vulnerability taxonomies and presented the Common Vulnerability Enumeration standard that uses a single vulnerability abstraction level [17]. This approach may be useful from the point of view of a vulnerability database.

Looking back to the test materials, the concept of vulnerability meta levels emerges. Meta levels present a multilevel vulnerability taxonomy designed to categorise vulnerabilities based on their impacts. The basic idea of the taxonomy is that a vulnerability affecting a low-level concept should have a high meta level, and vice versa. The rationale behind the idea is that low-level concepts are more prevalent in the realm of software than high-level ones. Thus, a systematic error in the implementation of a low-level concept will result in vulnerabilities with a great impact, whereas the impact of corresponding high-level vulnerabilities may be limited to a single implementation.

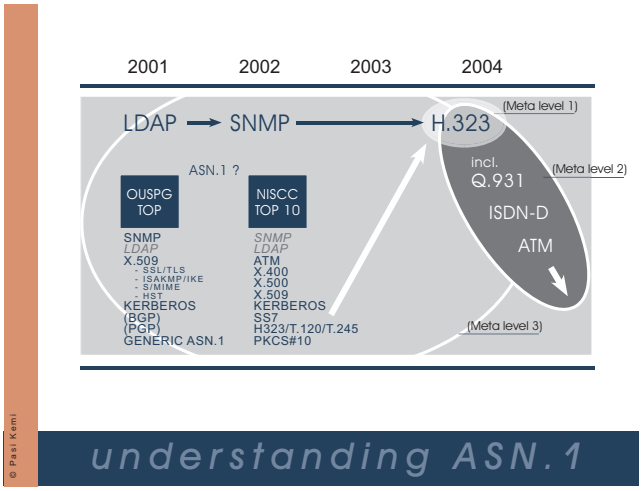

Figure 2. Vulnerability meta levels vs. PROTOS test materials

The following list presents the meta level's classification aided by the illustration of PROTOS test material relationships (see figure 2).

Meta level 0: As discussed earlier, traditional vulnerability research often focuses on a single vulnerability in a single implementation. Faults that usually lead to vulnerabilities are found from one or more versions of a single product. These vulnerabilities exhibit meta level zero.

Meta level 1: Vulnerabilities in multiple implementations of a single protocol are classified as meta level one. The PROTOS team set out to find these vulnerabilities as demonstrated by LDAP, SNMP and H.225.0 test materials when studied in isolation.

Meta level 2: Often protocols are created based on earlier specifications, or inherited from other protocol families that have a functionality similar to the desired. In this way multiple protocols or protocol families can share a common sub-protocol. If the shared protocol exhibits vulnerabilities, all the protocol families involved may have vulnerabilities. These shared vulnerabilities is what we perceive as meta level two. This is illustrated H.225.0 case where sub-protocol Q.931 turned out to be a shared protocol with at least ISDN and ATM (meta level two).

Meta level 3: Protocols also share a number of (en)coding schemes, encryption schemes, notations, and the like. If protocols are thought as the languages that protocol implementations communicate in, then notations that the protocols themselves are described in can be thought of as the alphabet of the language. In order for the implementation to handle a protocol message, it first has to parse the alphabet. A vulnerability in e.g. a coding scheme or in its common implementation can have unforeseeable scope and consequences. These we call meta level three vulnerabilities. In the test-materials this is reflected by ASN.1 chain, a shared scheme with numerous protocol families e.g. LDAP, SNMP, H.225.0 (subset of H.323) and others (meta level three).

