DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
DEGREE PROGRAMME IN INFORMATION ENGINEERING

A COLLABORATIVE METHOD FOR ASSESSING THE DEPENDENCIES OF CRITICAL INFORMATION INFRA-STRUCTURES

Author _____
Juhani Eronen

Supervisor _____
Juha Röning

Accepted _____ / _____ 2006

Grade _____

Eronen J. (2006) A collaborative method for assessing the dependencies of critical information infrastructures. Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland. Master's thesis, 79 p.

ABSTRACT

The critical infrastructures have been penetrated by information systems. The very basis that we depend on has become technologically entangled. New types of vulnerabilities that evade current risk analysis methods have emerged. Protocol dependency is one of the key culprits.

This work focuses on collaborative management of protocol related knowledge, which is required in order to understand and mitigate the risks that emerge from protocol dependency. The PROTOS-MATINE method was developed to illustrate inheritances and hidden links between protocols from multiple angles. Charting these linkages requires efficient knowledge management techniques. The semantic Graphingwiki tool was developed to support this process.

The protocol views created by Graphingwiki have been used in various stages of protocol-related vulnerability work. These visualisations proved to be an effective aid for apprehending protocol environments. They highlighted problem areas, such as protocols that are abundantly depended upon, baroque relations between protocol families, and the inherent complexity of modern networks. Moreover, initial experiences on the applicability of Graphinwiki for purposes outside its intended domain of application are very encouraging.

Keywords: critical infrastructure, risk assessment, vulnerability, dependency, protocol, semantic Wiki

Eronen J. (2006) Yhteistyömenetelmä kriittisten tietoinfrastruktuurien riippuvuuksien arviointiin. Oulun yliopisto, sähkö- ja tietotekniikan osasto. Diplomityö, 79 s.

TIIVISTELMÄ

Tietotekniikka on tunkeutunut syvälle kriittiseen infrastruktuuriin. Monet yhteiskunnan perustoiminnot ovat riippuvaisia tietojärjestelmistä. Tämä on synnyttänyt uudentyyppisiä haavoittuvuuksia, joita ei pystytä huomioimaan nykyisillä riskianalyysimenetelmillä. Eräs suurimpia syitä tähän tilanteeseen ovat protokollariippuvuudet.

Tämä työ keskittyy protokolliin liittyvän tiedon yhteiseen käsittelyyn, jota tarvitaan protokollariippuvuudesta johtuvien riskien ymmärtämiseen ja hallintaan. PROTOS-MATINE-menetelmä kehitettiin havainnollistamaan protokollien perimää ja esittämään monipuolisesti piileviä kytköksiä. Näiden kytkösten kartoittaminen vaatii tehokkaita tiedonhallintatekniikoita. Tämän prosessin tukemiseksi tässä työssä kehitettiin semanttinen Graphingwiki-työkalu.

Graphingwikin luomia näkymiä käytettiin protokolliin liittyvien haavoittuvuusprosessien useissa eri vaiheissa. Näkymät osoittautuivat tehokkaaksi menetelmäksi protokollaympäristöjen hahmottamiseen. Ne korostavat näiden ympäristöjen ongelma-alueita, kuten protokollia, joihin viitataan runsaasti. Näkymät myös esittävät rönsyilevät yhteydet eri protokollaperheiden välillä sekä paljastavat nykyisten tietoverkkojen luonteenomaisen monimutkaisuuden. Alustavat kokemukset Graphingwikin soveltuvuudesta sen alkuperäisestä tarkoituksesta poikkeavaan käyttöön ovat hyvin rohkaisevia.

Avainsanat: kriittinen infrastruktuuri, riskinarviointi, haavoittuvuus, riippuvuus, protokolla, semanttinen Wiki

TABLE OF CONTENTS

FOREWORD

This masters thesis represents a partial summary of seven years of work in the Oulu University Secure Programming Group. It has been an incredible journey, kudos to all my colleagues amidst the years.

Parts of this work have appeared in the papers “A Case for Protocol Dependency”, presented at the First IEEE International Workshop on Critical Infrastructure Protection, and “Graphingwiki - a Semantic Wiki Extension for Visualising and Inferring Protocol Dependency”, submitted to the First Workshop on Semantic Wikis in the 3rd Annual European Semantic Web Conference (ESWC).

I would like to thank the people who contributed to this masters thesis. A special thanks goes to my supervisor Juha Röning, for guidance and patience. I would also like to thank Marko Laakso, Jani Kenttälä, Christian Wieser, Toni Alatalo, Kati Karjalainen and Mikko Hiltunen for remarks on this manuscript, and Aki Helin, Joachim Viide and Erno Kuusela for help with coding. Finally, my fiancée Päivi Mäkinen, for being there.

Oulu, Finland 3.5.2006

Juhani Eronen

ABBREVIATIONS

3GPP	3rd Generation Partnership Project, a GSM-based consortium advocating standardization for mobile communications
ANSI	American National Standards Institute
AusCert	Australia's National Computer Emergency Response Team
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
B-ICI	A specification for broadband switched virtual connection between public networks
BER	Basic Encoding Rules
CGI	Common Gateway Interface
CSV	Comma-Separated Values
CVE	Common Vulnerabilities and Exposures
CVS	Concurrent Versions System
ETSI	European Telecommunication Standards Institute
FDDI	Fiber Distributed Data Interface
GSM	Global System for Mobile Communications
GUI	Graphical User Iterface
H.225.0	Call Signalling and RAS in H.323
H.323	The ITU-T standard suite for digital video conferencing over packet-switched networks
HTML	HyperText [sic] Markup Language
HTTP	Hypertext [sic] Transport Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IEFT	Internet Engineering Task Force
IRC	Internet Relay Chat
ISDN	Integrated Services Digital Network
ISO	International Organisation for Standardisation
ISUP	ISDN User Part
IT	Information Technology
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardisation Sector
LAN	Local Area Network
LANE	LAN emulation over ATM network
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Message Service
MPLS	Multi-Protocol Label Switching
N3	Notation 3 for expressing RDF data
NISCC	The National Infrastructure Security Co-ordination Centre of the United Kingdom
OSI	Open Systems Interconnection
OUSPG	Oulu University Secure Programming Group

OWL	Web Ontology Language
PROTOS	The Security Testing of Protocol Implementations project
Q.931	ISDN UNI layer 3 specification for basic call control
PDF	Portable Document Format
RAR	Roshal Archive
RAS	Registration, Admission and Status
RCS	Revision Control System
RDF	Resource Description Framework
RDFS	RDF Schema
RFC	Request for Comments
RS-232C	A serial data communications protocol
RSS	RDF Site Summary (version 1.0) or Really Simple Syndication (version 2.0)
SIP	Session Initiation Protocol
SME	Small and Medium sized Enterprises
SNMP	Simple Network Management Protocol
SVG	Scalable Vector Graphics
UMTS	Universal Mobile Telephone Service
UNI	User to Network Interface
URI	Uniform Resource Identifier
WAP	Wireless Application Protocol
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
Z39.50	A standard for the communication of two computers for the purpose of information retrieval

1. INTRODUCTION

During the past few decades, critical infrastructures have introduced information systems for reasons of reduced costs, increased efficiency, and new functionality. As a result, modern society as a whole depends on various computer systems for the continuous operation of many of its functions. These systems have become increasingly complex and federated across governmental, corporate, and national boundaries. [1][2]

A common maxim of quality control states that the total number of flaws in information systems grows linearly to complexity [3]. These flaws introduce failures and security problems that have a significant effect on society. As complex critical systems have become abundant, new risks have emerged that require sophisticated methods for their analysis and management [4].

Risk analysis methods for information systems have been introduced by technological nations and many corporations. Most of these methods operate on the operational and policy levels, taking into account inter-infrastructural linkages, best practices and other issues that affect the systems in an infrastructure. [5] [6]

However, there are few methods that truly take into account the complexity of computer systems as a major enhancer of the risk. Besides inflating the code base and thus introducing more flaws, complexity results in dependencies amongst the system and with other systems [7]. These dependencies create new types of vulnerabilities that are left unnoticed by current risk analysis methods.

This work presents the PROTOS-MATINE method and the Graphingwiki tool that provide risk analysis with new tools for fathoming complex environments. A key issue surrounding the subject is protocol dependency. Most current computer systems implement a number of protocols, which they use for communication via various interfaces. Different types of linkage between these protocols introduce vulnerabilities that threaten critical infrastructures. This work focuses on efficient management of protocol related knowledge, which is required in order to understand and mitigate the risks that emerge from protocol dependency.

The method and the tool are used to gather data from technical specifications and from experts of different protocol environments. The accumulated data is then visualised, bringing up different aspects from the data related to protocol usage, dependency and security. The resulting views can additionally be used as a communication method between researchers, managers, and other operatives. Inference is used as a method of gaining profound insight to dependency chains and networks.

This work argues that a similar knowledge management approach would also be effective for other domain-specific tasks where an universal topical scope and some of the other stumbling blocks of semantic technologies are not an issue [8] [9].

1.1. Critical Information Infrastructure

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." [10]

Vulnerabilities infest information technology. The number of information technology vulnerabilities tracked by the information security watchdog CERT/CC[1] 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 [11].

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 [12]. Vulnerabilities manifest in new threats that generate risks for industries, which in turn try to protect their assets and operations with the help of risk management. However, vulnerabilities cannot be mitigated efficiently without first understanding the dependencies involved [4].

Technological dependency has been investigated before in various studies and programs (see for example [4], [6], [12], [13], [14] and [15].) 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 [7]. Instead, efficient risk management of dependent technologies would require multifaceted analysis methods for different aspects of technological dependency [4]. The science of dependencies is relatively immature and many dependencies are not yet understood or even uncovered [13].

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 types of 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 disclosed 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. Previous work by the Oulu University Secure Programming Group (OUSPG) has derived 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.

[1] http://www.cert.org/

1.2. Collaborative Knowledge Management

Charting the linkages of protocols is a difficult subject that requires efficient knowledge management techniques. Hypertext has been advocated as a solution for problems resulting from information complexity [16]. In recent years, Wikis and the semantic web have become the state of the art methods for the management of information [8] [19]. Wikis have proven to be an effective means for the collective gathering and editing of bodies of data ranging from encyclopaedia to bug tracking and journals [18]. Semantic web is envisioned as a universal medium for data exchange and as a tool to manage the interconnection of information, enabling automated analysis of data [17].

Both of the technologies have strong selling points: Wikis enable collaborative, open, evolutionary, and easy modification of data, and the semantic web employs Resource Description Framework (RDF), a powerful yet relatively simple language for representing information about World Wide Web (WWW) resources [20].

This work introduces Graphingwiki, a Wiki extension that aims to enable knowledge engineering in Wikis by sidestepping the complexity of semantic technologies. Users introduce semantic data into the Wiki by simply tagging pages and page links with words or phrases that sound suitable to them.

Interactive visualisation is proposed as a method for understanding the relations of information on the Wiki pages. Visualisations can be used to navigate the Wiki, and they include facilities for filtering out non-relevant data. This enables the quick derivation of a general view on any desired topic or entity.

Furthermore, Graphingwiki includes some logic reasoning capabilities for refining specific knowledge from the Wiki tags. Special Wiki pages can include rules that lead to new conclusions about specific tags, and the resulting data can be queried for sets of pages and tags that fulfil the premises of the query. This presents a fine-grained method for discovering relations amongst the wealth of data. Visualising the results of these queries can further clarify the derived relations.

1.3. Contents

The following chapter presents a background on the main issues of critical information infrastructure protection and knowledge management. Analysis on these issues forms the requirements for Graphingwiki. The existing applications for collaborative knowledge management are surveyed based on these requirements. Chapter 3 introduces the concept of protocol dependency and the PROTOS-MATINE method for managing these dependencies. Data gathering processes of the method are explained generally and in the context of Graphingwiki. Chapter 4 first lays out the detailed design of the tool and continues to trace its prototyping phase. Chapter 5 will discuss the results along with their value and limitations, and will present outlines for future research and development of the tool. Finally, Chapter 6 will conclude the work.

2. BACKGROUND

The first section of this chapter presents the crucial role that vulnerabilities constitute for the risk analysis of critical infrastructures. Analysis on the nature and prevalence of vulnerabilities is carried out. The second section describes a set of disclosed vulnerabilities in well-established protocols. These vulnerabilities had substantial effects on critical infrastructures as they implicated faults in a number of subtly related protocols. The importance of finding these relations proactively is thus illustrated.

As the analysis of protocols requires efficient tools, the third section presents some of the current methods for the management and visualisation of knowledge. The fourth section fleshes out the requirements for efficient knowledge management in the context of protocol analysis, and peruses the applications with similar features. Finally, the fifth section summarises the chapter.

2.1. Critical Information Infrastructure Protection

Faults in technology result in failures and vulnerabilities that produce ill effects on its users, the most prominent of which are critical infrastructures. These effects are mitigated by risk analysis, which requires careful analysis of the technologies involved. As technology has grown in sophistication and complexity, it has become dependent on other technologies in obscure ways, resulting in equally obscure vulnerabilities. Meanwhile the most common vulnerabilities continue to surface in most software at a steady rate. This makes the study of vulnerabilities a noteworthy subject in the field of critical information infrastructure protection.

2.1.1. Risk Management

> "The essence of risk management lies in maximizing the areas where we have some control over the outcome while minimizing 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', p.199

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. [21] [5]

A crucial part of risk management is the risk assessment phase, in which the relevant processes and systems are identified, and 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. [21]

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 [6]. 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 [13].

The dependencies and interdependencies can be seen to embody multiple dimensions such as their environment, feedback mechanisms, or failure types [5]. The degree and type of the dependencies strongly influence the operating characteristics of the affected infrastructures. A vulnerability in linked infrastructures can cause failures due to a common cause, cascading failures, or even escalating failures among the affected infrastructures [13].

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. [2]

2.1.2. Dependencies

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 [14] [5].

However, risk analysis of a single Information Technology (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 [15] [12]. 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 [12]. The current trend of ubiquitous interconnectivity of IT systems has resulted in widespread vulnerability, where continuously increasing levels and varieties of attacks have emerged [22].

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 [13]. Therefore the research on vulnera-

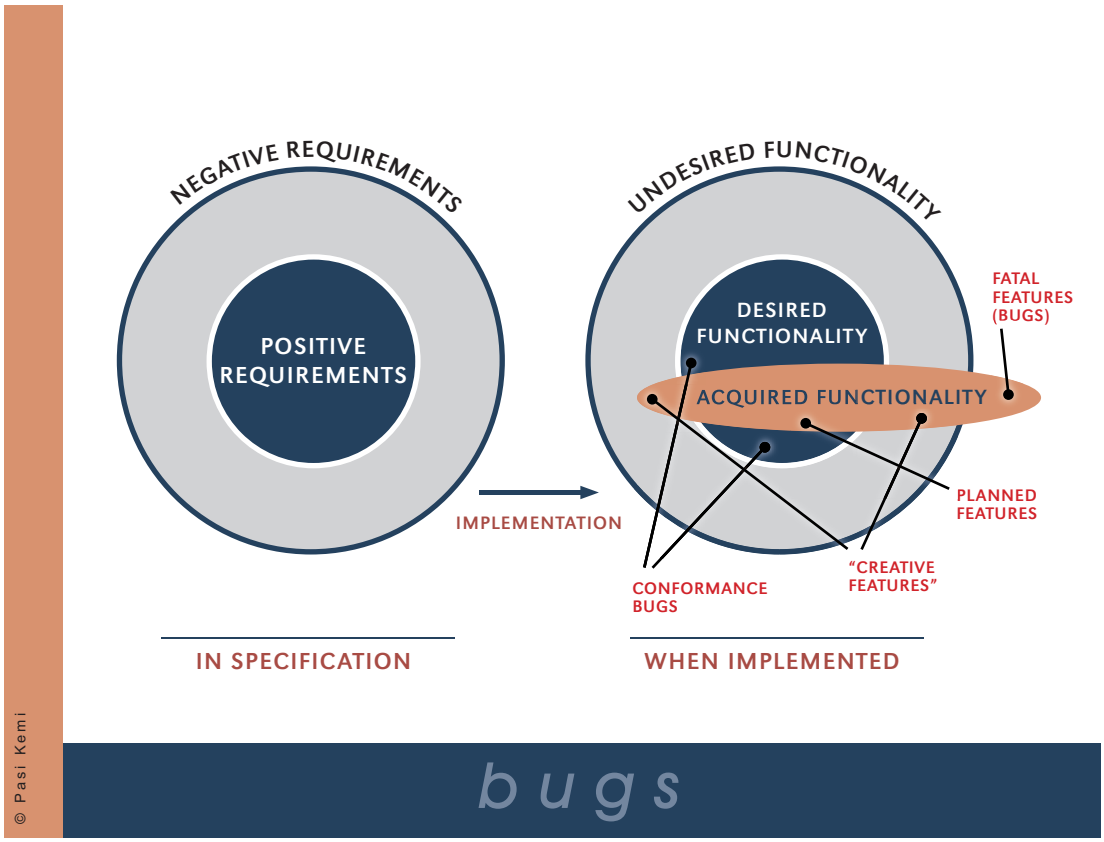

Figure 1. Plans vs. reality in implementation.

bilities is a decisive topic in mitigating the risks related to IT systems and ultimately to critical infrastructures.