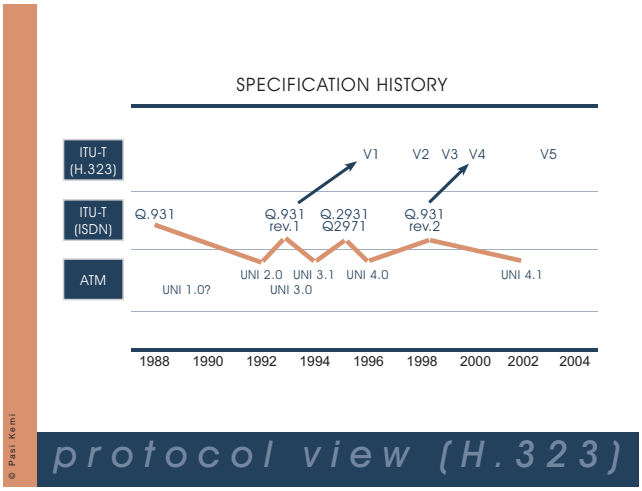

Figure 3. The specification history of Q.931 explicates its protocol dependencies

While meta level three vulnerabilities seem hard to grasp by any analytic means, it appears that meta level two vulnerabilities can be apprehended by charting protocol depen-

dency. Hitherto, the approach to protocol dependency in the PROTOS team has been a reactive one, dependencies have been taken into account only after they have surfaced in research. Post-mortem analysis on the H.225.0 vulnerabilities included a study on the history of Q.931 specifications. The results, presented in figure 3, clearly show the forks taken in the development of the specification, and the resulting dependencies to other protocol families. Had the analysis been performed in the early stages of test material development, it would have been an invaluable asset in impact assessment and test subject selection.

4.1. Impact of protocol dependency

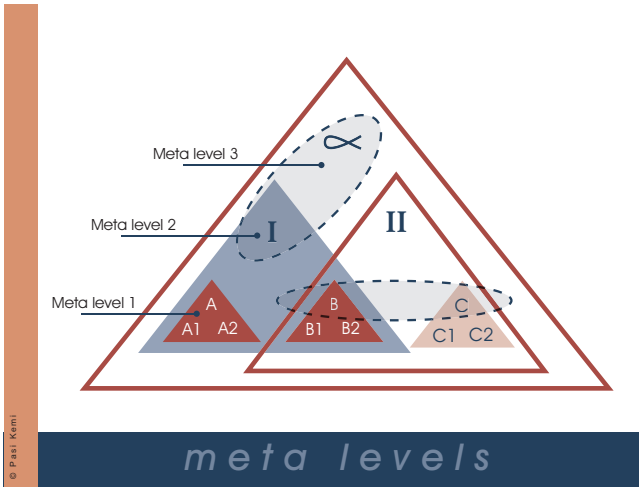

Figure 4. A scenario with different levels of dependencies

Figure 4 presents an example case on protocol dependency involving multiple implementations (A1, B1 ...), three protocols (A, B and C), two protocol families (I and II), and a higher level schema (α). It can be readily noted from the figure that the impact area of a vulnerability progresses geometrically as the vulnerability ascends meta levels of dependency.

However, not all dependencies result in such drastic increase in scope. The ellipses present the cases where entities of various meta levels exhibit certain dependencies between themselves only. The upper ellipse represents a dependency shared by notation α and protocol family I but not the protocol family II, and the lower ellipse a dependency between two individual protocols but not with the other protocols from family II.

This example advises that the impacts of protocol dependencies may be varied. Impact assessment calls for detailed analysis of the entities involved.

4.2. Causes of protocol dependency

Re-use of existing components (as-is or with only cosmetic changes) is a common habit of the IT industry, particularly in standardisation. Standardisation bodies such as ITU-T build new standards heavily based on the preexisting protocol portfolio. A significant factor behind this policy are the resources that have been used by the industry to implement the existing components. Re-use creates wider dependency on the shared components, and wildly increases the impact of the vulnerabilities related to these components.

Dependency through re-use also presents itself when identical code, methods or protocols appear in multiple implementations. This may occur due to shared legacy code, similar choices made during implementation, plain re-use or other factors. Identical structural foundations are used, albeit possibly for multiple purposes, and the affected implementations are subject to shared vulnerabilities.

In contrast, problems can arise when an implementation uses novel technologies instead of adopting well-tried and tested ones. This kind of development can have its benefits if proper care is taken. If not, it can lead to replicating the very errors that had plagued the present de facto standard in its early days. In this case, the new technology has merely reinvented the wheel and in a way becomes dependent on the technology it tried to recreate. It has been suggested that the WAP protocols are an exemplary case of the downsides of overlapping development [1].

4.3. Types of protocol dependency

If a protocol is shared among different protocol families and environments, it can be thought of as a subset of protocols that contain it. Similarly, a protocol can itself be a superset, i.e. it can use or contain a set of protocols that might in turn be shared by other protocols. Study on both of these dimensions is beneficial.

Charting which protocols incorporate the target protocol is useful for determining its scope of dependency and the impact of vulnerabilities in the protocol and its implementations. This was demonstrated with the case of the vulnerabilities in Q.931 message handling.

On the other hand, charting the protocols incorporated by the target protocol gives insight on its usage scenarios and on the services it affects. For example, H.225.0 contains units for call signalling and registration, admission and status (RAS). Therefore the affected services are H.323 terminals, gateways and gatekeepers providing admission control services. H.323 gateways provide protocol conversion service between different network and terminal types, which implicates the re-use of signalling data.

Additionally, protocols can form complex chains that

propagate data and/or control data between a source and a destination. For example, authentication data can travel from a client to an authentication database through a multitude of servers and authentication servers via various protocols. Some of the nodes in the chain will simply pass the data along but other nodes may try to interpret it. Thus, vulnerabilities may be triggered in the later parts of the protocol chain if the passed data is erroneous. [14] In this case, the vulnerability is a result of data propagation dependency.

Similarly, the passed data may represent control data for an implementation or a protocol along the chain. In the authentication example, the database server usually handles the data it receives as control data, and malicious authentication input by the user will trigger a vulnerability in the database. [14] This vulnerability can be thought to result from control propagation dependency.

Other types of protocol dependencies might be the result of incidental common component re-use between protocols. Dependent components may include interfaces, objects, parsing mechanisms and libraries.

5. Discussion

Most current IT systems implement a number of protocols, most of which they require for normal functionality. In effect, the system can be communicated with by a number of means from various locations, and it parses diverse network data. This makes the system, as well as other systems on the network, dependent on the implemented protocols in a multitude of ways. If any protocol, or its implementation, that the system is dependent on should exhibit vulnerabilities, all the operations of the system may be compromised.

As discussed earlier, risk management of a system is difficult due to the changing nature of information technology. A protocol, however, is more a static entity. Many successful protocols have been in use for decades, and changes in specifications are relatively scarce. This makes it easier to include protocols in risk assessment.

Protocol dependency is a subtle view on the technologies that form the information infrastructure. It complements the known views and offers new insights on technological dependency. Understanding protocol dependency aids the assessment of the dependencies of an infrastructure and the impact a vulnerability would have on it. It guides the coordination of resources in response to vulnerabilities in the infrastructure and allocation of resources towards effective research and pro-active work on improving the robustness of the infrastructure. This work includes the current vulnerability management work as well as crisis scenario planning "what-if" questions. The benefits are both technical and managerial.

Risk assessment methodologies have a need for complementary modelling tools [15] [25]. A modelling methodology for protocol dependency could be such a tool. The scope of protocol dependency is wide, and its study would require advanced data gathering methods along with visualisation methods to present the results in easily grasped and compressed forms.