2.1.3. Vulnerabilities

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 a 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.

Theory meets practice when the system undergoes implementation. As a result, we get both 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, that is, features that can be used to achieve functionality that neither the requirements, designer, nor even the programmer anticipated. 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 [22].

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 [24].

Traditional vulnerability research has proceeded mostly in a very reactive fashion, addressing vulnerabilities as they are discovered in a “penetrate & patch” paradigm [25]. 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. 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 disclosed vulnerabilities [26].

2.2. Vulnerability Assessment of Protocols

OUSPG has claimed that programming errors leading to vulnerabilities are systematic, and that many of those vulnerabilities could be eliminated by systematic testing [27]. In the PROTOS[1] 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.

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. [28]

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.

[1] http://www.ee.oulu.fi/research/ouspg/protos/

2.2.1. Linkage of Protocols

This chapter describes three of the test material suites published by the PROTOS team. These materials produced a large quantity of vulnerability data and revealed dependencies involved in information system vulnerabilities. The test suites, 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 from the test materials confirmed the claims stated by the PROTOS project: 80% of the products tested within the project failed due to exploitable flaws [27]. 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 National Infrastructure Security Co-Ordination Centre of the United Kingdom (NISCC). The advisory for the SNMPv1 test suite alone has statements from 140 vendors [29].

After testing the material for LDAP, it became suspect for the PROTOS team that there are implementation level vulnerabilities in various Abstract Syntax Notation 1 (ASN.1) parsers, which are prevalent in protocol implementations. Initial analysis and observations supported this. Thus, the most significant impact caused by the test materials reached far beyond the scope of the protocols involved. For example the LDAP and SNMP protocols both use a syntactic notation called 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 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 suite 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 critical national infrastructure perspective. This served as guidance for further research, and prompted the PROTOS team to generate test material for H.225.0. [30]

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 [31]. 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." [32]

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 International Telecommunication Union (ITU-T) recommendation Q.931 [33]. 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 tests 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.

2.3. Knowledge Management

Charting the linkages of protocols is a difficult subject that requires efficient knowledge management techniques. The management of organisational knowledge is currently seen as a key asset to the success of an organisation, and of the economy as a whole. Knowledge management includes tools, processes and practices necessary for the capture, transfer and reuse of the knowledge assets within the organisation.

Historically, a number of different technologies such as expert systems, knowledge bases and document management systems have been used as tools to enable knowledge management. However, the advent of the Internet has introduced numerous tools for collaborative handling of information. This section summarises some of the current technologies to enable knowledge management.

2.3.1. Semantic Web

The Semantic Web is a W3C project that aims to augment the contents of the World Wide Web with computer-understandable meaning, i.e. semantics [17]. A semantic WWW page contains machine-readable descriptions that add meaning to its content, thus enabling computers to process the page information in a more efficient manner.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rss="http://purl.org/rss/1.0/">

 <rss:item
  rdf:about="http://www.boingboing.net/2006/04/12/many_better_ways_to_.html">
  <rss:title>Many better ways to tie your shoes</rss:title>
  <rss:link>http://feeds.feedburner.com/boingboing/iBag?m=670</rss:link>
 </rss:item>

</rdf:RDF>
```

Figure 2. Semantic data as expressed in the RDF/XML notation.

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rss:  <http://purl.org/rss/1.0/> .

<http://www.boingboing.net/2006/04/12/many_better_ways_to_.html>
 rss:title ‘‘Many better ways to tie your shoes’’ .
<http://www.boingboing.net/2006/04/12/many_better_ways_to_.html>
 rss:link <http://feeds.feedburner.com/boingboing/iBag?m=670> .
```

Figure 3. Semantic data as expressed in the N3 notation.

The Semantic Web concept is in no way unique in its functions, but it provides standard technologies that heighten interoperability and ease of implementation.

The standards comprising the Semantic Web include Extensible Markup Language (XML), XML Schema, RDF, RDF schema (RDFS) and Web Ontology Language (OWL). XML provides the syntax for documents, whose structure can in turn be constrained with its schema. RDF is a simple data model for referring to resources and their relations. RDF Schema and OWL provide richer vocabulary for describing properties and classes of RDF resources. [34]

RDF is the heart of the proposed standards framework. It consists of subject-predicate-object triples that are used to make statements about resources. An RDF resource can basically be anything that has a Uniform Resource Identifier (URI), so it can be used to refer to any web resource. The triples describe either relationships between two resources, the subject and the object, or an aspect of the subject, the value of which is specified by the object. The predicate is a resource that the relationship or aspect describes. [20]

RDF can be expressed in various notations, of which RDF/XML is the most verbose. The RDF/XML structure presented in Figure 2 contains a single article from the RDF Site Summary (RSS 1.0) feed of the BoingBoing [2] weblog. The article is specified by the rdf:about URI, and it is stated to have a title aspect with a value and a link relationship with the feed item itself. The item, link and title types are specified according to the RSS 1.0 namespace.

Another popular notation for expressing RDF data is Notation 3 (N3), which is designed to be more compact and readable than RDF/XML [35]. Figure 3 shows the corresponding N3 rendering of the semantic data.

RDF has become popular even outside the context of the Semantic Web due to its simple data model and its ability to embody abstract, yet disparate concepts. The model is generally seen to be better suited to knowledge representation than most of

[2] http://boingboing.net

the previously used knowledge models. RDF triples are most often evaluated with the help of queries that can be used for analysis and rudimentary reasoning from the data.

When supplemented with RDFS and OWL, RDF is often used for creating and managing ontologies [8]. In this context, an ontology can be thought of as a data model representing an area of knowledge. The model consists of the classes of objects in the domain, the attributes of the objects and the relationships between the classes and the attributes. Effectively, an ontology defines a domain by introducing the terms that are used when referring to the objects in the domain, along with the rules for reasoning about such objects.

2.3.2. Wikis

A Wiki is a WWW site that allows its users to edit its content in an straightforward manner [8]. In essence, a Wiki is a simplification of the usual process of WWW site publication that eases and accelerates collaborative editing. Rapid development is also the property that has earned Wiki its name, which has been derived from the Hawaiian word “wiki wiki”, most commonly used to mean “quick” or “fast”.

The first Wiki was WikiWikiWeb [3], established by Ward Cunningham in 1995. He created the Wiki concept along with its first implementation. Since then there has been a plenitude of Wiki implementations, but a number of common functionalities have remained, as summarised in the following.

Markup Although Wiki pages are normally rendered to HTML and viewed with browsers like normal WWW pages, the rendering of the pages is controlled by simple text markup.

Linkage New articles can be created by linking to them, correspondingly empty pages can be linked to.

Iteration All pages with the possible exception of special system pages can be edited, often without any registration. Locking mechanisms prevent conflicting simultaneous edits.

History Previous versions of pages are saved and can be easily inspected. Changes between versions can be tracked and reverted.

Wikis have been a great success for a variety of purposes including software documentation[4], software development issue tracking[5], encyclopaedia[6], and corporate intranets [7].

As knowledge repositories Wikis have adopted the ideology that each Wiki page is titled with sufficient precision to represent a single, well-defined concept. The used open collaboration style may spawn inconsistency and redundancy that is managed by the constant revision from the user base. Although the state of a Wiki at any point in time is undetermined, the aim is at the eventual convergence of its content.

[3] http://c2.com/cgi/wiki
[4] http://wiki.apache.org/jakarta/
[5] http://www.edgewall.com/trac/
[6] http://www.wikipedia.org
[7] http://twiki.org/cgi-bin/view/Main/TWikiSuccessStories

2.3.3. Semantic Wikis

Combining the approaches and techniques of Wikis and semantic web has met little success. The little support traditional Wikis offer for semantic data usually culminates in page categories and different kinds of comment tags. Semantic web tools are usually single-user oriented and their operation often requires expert skills, which makes knowledge engineering challenging for domain experts. [36] [37] [8]

Wikis have the strength that they focus on the structure of the data instead of its presentation. Wiki users are accustomed to creating, linking and tagging content, which represent the bare minimum requirements for taking advantage of semantics. Adding semantic features to Wikis offers a smooth transition for exploiting different layers of knowledge. [38]

The bare minimum functionality for semantic capabilities in a Wiki includes the implementation of a small but functional subset of RDF. This follows the Wiki way of doing the simplest thing that could possibly work [39]. RDF resources are represented on a Wiki page as tagged links and tagged page data. Together the page tags and the link tags create RDF statements of the forms `<page> <tag> <linked page>`, `<page> <tag> <URI resource>` and `<page> <tag> <tag value>`.

The tags of a minimal semantic system represent a flat namespace and do not have a hierarchy of any kind. In a way, this method of adding semantic data resembles folksonomies such as del.icio.us [8]. Tagging is simple and unrestrained as it aims for easy diffusion in the user base. Existing mechanisms, such as different kinds of linking, category pages and macros, are utilised as much as possible. Users may freely select the tags they use, which thus sacrifices consistency for practicality. This approach can prove more useful than forcing any predefined tagging schema [40] [41].

A Wiki functions as its own ontology, formed by all the tags in the Wiki's pages [42]. Each descriptive tag is assigned a page of its own so that terms can be defined and refined in the Wiki itself. The resulting ontologies are expressive to humans but lack the complexity and formality required for elaborate machine-processable constraints on the page data. This does not present a hindrance for knowledge management — in fact, the most successful knowledge models tend to be very simple and specific [8].

2.3.4. Visualising Knowledge

In mathematics, a graph is a pair $G = (V, E)$ of sets ($V \cap E = \emptyset$) satisfying $E \subseteq [V]^2$. As graph theory suffers from multiple conflicting terminologies, the terminology used in this work is defined as follows: the elements of V are called nodes, while the elements of E are the edges, each of which connects two nodes. The usual way to picture a graph is to draw a dot for each node and to draw a line between two nodes if the graph contains an edge between them. [43]

Graphs are an intuitive knowledge representation format, and thus many knowledge models have graph representations. This section introduces some of these representation formats and introduces a new one to be used in the Graphingwiki implementation.

[8] http://del.icio.us/

RDF data can be visualised with directed labelled graphs. An RDF graph represents RDF resources and literals as nodes and the types of their relations as the edges. All constituents of the graph are considered equal in that all resources can be depicted as nodes that can in turn have attributes and relations. Figure 4 depicts the RDF graph representation of statements defined previously in this section.

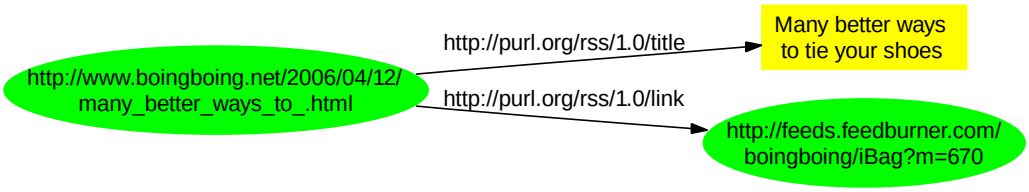

Figure 4. Statements represented with an RDF graph.

Conceptual graphs are a knowledge representation language consisting of concepts and their relations [44]. The concepts can be used to represent, for example, entities, attributes, states, or events and the relations can represent any interconnection between the concepts. In conceptual graphs, both the concepts and their relations are represented as nodes. Edges represent the connections of varied degrees between concepts and relations. These two sets of nodes are bipartite, i.e. concepts are only connected to relations and vice versa. Thus, contrarily to RDF graphs, the relations cannot have attributes and relations of their own. The conceptual graph representation of the statements are shown in Figure 5.

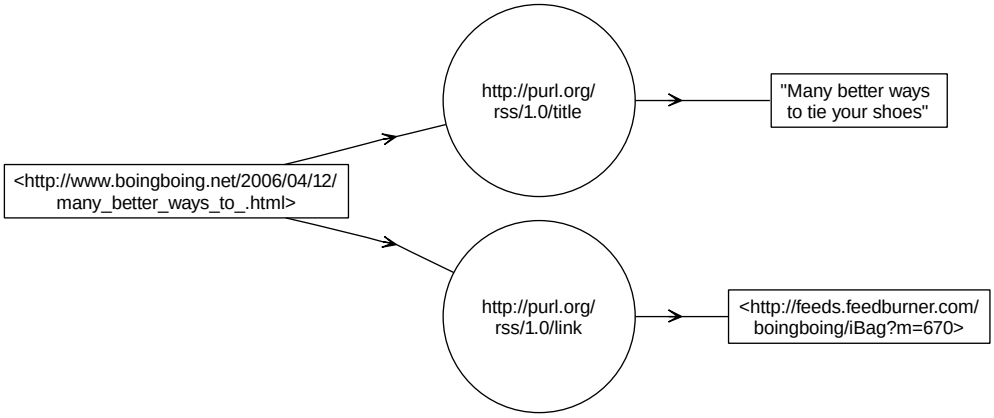

Figure 5. Statements represented with a conceptual graph.

While the previously introduced visualisation techniques focus on presenting all the data about the knowledge at once, an interactive and exploratory visualisation method is presented to reduce the visual clutter occasionally attributed to these methods. Exploratory visualisation is an often used technique in social network analysis, a branch of sociology developed from the social science of sociometry and mathematical graph theory. It has a heavy emphasis on using graphs, called sociograms, to gain insight into social relations between people and organisations. [45]

Graphingwiki visualises the semantic relations of a Wiki page, representing the concepts presented in the Wiki as graphs. Wiki pages and other resources are shown as nodes of the graph, while the edges correspond to the links between the pages or other

resources. The link tags are visualised with edge colours while the page tags can respectively be seen from the colours of the nodes. These colours are explained in a separate legend graph. Alternatively, the values of page tags can be shown by arranging all the nodes in respect to the tag values. Figures 6 and 7 illustrates the visualisation style.

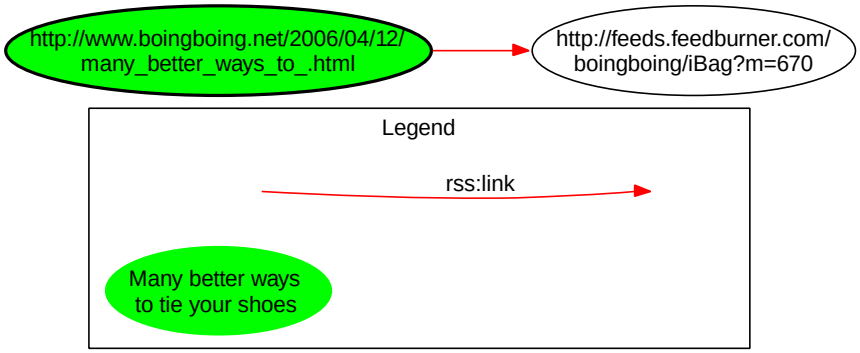

Figure 6. Statements represented with Graphingwiki, with the rss:title tag coloured.

While the Graphingwiki visualisations do not show all the different tags of the Wiki pages, any non-trivial Wiki has a sea of tags on its pages, most of which are irrelevant for a given visualisation. Thus, the exploration of visualisations by selecting the tags to be included may produce sufficiently limited, yet versatile views. Selecting the constituents of the visualised graph by the means of queries and inference might present an even more fine-grained method of exploration.

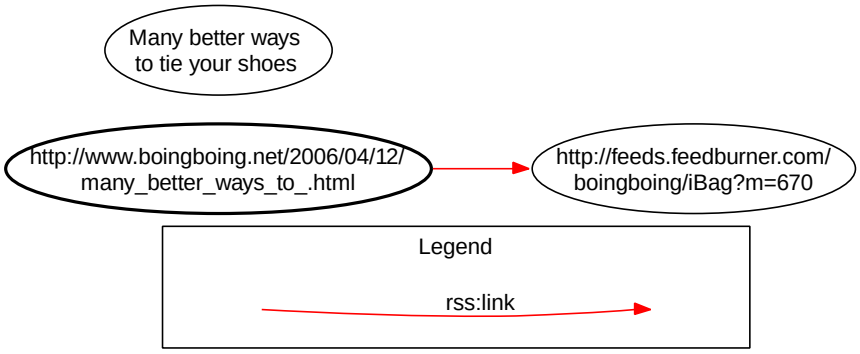

Figure 7. Statements represented with Graphingwiki, nodes ordered by the rss:title tag.

2.4. Applications

Traditionally, most knowledge management technologies have been grounded on knowledge base and ontology-centric approaches. These technologies have often resulted in monolithic applications that either strive for generality or are highly customised to a given purpose, e.g. storing accounting documents. In either case, the

knowledge content is derived by a group of experts in a top-down fashion. This requires a great amount of work for the extraction and taxonomisation of data and for the creation and upkeep of ontologies. [46]

Thus, traditional knowledge management tools are not valid for our approach. Graphingwiki aims to exploit easy collaboration so that data is gathered iteratively and its ontology emerges in a bottom-up fashion. Hypertext is a popular paradigm of content management whose human-centric approach has spawned many applications, the most successful of which has undoubtedly been the World Wide Web. A number of tools have been produced for knowledge management using basic hypertext approaches. mSpace[9] aims for organising and alleviating information searching, structuring, and analysis. Gzz[10] is a structure visualisation tool created by the Hyperstructure Group at the University of Jyväskylä. These approaches are based on previous work in the hypertext community on information representation: mSpace employs models called mSpaces, while Gzz uses Zzstructures created by the hypertext pioneer Ted Nelson [47].

Current knowledge management tools have increasingly embraced Semantic Web technologies, most notably RDF. Most of them are single-user oriented tools that enhance normal existing tools, such as desktop environments and editors. Examples of these include Gnowsis[11] and the Wiki-like SemperWiki[12]. Fenfire[13] is the follow-up project of Gzz that introduces RDF as its main data source. In addition to the Gzz-like structural views Fenfire includes mind map and Portable Document Format (PDF) document citation visualisations.

Existing semantic Wikis can roughly be divided into two categories, both of which have advantages and problems. Wikis of the first type treats semantic information as add-on data which has to be separately edited. This usually enables the Wiki to restrict the semantic data to specific schema and ontologies but renders it usable to experts only. Rhizome [14] is a typical representative of this type of semantic Wiki.

Semantic Wikis of the other type include semantic data as an integral part of the Wiki pages and have a markup style for it. These Wikis can further be divided into two groups with respect to their handling of ontologies. The first group includes the Wikis who can only import external ontologies and cannot form their own data models. Wikis of this group include Makna[15] and KendraBase[16].

Wikis of the latter subgroup have some abilities for forming a data model based on the semantic data in the Wiki itself. They include Semantic Mediawiki[17] and WikiSAR[18]. Semantic Mediawiki aims to make semantics available to the masses by taking a lightweight approach for expanding the types of content that can be included in a Wiki. WikiSAR is also a lightweight semantic Wiki, very similar in nature to

[9] http://www.mspace.fm/
[10] http://www.nongnu.org/gzz/
[11] http://www.gnowsis.org/
[12] http://semperwiki.org/
[13] http://fenfire.org/
[14] http://rx4rdf.liminalzone.org/Rhizome
[15] http://www.apps.ag-nbi.de/makna/wiki/Main
[16] http://www.kendra.org.uk/wiki/wiki.pl?KendraBase
[17] http://meta.wikimedia.org/wiki/Semantic MediaWiki
[18] http://wiki.navigable.info/

Makna but including rudimentary visualisation abilities for forming a general view on the Wiki.

As can be seen, there are a great amount of tools for the knowledge management of complex bodies of data. However, none of these tools are able to satisfy the following requirements identified for Graphingwiki:

R1. Supports the iterative collaboration of a large body of experts.

R2. Creates visualisations that can be interactively adapted to the needs of the user.

R3. Is widely available, enabling diffusion to a wide user base.

R4. Is easy to use and does not require any substantial training.

R5. Enables the creation of a data model based on the content.

R6. Has advanced semantic querying or rudimentary inferencing abilities.

Table 1 illustrates the features that the listed example applications implement, according to their documentation. In many cases the documentation did not present the features explicitly. In these cases a judgement on the availability of a feature was made based on overall analysis of the documentation. Whenever possible, the applications or their demo versions were also tried in practice.

Table 1. A comparison between application features and Graphingwiki requirements

Applications	R1	R2	R3	R4	R5	R6
mSpace	-	-	X	X	-	-
Gzz	-	X	-	-	X	-
Gnowsis	-	-	X	X	-	-
SemperWiki	-	-	X	X	-	X
Fenfire	X	X	-	-	X	-
Rhizome	X	-	X	-	-	X
Makna	X	-	X	X	-	X
KendraBase	X	-	-	X	-	X
Semantic Mediawiki	X	-	X	X	X	-
WikSAR	X	-	-	X	X	X

There are many Wikis that already fulfil the requirements R1, R3 and R4, which suggests that a Wiki extension would be the most natural implementation method for Graphingwiki. This work concentrates on implementing requirements R2, R5 and R6, in the same order of preference. The FRONTIER[19] project had already used a rudimentary form of visualisation in the wireless standard Wiki resource WiFiPedia[20], which shed light on the need for visualisations in the handling of large data sets. Graphingwiki is a more systematic approach for creating and exploring visualisations. Therefore, this work largely focuses on requirement R2, which was also the most neglected among the listed applications.

[19] http://www.ee.oulu.fi/research/ouspg/frontier/
[20] http://www.wifipedia.org/

2.5. Conclusions

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.

Charting the linkages of protocols is a difficult subject that requires efficient knowledge management techniques. Semantic Wikis are proposed as the state-of-the-art technique that enables the required collaborative knowledge gathering. As data on protocols and their implementations is abundant, visualisations are presented as the method for gathering a general view on the data. Interactive visualisations enable the examination of different aspects of the data in an easy, Wiki-like, and efficient manner.

3. THE PROTOS-MATINE METHOD FOR ASSESSING INFORMATION INFRASTRUCTURES

This chapter is divided into five sections. The first section introduces the concept of protocol dependency in the context of the examples and experiences detailed in the previous chapter. The second section asserts the need for a modelling methodology for discerning protocol dependency, and introduces the PROTOS-MATINE method to carry out the task. The views and the data required by the method are presented further in the section. The third section presents collaboration as an important component of the PROTOS-MATINE method and elaborates on the data gathering processes used. Finally, limitations and problems of the stated approach are presented in the fourth section, and summaries are drawn in the final section.

3.1. Protocol Dependency

> "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 disclosed with the test material. It has become apparent from the LDAP, SNMP, and H.225.0 cases that the impact assessment is not a trivial task due to the 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 [48]. 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.

The following list presents the classification of meta levels aided by the illustration of PROTOS test material relationships (see Figure 8).

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 disclosed 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.

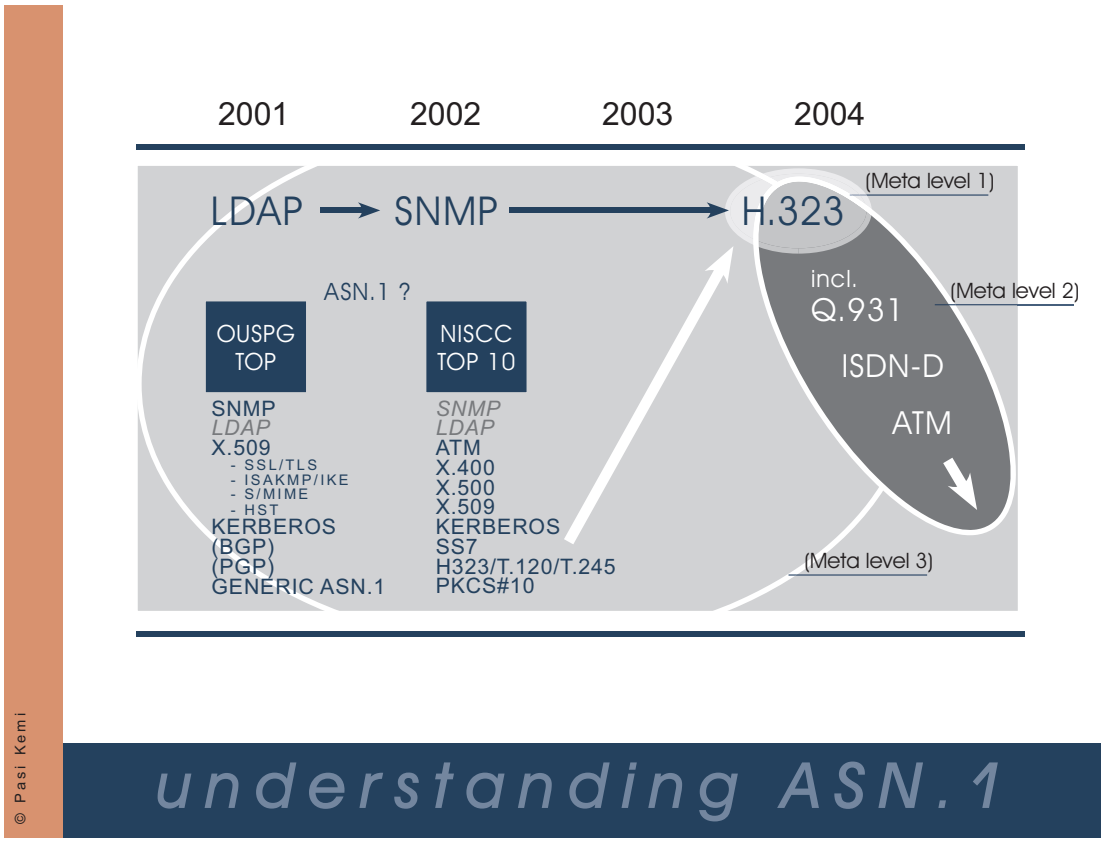

Figure 8. Vulnerability meta levels vs. PROTOS test materials.

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 are perceived as meta level two. This is illustrated H.225.0 case where sub-protocol Q.931 turned out to be a protocol shared 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, the 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 are called meta level three vulnerabilities. In the test-materials this is reflected by ASN.1 chain, a shared scheme with numerous protocol families, for example LDAP, SNMP, H.225.0 (subset of H.323), and others (meta level three).

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 dependency. 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

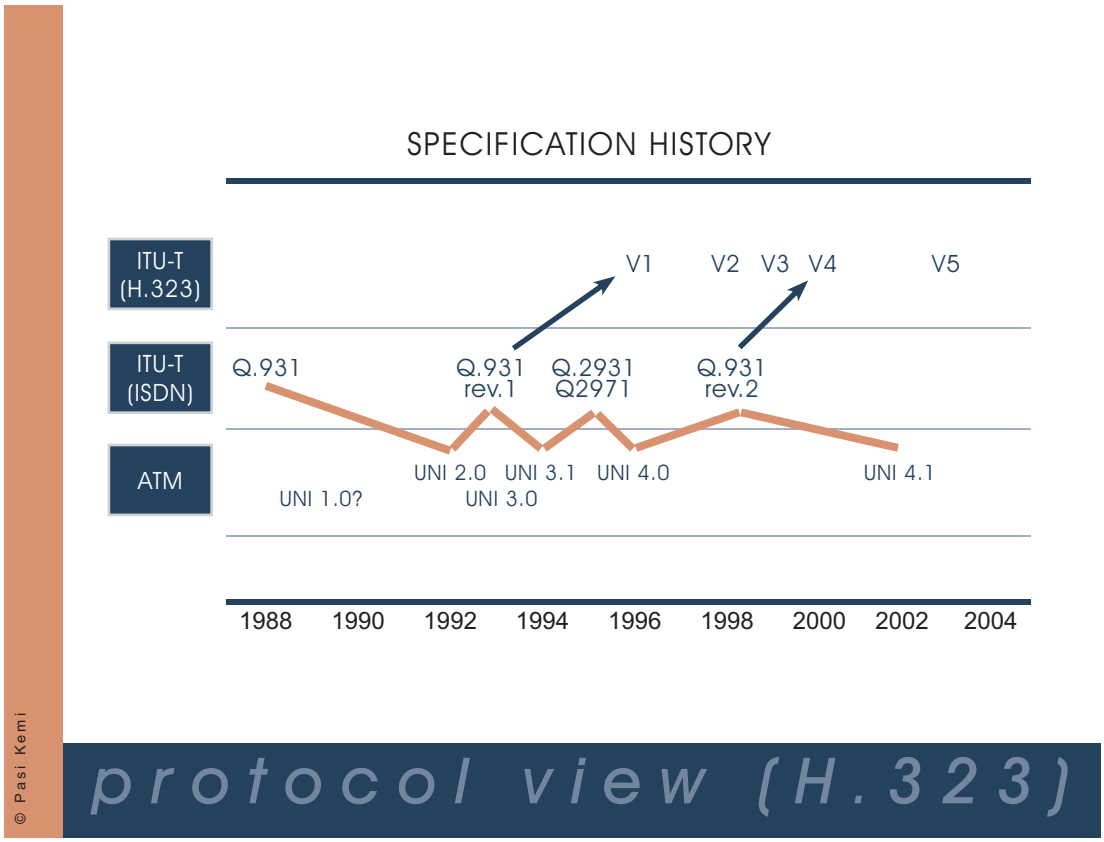

Figure 9. The specification history of Q.931 explicates its protocol dependencies.

study on the history of Q.931 specifications. The results, presented in Figure 9, 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. Testing could have included all affected implementations, focusing on the most important ones.

3.1.1. Impacts of Protocol Dependency

Figure 10 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 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.