Thus, the proposed methods for researching protocol dependency include:

1. Expert interviews augmenting the plain data mining approach to the following activities
2. Summaries of relevant technical specifications, their relations to other specifications and historical dependencies
3. Surveys of the public attention on the security of different protocols and protocol implementations
4. Surveys on the prevalence of protocol implementations and their usage environments

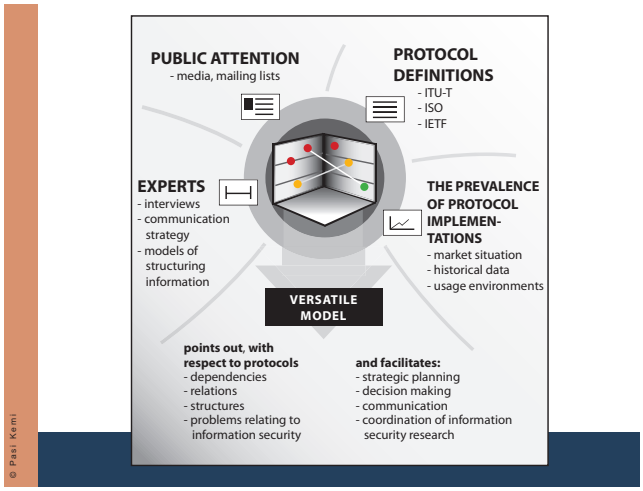

Figure 5. Data sources and research results of the proposed model

The accumulated data is presented with visual models that bring up different aspects from the data related to protocol dependency and security (see figure 5). As the model is highly visual, it can additionally be used as a communication method between researchers and other actors.

6. Conclusions

Systematic vulnerability assessment of information systems has proved that dependencies are not uncommon among protocols and their implementations. The study of protocol dependencies benefits vulnerability assessment,

and it should be performed proactively during the development and risk assessment of software. Furthermore, protocol dependency provides a subtle view on the technologies that form the information infrastructure. It complements the known views and offers new insights on technological dependency. The benefits are both technical and managerial.

Extrapolating from the experiences of a complex case, this new dependency dimension can be modelled. The model will benefit from going beyond a narrow technical view.

References

[1] M. Banahan. Impressions of using WAP/WML. http://ebusiness.gbdirect.co.uk/case_studies/wapimpressions.html; accessed June 27, 2005.

[2] T. Brown, W. Beyeler, and D. Barton. Assessing infrastructure interdependencies: the challenge of risk analysis for complex adaptive systems. *Int. J. Critical Infrastructures*, 1(1):108–117, 2004.

[3] CERT/CC. CERT advisory CA-2002-03 multiple vulnerabilities in many implementations of the Simple Network Management Protocol (SNMP). http://www.cert.org/advisories/CA-2002-03.html; accessed June 29, 2005.

[4] CERT/CC. CERT/CC overview: Incident and vulnerability trends. http://www.cert.org/present/cert-overview-trends/module-2.pdf; accessed June 29, 2005.

[5] CERT/CC. CERT/CC statistics 1988-2005. http://www.cert.org/stats/cert_stats.html; accessed June 13, 2005.

[6] R. Clarke. Looking at vulnerability issues in addressing cyber security, 2000. http://www.ncs.gov/nstac/march2002/nsatc_cybersecurity.html; accessed June 29, 2005.

[7] CVE. CVE abstraction content decisions: Rationale and application. http://www.cve.mitre.org/cve/cd_rationale_application.html#id_bugs; accessed June 29, 2005.

[8] M. Dunn, A. W. Isabelle Wigert, and J. Metzger. *International CIIP Handbook 2004 An Inventory and Analysis of Protection Policies in Fourteen Countries*. Comprehensive Risk Analysis and Management Network, 2004. http://www.isn.ethz.ch/crn/_docs/CIIP_Handbook_2004_web.pdf; accessed June 13, 2005.

[9] A. V. Gheorge and D. V. Vamanu. Complexity induced vulnerability. *Int. J. Critical Infrastructures*, 1(1):76–85, 2004.

[10] A. V. Gheorghe and L. Mili. Editorial: In risk management, integrating the social, economic and technical aspects of cascading failures across interdependent critical infrastructures. *Int. J. Critical Infrastructures*, 1(1):1–2, 2004.