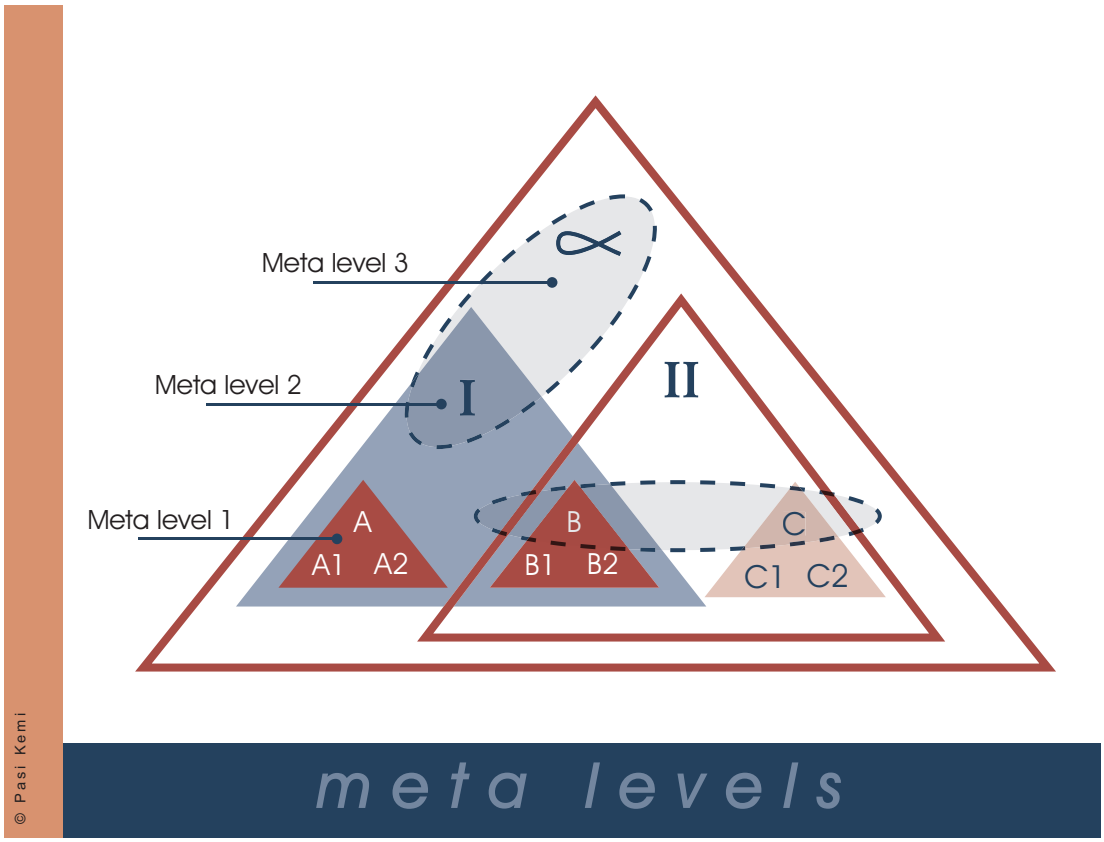

Figure 10. A scenario with different levels of dependencies.

3.1.2. 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.

Standard boards are also heavily occupied and otherwise influenced by actors from the industry that try to leverage the development of standards to their advantage. The standardisation process is often riddled with compromises among the various interest groups, sacrificing the technical integrity of the resulting standard for political and business interests.

Dependency through re-use also presents itself when identical code, methods or protocols appear in multiple implementations. This may occur due to a 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.

Component re-use often includes moving a component outside its original context, for example using components from a telephony network in the Internet. This may yield unexpected results as the assumptions made when creating the component may

not hold at all in the new context. Common examples of such a paradigm shift are the problems caused by using components from single-user systems in multi-user environments due to missing access control and input validation checks, among other issues.

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 Wireless Application Protocol (WAP) are is an exemplary case of the downsides of overlapping development [49].

3.1.3. 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 itself can 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 [50]. 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. 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 [50]. 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.

3.2. PROTOS-MATINE Method

As discussed in the previous chapter, the risk management of a system is difficult due to the changing nature of information technology. A protocol, however, is more of 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 [4] [13]. 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.

PROTOS-MATINE methods for researching protocol dependency include:

1. Expert interviews augmenting the plain data mining approach
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

The accumulated data is presented with views that bring up different aspects from the data related to protocol dependency and security (see Figure 11). As the results of the method are highly visual, they can additionally be used as a communication method between researchers and other actors.

3.2.1. Views

The PROTOS-MATINE method illustrates protocol dependencies from multiple angles. These different views aid in perceiving the inheritances and hidden links between protocols. The method puts forth the structures, dependencies and vulnerabilities present in the gathered protocol-related data. Multiple visual views are used to clarify different kinds of protocol data for various usage scenarios.

The three views mainly used in the PROTOS-MATINE method are the protocol view, the technological usage view and the organisational view. The main views can

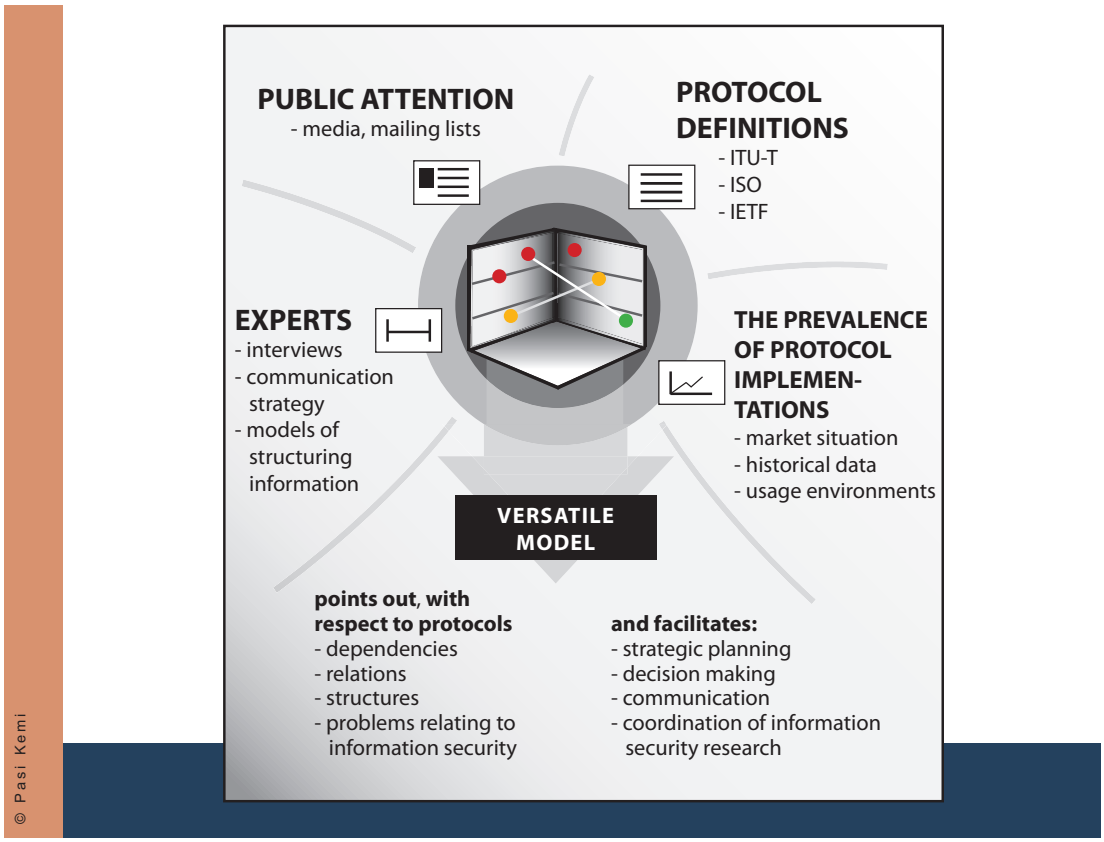

Figure 11. Data sources and research results of the PROTOS-MATINE method.

be modified according to a specific target group or usage scenario. As an example, a protocol view can be focused on the most essential dependencies for an organisation by including data on all the protocols used by software that is critical for the operations of the organisation. When used together, the three views present a general view that cannot be discovered without the richness of sources and viewpoints.

Protocol View

The protocol view describes the history of a protocol specification along with its connections to different versions of other protocols. This view is useful in outlining the scope of utilisation of the examined standard by other protocols. It describes the whole lifespan of the protocol from the standardisation organisation point of view.

A survey of the Multipurpose Internet Mail Extensions (MIME) protocol presents an example case of the protocol view. MIME was originally designed to be an email exchange standard that enables the delivery of multipart email messages. The parts of a MIME message body can include different data types represented in different formats. Currently MIME is used in a much wider scope, ranging from applications such as mobile messaging to the so-called enterprise web services. As there are two distinct ways in which applications use MIME, two protocol views are presented.

The first protocol view (Figure 12) includes the protocols that use MIME for data representation. It shows that, for example, the MMS protocol used in mobile telecommunications and the ISUP protocol present in the digital phone systems employ MIME

formatted messages. Use cases for the view include determining test subjects for a MIME test suite and discovering the scope of protocols including MIME that might be at risk to a MIME vulnerability.

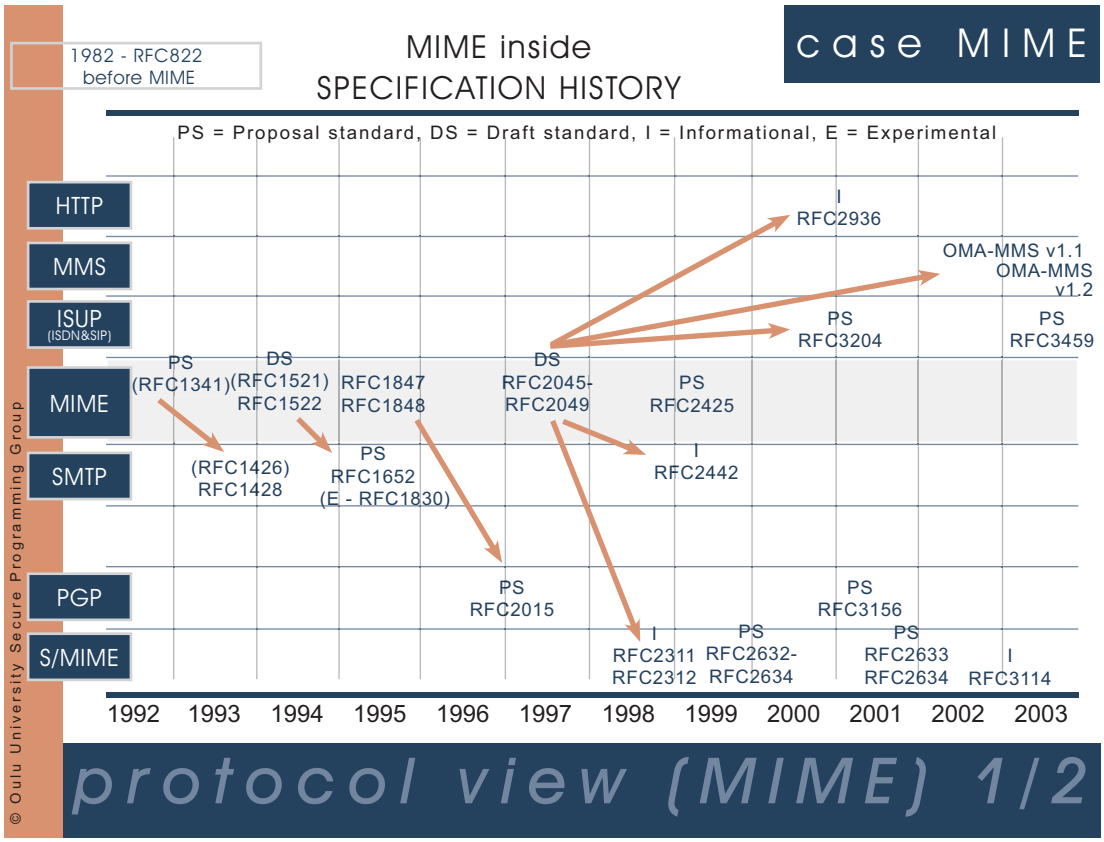

Figure 12. A view on the protocols that employ MIME.

The second protocol view illustrates the data formats delivered in MIME format, which sheds light on the software and hardware that employ MIME. Figure 13 clarifies the services and usage scenarios that can be susceptible to MIME faults. For example, MIME messages are used to describe fax messages and LDAP schema, which may render fax machines and directory servers vulnerable to a MIME vulnerability.

Technological Usage View

The technological usage view describes the usage environment of the examined protocol. It is used to chart the equipment that employ the protocol or handle protocol data.

The technological usage view of the MIME protocol presented in Figure 14 includes the appliances that produce, convey or accept MIME messages, along with inter-appliance linkages. The blue boxes represent the end user software that handles MIME messages, such as WWW-browsers, news readers and mail software. They are connected to servers that handle news and email messages or WWW pages with the help of the Hypertext Transfer Protocol (HTTP) protocol. HTTP or HTTP-like protocols relaying MIME messages are also used by many other protocols such as the

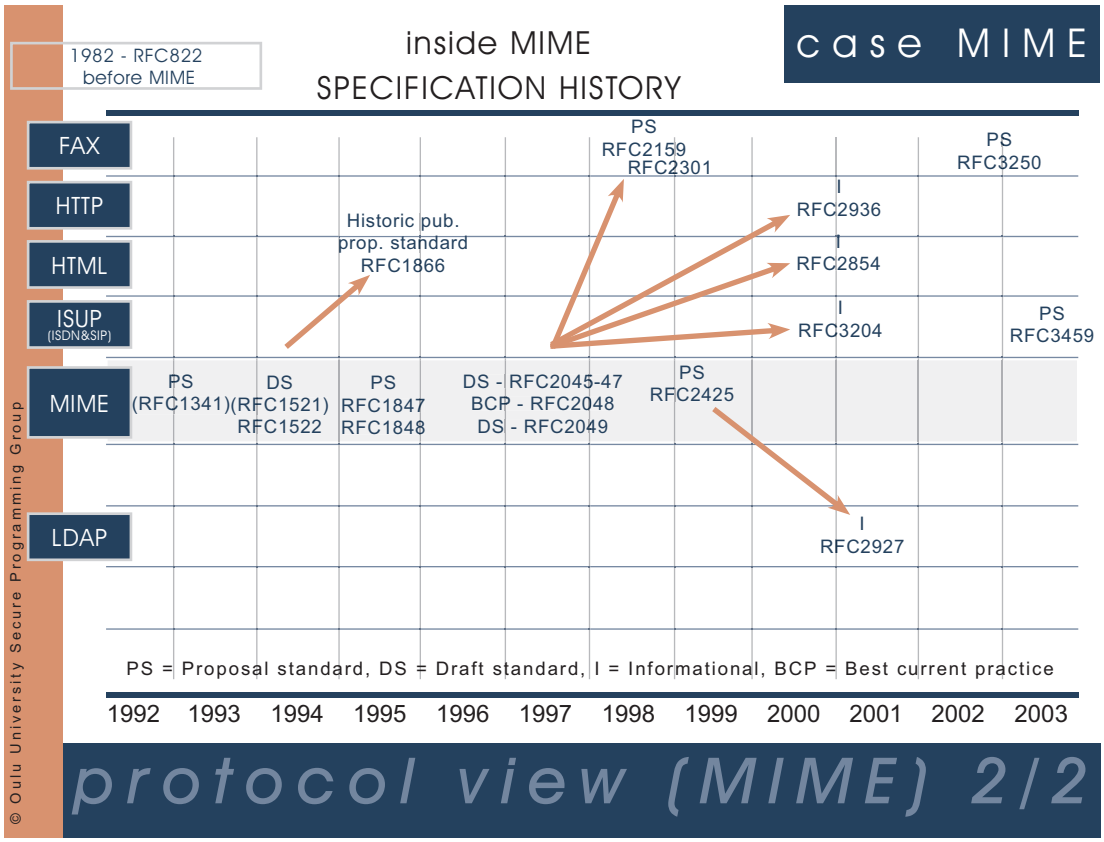

Figure 13. Data formats delivered in MIME format.

Session Initiation Protocol (SIP) used in Internet telephony. Also, the now-abundant firewall, virus scanner, and content filtering software have to process MIME messages.

The technological usage view illustrates the prevalence and usage environments of protocol implementations. If need be, the view can be expanded to organisation-specific product and vendor listings.

Organisational View

The organisational view depicts the sectors of an organisation where the examined protocol is used. The view lends itself well to organisations of different size from SME:s to infrastructures. For example, the organisational view of a corporate organisation shows which offices and branches use the examined protocol. This view displays the scope and criticality that a vulnerability in the examined protocol would have for the organisation along with the affected actors.

The organisational view in Figure 15 is an example mapping of the usage of H.323 protocol in the Finnish critical infrastructure. Data gathered from interviews and media follow-up is visualised in the view to clarify the usage and prevalence of the H.323 protocol suite in the critical infrastructure. This represents the importance of the protocol to the society, and the functions that could be rendered unavailable by a protocol vulnerability.

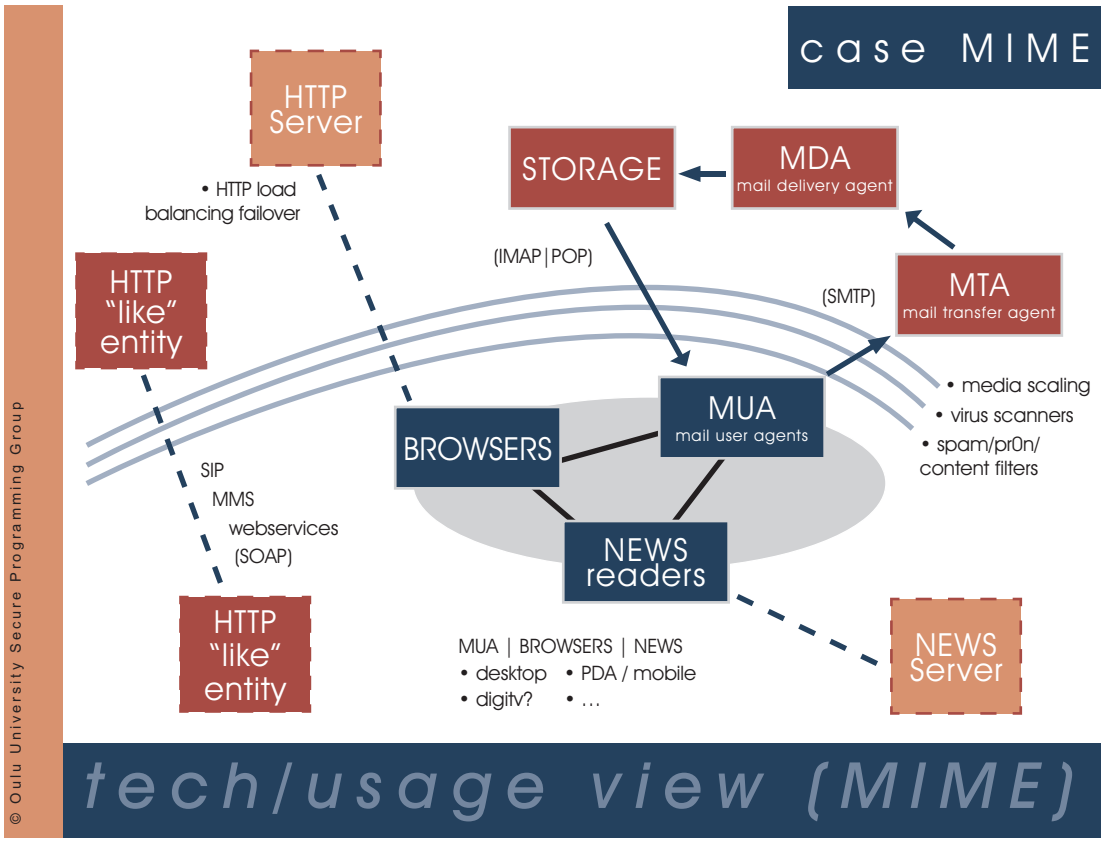

Figure 14. The technical usage view of the MIME protocol.

3.2.2. Data Sources

The major sources of data in the analysis of a protocol are the protocol specifications created by standardisation organisations, related literature, expert interviews, and media follow-up.

The analysis is started with a scan of the specifications and literature to get an initial outlook necessary for preparing an interview question framework. The forming of an expert contact network is started in the early stages of the analysis. The network can be tapped for interviews as soon as the question framework is complete. The contact network is supplemented with new contacts from previous interviews throughout the entire analysis. Media is also followed all along the analysis.

Visualisations are formed as soon as some data is available and constantly iterated upon as soon as further information is received. At interviews, the visualisations are presented to experts, whose comments represent an extremely valuable input for the development of more effective views. A central protocol data repository is introduced with data from analysis of standards and literature. The material from the interviews is transcripted and analysed for the repository, which is also updated with data from the media.

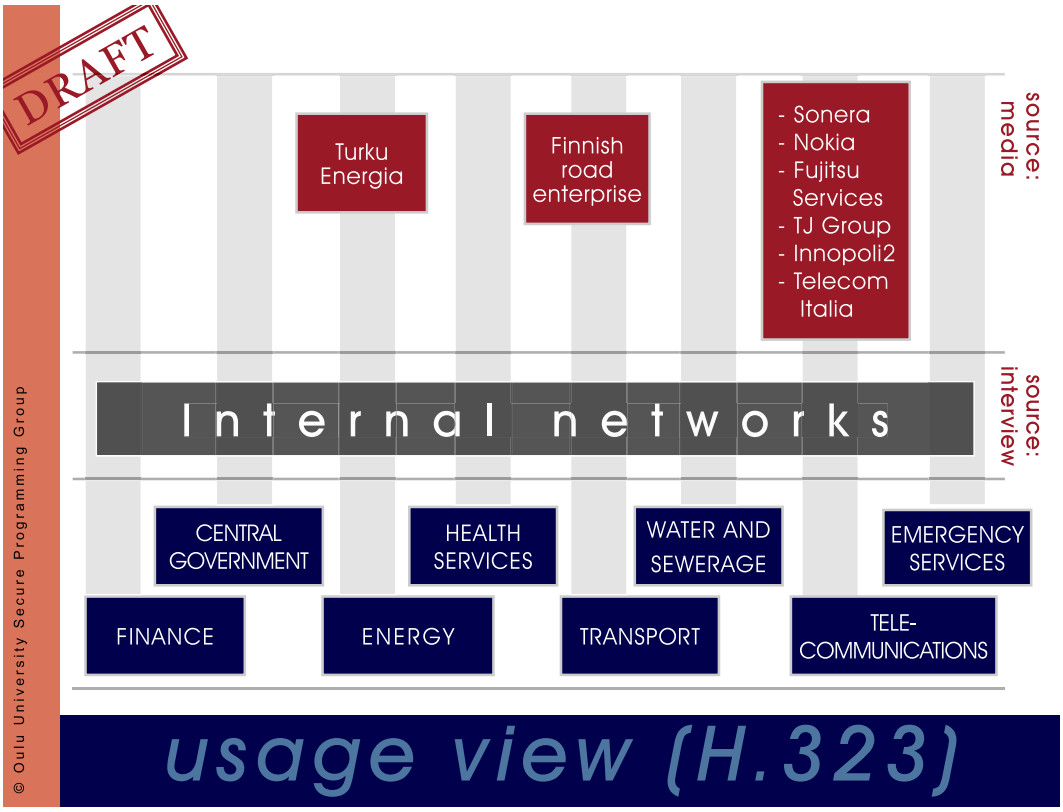

Figure 15. An example of an organisational view of the H.323 family of protocols from the viewpoint of the Finnish critical infrastructure.

Open Source Intelligence

The PROTOS-MATINE method aims for the analysis of critical infrastructure and other distributed, diverse and often privately owned or classified systems. Thus it is easy to argue that publically available information is not sufficient for such analysis. However, experience from the intelligence community suggests otherwise [51].

Historically, states have used intelligence to monitor the technical advances of adversaries to save time and money on domestic projects. On most of the cases, open sources have amounted to as much as 80 percent of the used intelligence. Highly classified and isolated cases are seen as the main limitations of the method. In most cases, however, a wealth of information on any technical subject is freely available. Some of the data could also be considered classified by its nature, as this distinction can be difficult to make. Thus open acquisition of information is seen preferable to other methods. [52] [53]

In recent years the development of communications and information technology have vastly expanded the exploitation opportunities of open source information. Currently it is estimated that even 90 percent of intelligence stems from public sources while six percent of the remaining represents grey intelligence, i.e. non-classified materials of limited distribution. The analysis based on open source information can be based on a ample body of data, which amplifies its trustworthiness although some details may be lacking. The direction of the intelligence towards economy and technology advocates its efficiency. Thus, it is proposed that although the PROTOS-MATINE

method is best used from within the examined organisation, it can also be applied from the outside. [51]

Protocol Specifications

Going through the technical documents describing a given protocol takes a great amount of time. Searching for specifications related to a protocol from the documentation of a standardisation organisation may prove challenging. Not all the documents are electronically available, and some of them may carry price tags. An example of a good source of specifications is IEFT, whose Request For Comments (RFC) documents are easily probed with help of good indexes and efficient search functions.

Other noteworthy standarisation organisations, along with exemplary related standards, are included in Table 2.

Table 2. Standardisation organisations and some of their standards

Organisation	Standard
ISO	OSI standards (X series)
ATM Forum	ATM related standards (B-ICI, LANE)
ITU-T	Telecommuncation standards (H, G and T series)
ETSI	Telecommunication standards (GSM, UMTS, 3GPP)
IEEE	Electrical interfaces (RS-232C, 802 series)
ANSI	Various standards (FDDI, Z39.50)

In addition to the specifications made by standardisation organisations, protocol related data is abundant in literature and the Internet. The following list includes some sources that have been found to be helpful in the analysis of various protocols.

- Protocols.com [1],
- Network communication protocols map [2] and
- TCP/IP Illustrated Volume 1, The Protocols [54].

Expert Interviews

Expert interviews are an instrumental source in the formation of the technological usage views of a protocol. Interviews deliver the most genuine view on the protocol related technologies, for the specifications do not necessarily give a realistic outlook on the utilisation of the protocol. In reality, the specifications may not be implemented at all or the implementations may be missing critical portions of the specifications. It may also be the case that even if implementations do exist they are not used widely, or at all. Often standard bodies have defined several protocols for very similar uses, and their degrees of implementation and usage may vary greatly. On the other hand, many implementations use protocols that are proprietary and/or not formally defined at all, which makes their analysis challenging.

[1] http://www.protocols.com/
[2] http://www.javvin.com/map.html

Experts in the organisation making the analysis are in a key role for the first interviews. Getting further interviews may prove to be difficult as the experts may lack time and incentive for giving interviews. Existing organisational contacts should be maximally utilised, partners in co-operation should be first to be queried for interviewees. Other good sources of experts are corporate clients, hardware and software vendors, data administration and network suppliers and administration. Snowball sampling methods are effective in gathering more interviewees from an existing basis [55]. It is used by querying old contacts for new ones and asking interviewees to name other experts in their field. Suitable interviewees can also be found during the course of a media follow-up.

The interview material is sufficiently large when the interviews mainly produce recurring data, i.e. when the material saturates. Interview material concerning standardised technical matters saturates in much earlier stages than that of opinion interviews [55]. Research in the PROTOS-MATINE project indicates that a minimum of eight interviews is required to form a trustworthy view on a technology. This figure is highly dependent on the scope of the examined technologies and the quality of the interviews.

The interviews are carried out face to face or by telephone, and they are recorded for further examination. Email interviews can be used to lessen the required transcription effort. However, difficulties may ensue in getting interviews by email as the request emails are easier to ignore and can be lost amongst the abundance of other correspondence. Personal contacts are by far the most effective method for getting interviews.

The interviews use a basic question framework that is augmented with specific questions concerning the examined protocol. Questions regarding confidential data such as the specifics of products or services should be avoided. The first interviews on a protocol are a great help for further focusing of the question framework. The framework is divided into four parts representing the viewpoints of usage, technology, security and vulnerability disclosure. At the end of the interview, the interviewees are asked to name other experts that could contribute to the method. The following describes key topics of the interviews, divided in the aforementioned viewpoints.

The usage view

Questions related to the usage view aim to gather protocol related usage scenarios from real network environments. The resulting data is instrumental in the formation of the technological usage view.

- Where, how and for what purpose is the protocol used?
- Which operations are the protocol and its implementations used to perform?
- Is the protocol used to replace an existing technology?
- Have new uses emerged for the protocol or its implementations?
- What will the status of the protocol be in the future?
- Which services employing the protocol do you offer?

The technological view

Questions related to the technological view aim to discover the context of the protocol in the spectrum of technologies. They complement the technical usage

view and add footnotes to the usage scenarios. Questions related to the implementations of the protocol lay a foundation for forming the organisational view.

- Who are the most important technology providers related to the protocol?
- Does any other equipment become involved with network traffic related to the protocol? What equipment?
- Does any equipment you are using employ the protocol?
- What plans do you have for the future use of the protocol?
- Do other protocols handle data related to the protocol, or the data that the protocol is used to carry?

Security
: Questions regarding the security issues of the protocol are used to map the criticality of the protocol for the organisation. The resulting data is used in the formation of the organisational view.

- How important is the protocol to your organisation? Which organisational functions are fulfilled by implementations of the protocol? How critical would a vulnerability in those implementations be for your organisation?
- How is the security of the hardware or software employing the protocol evaluated?
- Who would you consider responsible for vulnerabilities in the protocol or its implementations?

Vulnerability disclosure
: Questions about vulnerability disclosure are used to gather practical experiences on vulnerabilities in the implementations of the examined protocol. The resulting data presents the organisational view with practical viewpoints on the importance of the protocol.

- Have you heard about vulnerabilities in the protocol or its implementations? Through which channels have you received this information?
- How have these vulnerabilities affected your organisation?

Snowball sampling
: Snowball sampling is used to collect a list of experts that can be put to use to get further interviews.

- Can you name other experts on the protocol or related technologies?

Media Follow-Up

While following the media is laborious, it rarely presents data that is important for forming any of the views. However, continuous media follow-up is useful for forming a general view on protocols and related technologies. The media follow-up in an

extended and expansive protocol survey should be limited to gathering relevant details that are discovered by following the media in normal course of work.

In a short-term survey, a representative sample from newspapers and technical journals is selected as the media source of the follow-up. The scope of the follow-up depends on the period of publications of the sources and on the resources available for the follow-up. It may be useful to have publications that reprint their articles on the web used for the follow-up, as they somewhat relieve the otherwise arduous data gathering.

Newspapers do not usually handle technology as such, concentrating instead on the effects of technology on the society and on the day to day life of the common man. This is why they represent a valuable counterbalance to the technical journals that do not usually study protocol from the point of view of infrastructures.

The media sources are searched for data on the examined protocol, its implementations or related technologies. This data includes the prevalence and usage scenarios of the protocol, vendors of the implementations along with their market shares and contracts, users of related technologies, and vulnerabilities in the implementations. Experts that have been interviewed in the media on the protocol represent a source of interviewees and shed further light on the users of the protocol.

3.3. Collaborative Data Gathering with the PROTOS-MATINE Method

This section explains the views that can be generated using the MATINE method. First, the general processes for the creation of the different views are presented. Next, procedural steps for using Graphingwiki to follow these processes are fleshed out. Finally, some rationale is presented for the usage of the Graphingwiki features of visualisation and inference in the PROTOS-MATINE method, and examples of use cases are presented.