[11] M. Harrop. *Creating Trust in Critical Network Infrastructures: Canadian Case Study*. International Telecommunication Union, 2002. http://www.itu.int/osg/spu/ni/security/docs/cni.07.doc; accessed June 13, 2005.

[12] IWCIP2005. Call for papers. http://www.iwcip.org/2005/CfP_WS2005.html; accessed June 13, 2005.

[13] R. Kaksonen, M. Laakso, and A. Takanen. Vulnerability analysis of software through syntax testing, 2000. http://www.ee.oulu.fi/research/ouspg/protos/analysis/WP2000-robustness; accessed June 29, 2005.

[14] J. Kenttälä. Exploiting communication patterns in complex information networks. Master's thesis, Department of Information Processing Science at University of Oulu, 2005. http://www.ee.oulu.fi/research/ouspg/frontier/; accessed June 29, 2005.

[15] A. Koubatis and J. Y. Schönberger. Risk management of complex critical systems. *Int. J. Critical Infrastructures*, 1(2/3):206–208, 2005.

[16] M. Laakso, A. Takanen, and J. Röning. The vulnerability process: A tiger team approach to resolving vulnerability cases. In *Proceedings of the 11th FIRST Conference on Computer Security Incident Handling and Response. (Brisbane, Australia. June 13-18, 1999)*, 1999. http://www.ee.oulu.fi/research/ouspg/protos/sota/FIRST1999-process/; accessed June 29, 2005.

[17] D. E. Mann and S. M. Christey. Towards a common enumeration of vulnerabilities. http://www.cve.mitre.org/docs/cerias.html; accessed June 29, 2005.

[18] M. Masera and M. Wilikens. Interdependencies with the information infrastructure: Dependability and complexity issues. In *Paper at the 5th International Conference on Technology, Policy, and Innovation. (Ispra, 26-29 June, 2001)*, 2001. http://www.delft2001.tudelft.nl/paper%20files/paper1168.doc; accessed June 13, 2005.

[19] R. Mock and M. Corvo. Risk analysis of information systems by event process chains. *Int. J. Critical Infrastructures*, 1(2/3):247–257, 2005.

[20] NISCC. NISCC vulnerability advisory 006489/H323 vulnerability issues in implementations of the H.323 protocol. http://www.niscc.gov.uk/niscc/docs/re-20040113-00387.pdf; accessed June 29, 2005.

[21] OUSPG. PROTOS - security testing of protocol implementations. http://www.ee.oulu.fi/research/ouspg/protos/; accessed June 29, 2005.

[22] OUSPG. PROTOS test-suite: c07-h2250v4. http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/h2250v4/; accessed June 29, 2005.

[23] PCCIP. *Critical Foundations: Protecting America's Infrastructures*. President's Commission on Critical Infrastructure Protection, 1997.

[24] PITAC. *Cyber Security: A Crisis of Priorisation*. National Coordination Office for Information Technology Research and Development, February 2005.

[25] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdepencencies. *IEEE Control Systems Magazine*, 21(6):111–25, December 2001.

[26] J. Röning and J. Eronen. Software considered harmful: Why software is insecure, 2002. http://www.ee.oulu.fi/research/ouspg/protos/sota/CorpSec2002/; accessed June 29, 2005.

[27] G. Stoneburner, A. Goguen, and A. Feringa. *Risk Management Guide for Information Technology Systems*. National Institute of Standards and Technology, July 2002. http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf; accessed June 13, 2005.

[28] A. Wenger, J. Metzger, and D. Myriam. *International CIIP Handbook: An Inventory of Protection Policies in Eight Countries*. Center for Security Studies and Conflict Research, Seilergraben 45 49 ETH Zentrum SEI CH-8092 Zurich Switzerland, 2002. http://www.isn.ethz.ch/crn/_docs/CIIP_Handbook_2002_bw.pdf; accessed June 13, 2005.