3.3.1. Protocol View

The protocol view includes the development history of the protocol along with the organisations that have participated in its development. It is initially formed on the basis of protocol definitions of the standardisation organisations. The organisations involved in the development and standardisation of a protocol are first tracked with the help of literature and databases on standards. After this the relevant protocols are queried with the search functions provided by standardisation organisations. These are initially examined, especially with respect to their references. Most protocols depend upon other hosts of specifications for complete functionality, and these dependencies are essential data for the protocol view. Protocol specifications that include the examined protocol as a functional part also represent an important data source for the view. During the standardisation history, a protocol may have been split into two distinct protocols, or two different protocols may have been merged into one.

Also the different statuses of the protocols should be marked up. As an example, Internet Engineering Task Force (IETF) specifications come in all shapes and sizes: Internet Drafts, Standards; Proposed Standard, Draft Standard, Internet Standard, Non-

Standards; Experimental, Informational, Historic. Usually, the statuses of the standards shed light on the importance of the standards. Some of the specifications might only include experimental implementations or guidance on the use of protocol. This is why they should be symbolised in different ways in the view.

The expert interviews contribute to the protocol view with a more realistic outlook on the standards. The main input from the experts is data on the actual implementation status of the standards: which parts of the standards have been implemented, which parts have not been implemented, and what kinds of extraneous functionality is included.

Figure 9 presented earlier represents a protocol view of the Q.931 standard of the H.323 protocol family. It illustrates the organisations that have been involved in the development of the Q.931 standard, and the different versions of it that have been used as components in other protocol families. This indicates clearly that protocol specifications are under constant development. Different standardisation organisations create new protocols and protocol families based on existing components. This reuse adds to the dependencies of the components involved. In the case of the Q.931 protocol, the dependencies have arisen in the standardisation process between the protocols of ITU-T and ATM Forum.

3.3.2. Technological Usage View

The technological usage is formed on the basis of expert interviews, literature, media follow-up and protocol specifications. The role of expert interviews is focal as they shed light on the practical details of protocol specifications and implementations. Convergence has made it increasingly difficult to map different network environments on the base of specifications alone. Thus, discerning complex networks requires the involvement of experts of various environments, so that the resulting view gets closer to reality. Furthermore, hardware and software external to the specifications may need to handle data of the examined protocol, even if the specifications do not state this or states conversely. A vulnerability in the examined protocol may involve these hardware and software, which should therefore be tested for similar vulnerabilities.

Specifications and literature sometimes include conflicting data. Expert interviews help to verify the actual situation in these cases. Although interviewing technical experts usually results in enough data on real protocol implementations and network environments, some situations may require inspection of live networks in order to get a sufficient view. Depending on the needs of the organisation performing the analysis, the technological usage view can also include the vendors and suppliers of used equipment. They are found with the help of interviews and media follow-up.

3.3.3. Organisational View

Forming the organisational view on e.g. the critical infrastructure requires interviews from all the sectors of the critical infrastructure, which requires a lot of work. The PROTOS-MATINE method divides the view into the following sectors:

- economy,
- energy,
- communications,
- traffic and transportation,
- central government,
- maintenance and administration of information technology,
- crisis management,
- mass communication and
- health care.

Other sector classifications of the critical infrastructure can be found from various sources, for example the publications of the Finnish National Emergency Supply Center [1] and the [22] US President's Information Technology Advisory Committee. Other infrastructures, such as those of corporate organisations, can be classified along branches or places of business, as appropriate.

The used implementations along with their vendors are indicated for each of the sectors in the organisational view. The view presents the sectors of the infrastructure or the organisations that employ the examined protocol. Most organisations are fully networked and contain the basic Internet protocols throughout all sectors, which renders the organisational view ineffective for mapping these protocols. On the other hand, the forming of the organisational view is recommended for protocols whose functions are more specialised. The view presents the scope of the protocol and related technologies in the examined organisation, and identifies the functions that would be affected by a vulnerability or other failure of operation in the protocol or its implementations. This sheds light on the criticality of the protocol to the operations of the organisation.

Forming the organisational view requires a careful inspection of a significant body of data from reliable sources. A realistic view requires following different organisational IT security policies and practices in addition to highly focused media follow-up and expert interviews. Thus, it is exceedingly important that the visualisations are created in an iterative manner and continually focused with input from the interviews and other new data.

3.3.4. Extraction and Augmentation of Data

Due to the wide area of protocol dependency, the use of Graphingwiki in the scope of this thesis is focused on forming the protocol view for the PROTOS-MATINE method. This work consists mainly of the phases of automated data extraction, augmentation and collaboration.

Initially, protocol data is gathered from standardisation organisations and from indices collecting data on standards. Examples of semantic data in standards include status, types of relations with other standards, the protocols involved and so forth. The

data is gathered with scripted methods and inserted into corresponding Wiki pages with similar means. Most of the structured data in the standard texts is imported, following the approach of aggressive population of semantic and ontological data from existing databases [56]. This results in the quick generation of a relatively rich body of data as a starting point for a comprehensive protocol Wiki. Also other semistructured data on standards can be inserted.

While the process of adding given semistructured data cannot be effectively automated for all cases, the extraction approach is a pragmatic one, making the best use of the data available. Although the different data sources may adhere to any number of conflicting explicit or implicit ontologies, a lightweight approach to ontology gives the leverage to process the resulting primordial soup. This represents a bootstrapping process for semantic Wikis, as the benefits of semantic data are illustrated only by the availability of such data. These benefits far outweigh the costs of generating the semantic data along with the data. Similar approaches to data extraction have been applied successfully [57] [58].

After the initial data gathering phase, the data gathered from the different sources of the PROTOS-MATINE method is inserted into Graphingwiki. The details of this process are somewhat subject-dependent, but follow the same basic principles. Whenever new concepts are introduced in the data, new Wiki pages are created to describe them, and data concerning a protocol or other concept already in the Wiki is simply updated to that page.

As much of this data as possible is inserted to the pages in the forms of the attributes of the concept and its relations to other concepts, as these forms of data are machine-processable. Page templates can be used to help formalise the extended markup [18]. On the other hand, custom semantic tags for specific situations or scenario can be used. Explanations, quotes, and WWW resources can be written on the page as is.

In the collaboration phase, the experts are invited to join in to view and augment the results gathered in the Wiki from their interviews and additional sources. Experience has indicated that it may help in this phase if the data gathering phase has not been exceedingly careful in filtering contradictory or controversial arguments about the protocols. This is due to the fact that experts are often more keen to remove such flaws from existing data than to add complementary data to an empty page.

During these phases the data body is developed from a fairly generic and dry viewpoint towards exceedingly rich and specific use cases. Users immediately benefit from the practical domain experience included in the Wiki.

3.3.5. Visualisation and Reasoning

The ability to make logic deductions on the expert-supplied data can unearth results not easily discovered by traditional means. As an example from the Wiki context, Decker et al. uses reasoning to enable reuse of software engineering knowledge [59]. The approach taken in the development of Graphingwiki with respect to reasoning techniques is straightforward and pragmatic, so that the inclusion of logic is based on approaches that are known to work and are required. The focus lies heavily on immediate benefits of reasoning, the inclusion of higher-order structures is deferred until they are explicitly needed [56].

As an example case of inference on the domain of protocol dependency, the true cause of a network error related to two hosts containing a plenitude of services can be inferred from a data body on protocols and related implementations. Similarly, the gross effect of a single vulnerability for a network can be assessed, optionally involving even chains of vulnerabilities and exploits. Similar approaches have emerged in the context of security research, particularly in network vulnerability assessment (e.g. [60]), but also in inspecting the configurations of single workstations (e.g. [61]).

3.4. Limitations

The main limitation with the PROTOS-MATINE method is the laboriousness of the data gathering needed for the generation of views. This stems from the massive amount of data and diverse experience required for the analysis of any non-trivial technology. The accelerating scope and convergence of technologies increases the need for diverse experience in the analysis.

A focal source of data in the method are interviews, which can be problematic in several ways. Getting interviews may prove a major challenge as most of the experts on any subject have very busy schedules. The interviewees gathered with various intractable methods may eventually invoke the non-disclosure of company secrets excuse and refuse to provide any useful information on any subject at the interview. Transcribing and abstracting the interview material amounts to another phase of hard labour.

The population of a Wiki with data from semistructured sources is a useful facility, but it may not be applicable to a portion of available material due to technical or licensing issues. In some cases, the data abstraction features may suffer from some constraints. Visualisation techniques are naturally limited to a certain volume of data that they can relay in an efficient manner.

Reasoning also has its limitations that have hindered its use in many cases. Main problem is the state space explosion resulting from massive knowledge bases. This can be countered by using monotonic logic and highly domain-specific data sets, although limits on query tree depth and traversal time can also be of help. All the statements made with Graphingwiki are essentially monotonic, as they only bring more data to the knowledge base without contradicting earlier statements. This is due to the inherent lack of meaning of the statements in the Wiki, as the different aspects and relations are only given meaning by humans interpreting them, or by the inference rules and queries.

While the statements are limited in their effect, there are no similar restrictions to the inference rules queries. Thus, great care must be taken when generating them, as they might bring contradiction or belief revision into the system. The heterogeneity of the data gathered from various sources can present limitations to reasoning. As there are no guarantees on given semantic data being present on all concerned pages, the inference rules may not match all relevant data [58].

3.5. Summary

Technologies exhibiting protocol dependency may be at a risk from vulnerabilities whose scopes and impacts are amplified by the chains of dependency. The PROTOSMATINE method can be used to discern protocol dependency in the context of risk analysis. The method can be used for multiple functions which require different views for their operation. The current views in the method are the protocol view that shows the development of the protocol, the technological usage view that depicts its technical context and the organisational view that visualises its social scope. The data required for the formation of these views is gathered primarily by the means of reviewing technical documents, following the media, and making interviews.

In the context of the method, this work focuses of forming the protocol view with the help of the Graphingwiki. It facilitates collaboration and provides for and the automated extraction of data from existing sources. The visualisation and inference features of Graphingiki can be used to create the protocol views needed in the PROTOSMATINE method from the data derived by it. The open-ended approach to knowledge combined with collaboration and visualisation may prove a mixture that asymptotically yields clarity in the confusion surrounding the protocols.

4. PROTOTYPING

This chapter will describe the construction process of Graphingwiki. First the used model of development is described. Then the development framework is introduced and some of the requirements are specified in a more detailed fashion. Finally, the cycles of development are elaborated upon.

The construction process of Graphingwiki began with laying out a set of general objectives for the visualisation of collaboratively editable data. The objectives included an initial form for the visualisations based on hand-made experiments of the PROTOS-MATINE project, but there was no real certainty of the efficiency or feasibility of automating such visualisations. Conventional wisdom on software engineering dictates that this situation may be best addressed by the prototyping paradigm [62]. Prototyping consists of phases for gathering requirements, creating a quick implementation based on them, and user evaluation of the prototype, from which the requirements for the next prototype are derived [63]. Graphingwiki was created for the explicit need of creating the protocol views of the PROTOS-MATINE method. While the exact requirements for the view were uncertain, it was deemed that the project group would still be able to steer the implementation effort by using prototypes that provide preliminary views. Thus, it was decided that the work on Graphingwiki should commence by means of evolutionary prototyping.

The first prototype was decided to be a throw-away system for trying out the visualisation form and methods of user interaction upon it [64]. Experiences from this system would then form the requirements for the Wiki extension. The development of the extension would then continue in an evolutionary manner. Basic Wiki integration was to be performed first, with user interaction features following suit.

Users of the extension test out the requirements in practice, adjusting them to better match the real needs for the software and creating further requirements. Based on the experiences with the Wiki prototype, the final apparatus is created. Besides concentrating to fulfill the requirements, this phase aimed to demonstrate the usefulness of the data gathering methods stated in the previous chapter in the Wiki environment and to provide effective visualisations on the given data.

4.1. Graphingwiki Extension

The main methods used in Graphingwiki include additions to the MoinMoin[1] Wiki markup and plugin tools that save the semantic data for later processing, visualise the semantic data and make logical reasoning based on it.

4.1.1. Wiki Selection Criteria

Naturally the most important selection criteria for the Wiki to base Graphingwiki on was easy extendability. Otherwise the work would be moot or greatly increased in

[1] http://www.moinmoin.wikiwikiweb.de

difficulty. As this work was carried out for the OUSPG, the needs of the group also played an instrumental role in the selection of the Wiki.

Wikimatrix[2] is a WWW resource designed to help in Wiki selection. It includes feature comparison tables and a query wizard that lists Wikis adhering to given needs. The Wiki requirements for Graphingwiki are listed in Table 3 along with their sources and rationale. The original functional requirements of Graphingwiki are marked in the source column with their requirement codes as per Table 1.

Table 3. Wiki requirements for the Graphingwiki extension

Feature	Source	Reason
Extendability	OUSPG	Easy development
Open source license	R3, OUSPG	Academic use, increased availability
Stand-alone installation	OUSPG	Security, ease of administration
File storage	OUSPG	Easy access to Wiki data
Page history	OUSPG	Change tracking
Written in Python[3]	OUSPG	Language preferred by group

The MoinMoin Wiki was discovered to be the only one fulfilling all the stated requirements, which made the Wiki selection straightforward.

4.1.2. Architectural Design

The high-level architecture of MoinMoin is presented in Figure 16. The main framework is divided into the main engine and its plugin extensions that can be unique to each Wiki instance.

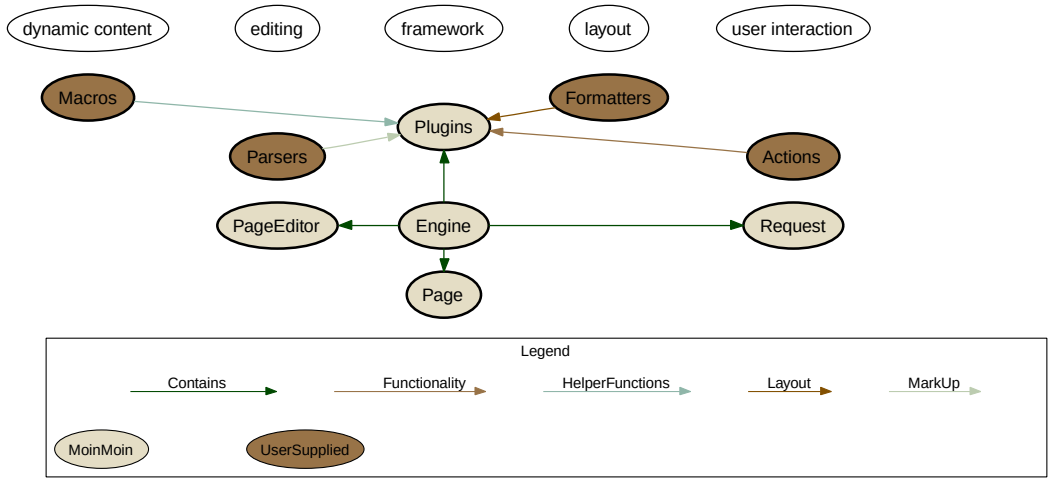

Figure 16. Overview on the MoinMoin architecture.

[2]http://www.wikimatrix.org/
[3]http://www.python.org/

The main portions of the MoinMoin engine include the *request* class, which is the main user interface of the engine. The *request* class gets HTTP headers and user input from the user agent, which in most cases is a web browser. It also performs the functions necessary to show the Wiki page or to perform other user actions, as necessary.

The *page* class is a framework class that contains methods for getting page text, its current revision, status, path, access control data, and other details. The *page editor* class is responsible for sending the editor form to the user agent, getting responses and saving data to page files if possible.

The basic MoinMoin installation contains many plugins for standard tasks such as rendering page file contents in HTML and performing text searches on the entire Wiki. Additional plugins may be installed in individual Wikis by their administrators.

Action plugins provide the system with extra functionality, such as spell-checking page content or showing a local site map with respect to the page. The actions are always related to the page viewed at the time of their invocation.

Macro plugins provide small helper functions that create dynamic content embedded on a page. The content can be presented from various sources such as user input, database queries, tables of content, or random text inserted from another wiki page.

Formatter plugins present layouts of the page in various forms such as HTML, XML, DocBook[4], or Python code. *Parsers* on the other hand can be used to input portions of the page in data formats other than the basic Wiki markup, such as Comma-Separated Values (CSV) data, Internet Relay Chat (IRC) logs, and the source code of many programming languages.

The MoinMoin plugin system can be mostly thought of as data flow architecture, where user input and page content determine the plugins that are called, along with their order. Figure 17 presents the flow of data in a case where the user desires to see a Wiki page in XML format. In this case the XML output is implemented as an *action* plugin, which the *request* calls as per user input. The *action* goes forth to instantiate an XML *formatter*, with which it calls the method of the *page* class for displaying its content. *Page* then calls the Wiki markup *parser*, which starts to call *formatter* on page contents. Whenever the *parser* encounters macros on the pages, it calls the corresponding *macro* functions that format their output with the *formatter* given by the *parser*. Respectively when the *parser* encounters a page portion in another markup, it calls the *parser* for this markup, which also formats its output accordingly. As the page data is formatted, it is sent back to the *request*, which sends it back to the user agent. The observant reader may note that the colouring in Figure 17 corresponds to the ordering of Figure 16.

Graphingwiki is implemented as a set of plugin *actions* to manipulate the page data, *macros*, and *formatters* to render the semantic data to the desired viewable or processable forms. The design strives to maximise backwards-compatibility and the use of existing MoinMoin features.

[4]http://www.oasis-open.org/docbook/

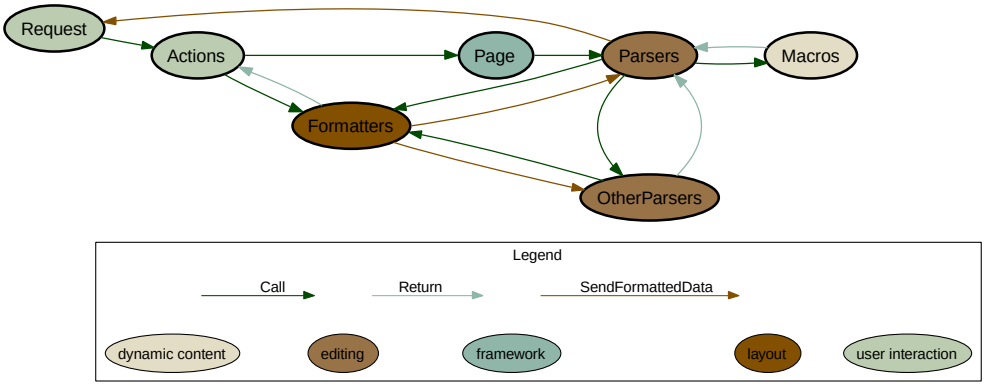

Figure 17. Data flow of a MoinMoin action.

4.1.3. Wiki Markup Additions

The chosen markup additions resemble closely those used by the semantic Wikipedia -project [18] and semantic Mediawiki [65]. Similar semantic additions developed for MoinMoin [5] were investigated but deemed to include only a portion of the desired features.

The goal of the markup additions is not to implement the whole of the RDF notation, but to present the user a simple and intuitive way to make statements about a Wiki page. Statements can only describe the containing Wiki page in relation to page tag values, Wiki pages and URI resources. Semantic data is marked up within page content and rendered in a meaningful manner when the page is viewed.

There are two kinds of statements users can make about a Wiki page: MetaData statements and augmented link statements. MetaData statements are used to realise semantic page tags. They are implemented with a macro and therefore follow the MoinMoin macro syntax of the form `[[MacroName(arguments)]]`. The arguments of the MetaData macro consist of tag-value pairs with an optional third argument that omits the macro from page rendering. For example, the statement
`[[MetaData(SpecialPower, x-ray vision)]]`
on a superhero Wiki page denotes that he or she has the extraordinary ability to conduct airport security checks without external hardware, among other things.

Respectively, augmented link statements are used to implement semantic link tags. They extend the MoinMoin named link syntax forms
`[:OtherPage:Wiki page]` and
`[http://example.com URI resource]`
that create links with descriptive labels (see Figure 18). Augmented link syntax adds a link tag to this markup, resulting in links of the forms
`[:OtherPage:linktag: page]` and
`[http://example.com linktag: URI resource]`.

[5] http://theendmusic.org/programming/MetaDataPlugin

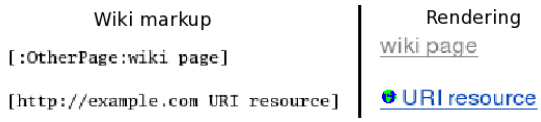

Figure 18. Rendering of normal MoinMoin links.

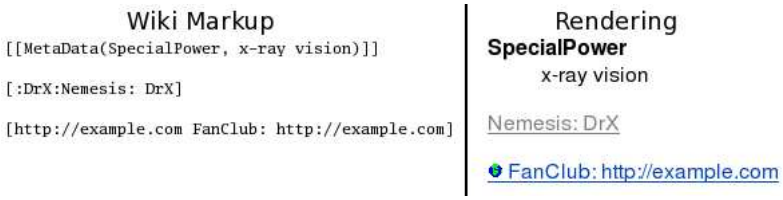

Figure 19. Rendering of semantic statements.

The special keyword “From” in the end of the type string denotes that the link is an incoming link, i.e. the referenced page links to the current page instead of the current page linking to it. For example, the statements
`[:OtherPage:linktagFrom: page]` and
`[http://example.com linktagFrom: URI resource]`
indicate that the current page is referenced by the Wiki page or the WWW page, respectively.

The statement `[:DrX:Nemesis: DrX]` on the superhero Wiki page tells that the nemesis of our hero is Dr. X, described in the same Wiki. Respectively,
`[http://example.com FanClub: http://example.com]`
states that the hero’s fan club has its web page at the URI http://example.com. Repeating the link in the descriptive string is not required, the examples do so for reasons of clarity only. Figure 19 illustrates the rendering of these statements.

The notation defaults to the namespace designated by the Wiki. To avoid collisions with regular Wiki pages, the pages describing the page tags and the link tags are prefixed with ’Property’. Thus, in the examples of the previous paragraphs, ’PropertySpecialPower’ and ’PropertyFanClub’ are pages in the same Wiki.

By editing the descriptions and semantic data on the Wiki pages describing the page tags and the link tags, the community creates a contract on the formal meaning of a domain - effectively an ontology. This lets the users freely edit the ontology in a very Wiki-like fashion, which reduces the entry barrier and encourages vocabulary growth and expressiveness. For example, users of the superhero Wiki can elaborate on the concept of special powers (i.e. the content of the ’PropertySpecialPower’ page), adding further information, declaring exceptions, and so forth. The availability of discussions on the subject, along with relevant links and multimedia, will help in understanding the concept. [66]

Graphingwiki is not planned to support any deeper semantic meaning to ontology entries. RDF schema or datatypes are not supported, nor are pages checked for consistency with any formalism. However, template pages can be used to create implicit meta-ontologies similarly as in Wikitology [59]. For example, a ’SuperheroTemplate’ could include statements common for all superheroes, so that when a page for a superhero is created using that template, the author is reminded about the kinds of semantic data that should probably be included.

The semantic markup supports namespaced statements. The list of valid namespaces is gathered from the Wiki's *InterWiki* list. For example, the statement
`[[MetaData(Wardrobe:JumpSuit, Spandex)]]`
tells us that the hero in question wears a flashy spandex jump suit, and that the specifics on the style of dress can be found in the Wardrobe Wiki. Respectively, the statement
`[wiki:WikiTwo/PageTwo OtherWiki:SeeFrom: wiki:WikiTwo/PageTwo]`
represents the situation where the page 'PageTwo' of the Wiki 'WikiTwo' has a relation with the referencing page defined by the page 'PropertySee' in the Wiki 'OtherWiki'. Naturally, by adding the line
`dc http://purl.org/dc/elements/1.1/`
to the *InterWiki* list of the Wiki in question enables the user to employ Dublin Core[6] definitions in the Wiki pages. Although *InterWiki* lists are currently not user-editable in MoinMoin, the *InterWiki* list provides a relatively clean and straightforward way to add new scope to Wiki editing. Graphingwiki uses the namespaces merely as URI prefixes to the resource names, the RDF data corresponding to the resource is not fetched. Still, the namespaced URIs offer some advantages, as users can use standardised semantic tags with well-defined meanings, some primitive inference rules involving different namespaces can be used, and external RDF tools can utilise the full scope of the external semantic data. The semantic data in the Wiki data can also be dumped from the Wiki in N3 notation for further analysis with external RDF tools.

4.1.4. Visualisation

As the development of Graphingwiki focuses on generating the protocol views of the PROTOS-MATINE method, a practical engineering approach has been taken on the visualisations. The visualisation style resembles closely the preliminary visualisations created during the PROTOS-MATINE project (see Figures 12 and 13). Further development of the visualisation style is beyond this work.

Visualisations are composed of the node of the current Wiki page, the links leading to the page and from the page, and the nodes depicting the linked pages. Alternatively, all pages belonging to a category of the current page can be used as the root nodes of the graph, instead of merely the current page node. Visualising a category shows a whole field at one glance, including the direct and indirect relations of all the members, along with their immediate surroundings.

Page tags can be used to colour the nodes of the graph, and pages can be filtered based on their tags. Respectively, augmented links are coloured with respect to their link tags, by which they can also be filtered. Filtering can greatly reduce the clutter in the visualisation, and helps in concentrating to desired aspects of the data. Graphs can also be ordered with respect to one of the page tags. The tag values are lexically sorted, determining the rank of the nodes corresponding to the pages. Colouring and ordering the nodes offers two dimensions by which to organise the semantic data. Figures 16 and 17 illustrate the visualisation style.

[6] http://dublincore.org/

4.1.5. Inference

While visualisation makes semantics comprehensible, inference makes it operational. Generally speaking, inference is used to extend the set of known facts with the help of rules that concern them, and to find the facts, if any, that prove a stated goal. Inference engines that take the first approach are called forward chaining, as they start from valid data, while backwards chaining engines start from the goal to be proved, and apply known facts and rules to produce a proof. [67]

A backwards-chaining inference engine is used to answer queries on semantic data. The engine uses Horn clause logic, i.e. clauses that do not have more than one positive literal, also used by many logic programming approaches such as Prolog. Horn clauses have desirable properties in that their satisfiability is solvable in polynomial time with algorithms linear to formula size. As the semantic data can be expressed in the terms of RDF triples, which are basically simple relations, it is straightforward to map them as clauses.

The inference rules and queries are stored as Wiki pages for easy editing and reference. The rules are expressed in the N3 notation, as Graphingwiki markup extensions do not include any way to express them. The result of the query is a set of RDF triples, also in N3 format, that maintain the conditions presented by the query. For example, according to the old adage “the enemy of my enemy is my friend” an evil mastermind, Dr. X, might want to query the superhero Wiki for enemies of his enemies to find new allies to battle his nemesis, Goody Two Shoes. The rule, representing Dr. X’s notion on enemies and allies, and the query would be as follows:

```
{?x Enemy ?y. ?z Enemy ?x} => {?z Ally ?y} .
{?who Ally DrX} => [] .
```

The query could result in the following reply:

```
CookieMonster Ally DrX .
DrEvil Ally DrX .
PowderedToastMan Ally DrX .
GoodyTwoShoes Ally DrX .
```

Having disproved the old adage, Dr. X curses his wretched query in frustration.

The results of the query could themselves represent a complex structure, especially in the case of comprehensive knowledge bases and badly formed queries. In these cases, the results could also be used as the input data of the visualisation engine for easier understanding and further processing.

4.2. Prototype 1 - Visualisation

The purpose of the first prototype was to test visualisation in a simple standalone tool. This way the possible complexity of an underlying Wiki system could be evaded, and focus could remain on implementing basic features and testing their feasibility.

4.2.1. Analysis and Design

As the focus of Graphingwiki is to provide for the protocol view for the PROTOS-MATINE method, data on protocol specifications was chosen as a starting point for the visualisations tested by the prototype. RFC specifications were the most readily available, and thus the body of data used as a starting point for the visualisations was derived from the RFC editor index [7] and summaries by Anne and Lynn Wheeler[8].

The first visualisations were contrived from the starting points of programmatical simplicity and similarity to the existing hand-made views. Graphs were selected as the most appropriate visualisation style for these purposes. Nodes in the graphs were chosen to represent specifications, the links between which were depicted by the edges. As these links are directed, they are represented with directed edges. In these edges of the form $\{A, B\}$, A is called the *parent* of B, and B the *child* of A, respectively. The graphs were limited to have only a single edge for a given parent, child pair.

The graphs for the visualisations were created by taking a set of nodes as the starting point and adding edges and nodes so that their parents and children are included in the graph. Ordering the nodes and highlighting their features as per earlier visualisations (see Figures 12 and 13) was defined as an additional feature to the visualisations. Node coloring was deemed a straightforward yet intuitive highlighting method.

As the first prototype was to be a throw-away system, it was decided to provide for a suitable environment for user interface experiments. As visual observability via graph exploration was seen as the focal feature of Graphingwiki, it was concluded that a similarly visual and interactive interface would be needed.

4.2.2. Implementation

The first implementation task was the automated data extraction from indices. The data collected includes standard name, type, date, status and linkages of different types to other standards. This data was processed with a Python script to a simple hash table data structure that was serialised to a file using Python's cPickle[9] module. The Python extension module pyparsing[10] was used to facilitate the handling of the index formats.

The third party tool Graphviz[11] was chosen to visualise the handled data. The Python extension module Pydot[12] was used to express the graphs and to interface with the tool. The ordered view was generated using the *dot* filter of the Graphviz tool, which uses an algorithm that produces hierarchical graphs. All the produced views were coloured and ordered by the type of standard and publication year of the standards, respectively. These values were hardcoded and could not be changed.

Graphviz proved a powerful and flexible tool with an impressive feature set. Despite concise documentation, occasional difficulties were encountered in its use. The most

[7] http://www.rfc-editor.org/
[8] http://www.garlic.com/ lynn/rfcietf.htm
[9] http://docs.python.org/lib/module-cPickle.html
[10] http://pyparsing.sourceforge.net/
[11] http://www.graphviz.org/
[12] http://dkbza.org/pydot.html

prominent of these was the ordering process of the graph, which is detailed here for posterity as related documentation is somewhat unfocused.

The nodes of the graph are treated first. Nodes are introduced to the graph to be output in hierarchical order, as this is assumed by the *dot* filter. The input order of edges is contrarily not relevant. Nodes of equal rank are inserted into their own subgraphs, for which parameters are set for equalising node rank. An invisible rank node is added to each of these subgraphs.

After setting up the nodes, edge parameters are set for restraining the layout according to the hierarchy. This procedure starts with assigning invisible edges between adjacent rank nodes. Parameters are set for keeping these edges as short as possible. After this, the edges of the input are taken into consideration.

An important part of introducing edges to the ordered graph is setting their minimum lengths to match the rank separation of nodes in the hierarchy. If the parent node has a lower rank, setting the edge length is sufficient. In the converse situation, the action taken is dependent on the existence of an edge in the opposite direction between the same nodes. If such an edge exists, the current node is inserted into the graph with parameters set for ignoring any constraints it might have on the hierarchy. Otherwise the edge is reverted in direction and inserted with parameters for minimum length and backwards direction arrow.

After these phases the ordering is complete and results in a graph where all the nodes have defined ranks and all the edges have lengths with respect to this hierarchy. Visualisation is then completed by setting any other desired visual parameters and getting an image in the desired format from Graphviz. This image could then be presented to the user.

The first prototype was implemented as a Common Gateway Interface (CGI) script, following the presumed implementation style of the final Wiki version. The user operated the prototype by the means of a web browser. User interfaces providing different levels of interaction were created. First, a minimal Graphical User Interface (GUI) was created. It used image maps to enable navigation of the visualisation by clicking the nodes in the graph. This approach works with any user agent and provides intuitive navigation but lacks any other means for modifying the visualisations.

To counter these faults and to increase interactiveness with the user, the GUI was augmented with more sophisticated features. They were implemented using Scalable Vector Graphics (SVG) images that could be dynamically changed with the Ecmascript scripting language. As SVG viewer was not integrated in any of the major web browsers at the time of prototyping, a separate plugin by Adobe Software had to be installed for the viewing task. The GUI had functions for filtering any given set of nodes and edges from the graph as well as selecting any set of nodes from a visualisation as the starting point of a new visualisation. The selection of nodes and navigating the graph were provided with the left and center buttons of the mouse, whereas the other functions resided in a context menu activated with the right button.

The graphs resulting from standard data sometimes proved too large to effectively fit the screen in such manner that they would be both comprehensible and readable. A major expanding factor for the graphs was the amount of space given to the edges by the *dot* filter. Thus, a visual notation for compressing the graph was attempted. The notation omitted some edges from the graph, showing them as augmentation of nodes

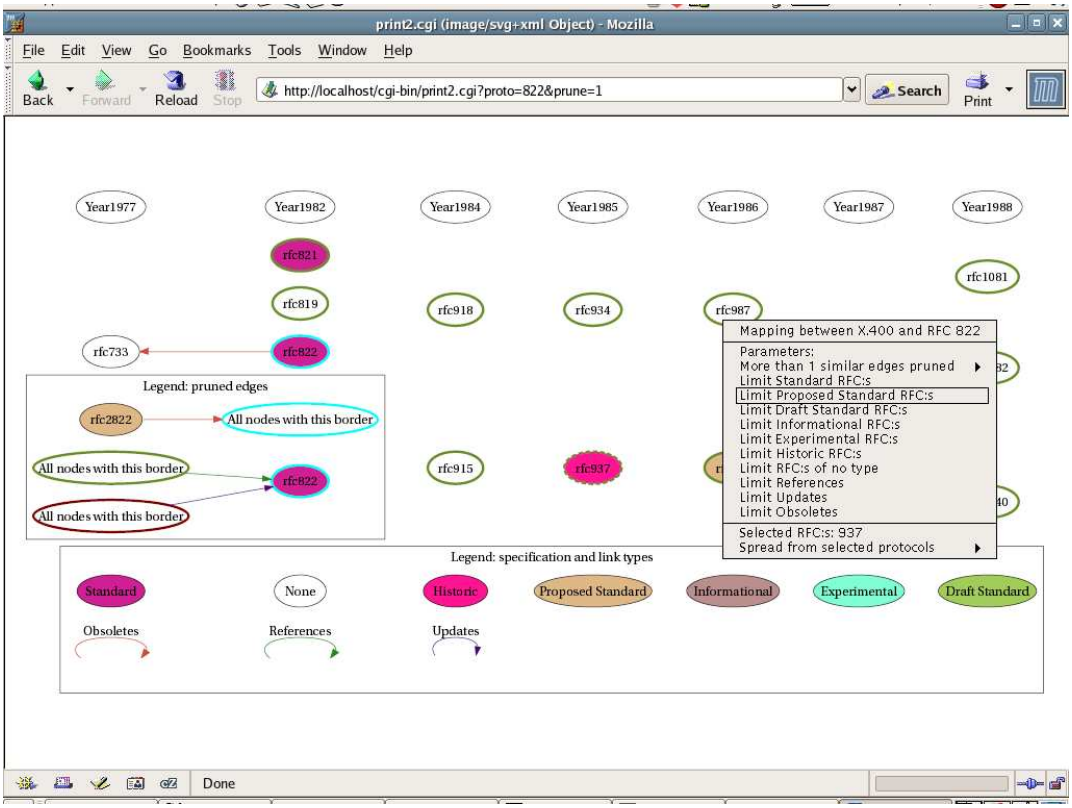

Figure 20. Example view by the the first prototype, using graph compression.

with a border of different colour and thickness. This greatly reduced the size of the resulting graphs.

Node coloring was implemented by selecting the color for a specific tag from a predefined list with a random lookup seeded with the Message Digest 5 (MD5) hash of the tag string. This way the tags get seemingly random yet static values. The generation of a list of distinct colors for this purpose proved to be problematic.

4.2.3. Experimentation

Figure 20 depicts an example visualisation created by the later GUI. A separate legend graph explains the notation used for edge reduction. The menu presenting the possible layout modifications is also visible.

The prototype was used by the PROTOS-MATINE project team to produce preliminary protocol views for some protocol families. The experiences on the chosen visualisation styles were positive with the exception of graph compression, which was seen unintuitive. Another approach of reducing graph size was attempted by grouping similar nodes into cluster subgraphs. This pursuit was quickly dropped as the current implementation of Graphviz does not fully support cluster subgraphs.

Experiences with the user interfaces were mixed. The minimal GUI was seen more usable and effective despite its very limited feature set whereas the context menu approach taken in the later development of the GUI was seen as unintuitive and laborious. In addition, the implementation of the GUI suffered from various hardships related to

the immaturity of SVG technologies. As it was at the same time limited in deployability due to the plugin installation requirement, the whole approach was dropped.

Based on the experiences and feedback from the first prototype, requirements for the second prototype were identified. They are presented in Table 4 in the order of importance.

Table 4. Requirements from the first cycle

Requirement #	Requirement	Priority
1	Wiki markup for semantic data	1
2	Parent/child visualisation of Wiki page data	1
3	Ordering, colouring, and filtering visualisations	2
4	Path visualisation	3

Requirement 1, Wiki markup of semantic data, is a direct result from the Wiki implementation choice; in order to visualise the semantic data of Wiki pages, there has to be a method for marking up the data in some manner.

Requirements 2 and 3, parent/child visualisation and modifying the visualisations, stem from the usage experiences of the prototype. The visualisation style was seen sufficient for creating the existing views with the aid of the ability to filter irrelevant data.

Requirement 4, path visualisation, is based on user requests for a feature that would show longer paths of linkage between a given set of nodes. In the last stages of development a quick implementation for this feature was added. As the use of path visualisation was seen to be marginal to that of the parent/child visualisations, it was assigned a low priority.

4.3. Prototype 2 - Wiki Integration

The main objective of the second prototype was to bring the experiences from the first prototype to the Wiki environment for visualising the page data. Additional markup was developed for including more expressive forms of data to be presented on the Wiki pages.

4.3.1. Analysis and Design

Creating a throwaway prototype proved a good decision in the analysis for the Wiki prototype, as it was seen that many of the tools selected for the first prototype were seriously lacking. Pydot was too slow for practical purposes and did not include some basic graph editing facilities. Additionally, much of the existing code suffered by insufficient separation of data from its representation.

Implementation of the second prototype was started from scratch. As the requirements for the visualisation from the previous cycle were quite clear, the analysis and design was concentrated on the details of the Wiki markup and architecture. These are detailed in sections 4.1.2 and 4.1.3. The semantic data in each Wiki page was chosen

to be stored into a file of its own, in a symmetrical manner with the page data storage in the MoinMoin Wiki.

4.3.2. Implementation

The first implementation task was to establish a storage form for graph data. A general-purpose graph library was created for this purpose. Next, the Wiki markup was formalised. The third task combined the results of these tasks, resulting in the interpretation and storage of semantic data with the help of existing and augmented Wiki markup. The semantic data was serialised in the defined graph format.

As the markup allows for incoming links links that are not shown on the wiki page itself, a global database of page linkage was also implemented using Python's shelve[13] module that provides object persistence. File locking, as provided by the lockfile command of the procmail[14] suite of tools, is used to prevent simultaneous writes that are not supported by the shelve.

The next task was to visualise the saved graphs. The latest versions of Graphviz included Python bindings for its graph generating and formatting routines, which were taken into use. The first step taken in this task was the generation of wrapper libraries to abstract the visualisation from the viewpoint of the saved graph format. This also streamlines the generation of views, as layouts can be generated from any graph source in a straightforward manner using the wrappers.

This stage of implementation suffered from many problems as the Python bindings were at a very early stage. Various bugs were found, for which bug reports were filed to the developers of Graphviz. Waiting for new enhancements and testing out unstable fixes resulted in weeks of delays in schedule.

As the implementation of the wrappers was at a standstill, alternate methods for visualisation were explored. An interface for generating visualisations with Dynagraph[15] was implemented in the wrappers. Dynagraph also employs the Graphviz layout engine, but concentrates on providing highly interactive layouts using a client-server architecture. It has also been used for the handling of huge graphs[16]. After initial experiments this trail of development was discontinued, for Dynagraph did not implement all the needed functionality, namely subgraphs.

As a result from the delays, only parent/child visualisations were implemented in the second prototype. This work was concentrated on creating filtering functionalities and enabling further enhancements. The ordering and colouring of views was made possible with respect to any semantic aspect. Unordered views were created with *neato* filter of Graphviz that draws unordered graphs using a spring model layout.

The visualisations employ a specialised forward chaining inference engine that uses pattern matching operations [67] to create visualisations by selecting properties of graphs assembled from the semantic data in the pages. Use of the engine presents opportunities for creating supplemental views for various, even highly specific purposes.

[13] http://docs.python.org/lib/module-shelve.html
[14] http://www.procmail.org/
[15] http://www.dynagraph.org/
[16] http://gordon.woodhull.com/dinograph/

4.3.3. Experimentation

The scripts that provided the data for the previous cycle were modified to save the RFC data as wiki pages. Previous views were then repeated with the new system. A series of housekeeping scripts were created in the process to manage the graph and index files used by the prototype. These files and their formats were under constant modification, and the scripts were used to derive them anew from the page files. Another set of pages was created to test the compliance of the prototype to various types of links provided by the MoinMoin Wiki.

Ideas for further visualisation types were searched from the field of social networking analysis. Graph data was output with small scripts to a format suitable for exploration with the Pajek[17] network analysis program. However, no generally suitable visualisation forms were discovered.

The second prototype provided requirements 1, 2 and 3 from the previous cycle. As development progressed along expected lines, new requirements were added as per earlier plans and experiences as summarised in Table 5.

Table 5. Requirements from the second cycle

Requirement #	Requirement	Priority
1	Wiki markup for semantic data	Done
2	Parent/child visualisation of Wiki page data	Done
3	Ordering, colouring, and filtering visualisations	Done
4	Path visualisation	3
5	Improved user interface	1
6	Variable starting points and depths for visualisations	1
7	Rule-added queries by inference	2
8	Visualisation of inference results	3
9	N3 export of semantic data	3

The second prototype enabled only the most basic user interaction, making requirement 5, improved user interface, quite self-explanatory. Based on earlier experience, users should be presented with simple and clear methods for customising the visualisations.

The ability to choose variable starting points as in requirement 6 was implemented by the previous cycle but was left out of the second prototype as its schedule became overdue. It was however requested by users along with views that go deeper in the parent-child relationships of the data. With depth n, not only the parents and children of the start pages are included in the visualisation, but also the grandparents and grandchildren, up to n generations.

Requirement 7, rule-added queries by inference, was an integral part in the original problem specification. It was further augmented by requirement 8, visualisation of inference results, which quite simply aims to show the statements resulting from the queries as a graph, in a similar manner as other visualisations, and including the same view modification and exploration features.

[17] http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Requirement 9, N3 export of semantic data, was added as it was realised that there is a great wealth of tools to process semantic data in this format. The ability to process the data of Graphingwiki with these tools is an easy way to increase the scope of the tool, while enabling the comparison of the results acquired by it to those of third party tools. An additional factor for the selection of N3 for the output data format was its syntax, which is easy to generate and parse, yet legible from people.

4.4. Prototype 3 - Usability and Finalisation

While the previous cycle was mainly concerned with enabling collaboration on semantic data, the goal of the third prototype is to provide easily usable interface to that data along with extra functionality to benefit the user. Carrying out inference queries represents the main new feature of the cycle.

4.4.1. Analysis and Design

After the failed attempt in the first cycle for providing a sophisticated and interactive user interface, the aim for the GUI in the third cycle was simplicity both in implementation and usage. Thus a simple HTML form using HTTP GET was selected as the method of implementation. This method has the downside of requiring communication, usually page reload, with the server for each user interaction. On the other hand, forms are a de facto method of user interaction with HTML, and users are highly accustomed to it.

The Wiki concept of categories was decided to be utilised for the variable starting points for visualisations, as per requirement 6 from the previous cycle. Wikis usually use categories to represent a group of pages treating similar or related subjects. Thus it was decided that visualising a whole category of pages in one view could prove highly useful. An additional feature for adding visualisation start pages on a whim was also deemed necessary. Path visualisation, requirement 4 from the first cycle, was thought to be trivial to realise using the pattern matching engine implemented in the previous cycle.

The N3 export was a self-evident feature that needed no additional analysis, but it lead to the selection of N3 also as the query syntax for the purposes of symmetry and interoperability. The inference engine itself was the subject of much analysis, on which more details are available in section 4.1.5. A simple unifier-based implementation in the style of many Prolog implementations was chosen as a basis for the inference module [67].

4.4.2. Implementation

As expected, the creation of the form GUI ran without major difficulties. New versions of the prototype were quickly adopted by users in the OUSPG. This resulted in the rapid improvement of the prototype, as bugs were discovered and new GUI options were requested. Variable starting points, filtering, and other options were implemented

without further ado. Output formats were also added for SVG and the dot language of Graphviz.

Further issues of browser incompatibility with the GUI manifested with the widening of the user base. To ease the implementation, the visualisations were embedded to HTML pages with image tags containing a data URI. Contrarily to normal image tags which simply state the URI of the image, this URI scheme defined in RFC 2397[18] includes the entire image data in base64 encoding. However, all browsers do not support the data URI scheme. Due to this restriction, another version of the GUI was created using the MIME HTML scheme as defined in RFC 2557[19]. Using both these schemes seems to guarantee an adequate browser coverage.

The implementation hit another snag with path visualisation, as the pattern matching engine proved sub-optimal with long-spanning relations, resulting in unacceptable response times for the efficient usage of the prototype. Thus, the path visualisation was implemented a simple Moore's breadth first search algorithm that can be used to search either for the shortest path or all paths between a given set of nodes [68]. This approach was based on the experiences from the first cycle.

The creation of the inference engine was approached carefully. The N3 export of statements was created first, as it mainly involved printing the data that already existed in the graph using a slightly different syntax. After this step, inference capabilities were tested upon with euler.py[20], a third party reasoner component that works on the N3 formatted data. As the initial experiments were successful, work on a custom inference engine was commenced. As Prolog-style reasoning is a well-established technique with various reference implementations, creating a simple inference module was not a daunting task.

4.4.3. Experimentation

As the third prototype enjoyed the attention of an extending user base, it was employed in several usage scenarios with diverse automatically extracted and hand-made data. As an example, Figure 21 depicts a visualisation made by the prototype with data automatically extracted from the WiFiPedia wireless standard resource.

The inference features are relatively fresh and have not yet seen much use. Although a quick experimental implementation for requirement 8, visualising the results of inference, would not have required much effort, the work was deferred to be done after further analysis. As the third prototype already utilises two different approaches to create visualisations, analysis is required for determining if any of these approaches might be unified. One possible scenario would be to create all the views used in Graphingwiki by the inference engine, so that the now pivotal parent/child visualisation would be downplayed to a case among others.

In addition to this observation, experimentation with the third prototype spawned new ideas and uncovered pressing problem areas. These are summarised in Table 6 as the requirements from the third prototyping cycle.

[18] http://www.ietf.org/rfc/rfc2397.txt
[19] http://www.ietf.org/rfc/rfc2557.txt
[20] http://www.agfa.com/w3c/euler/

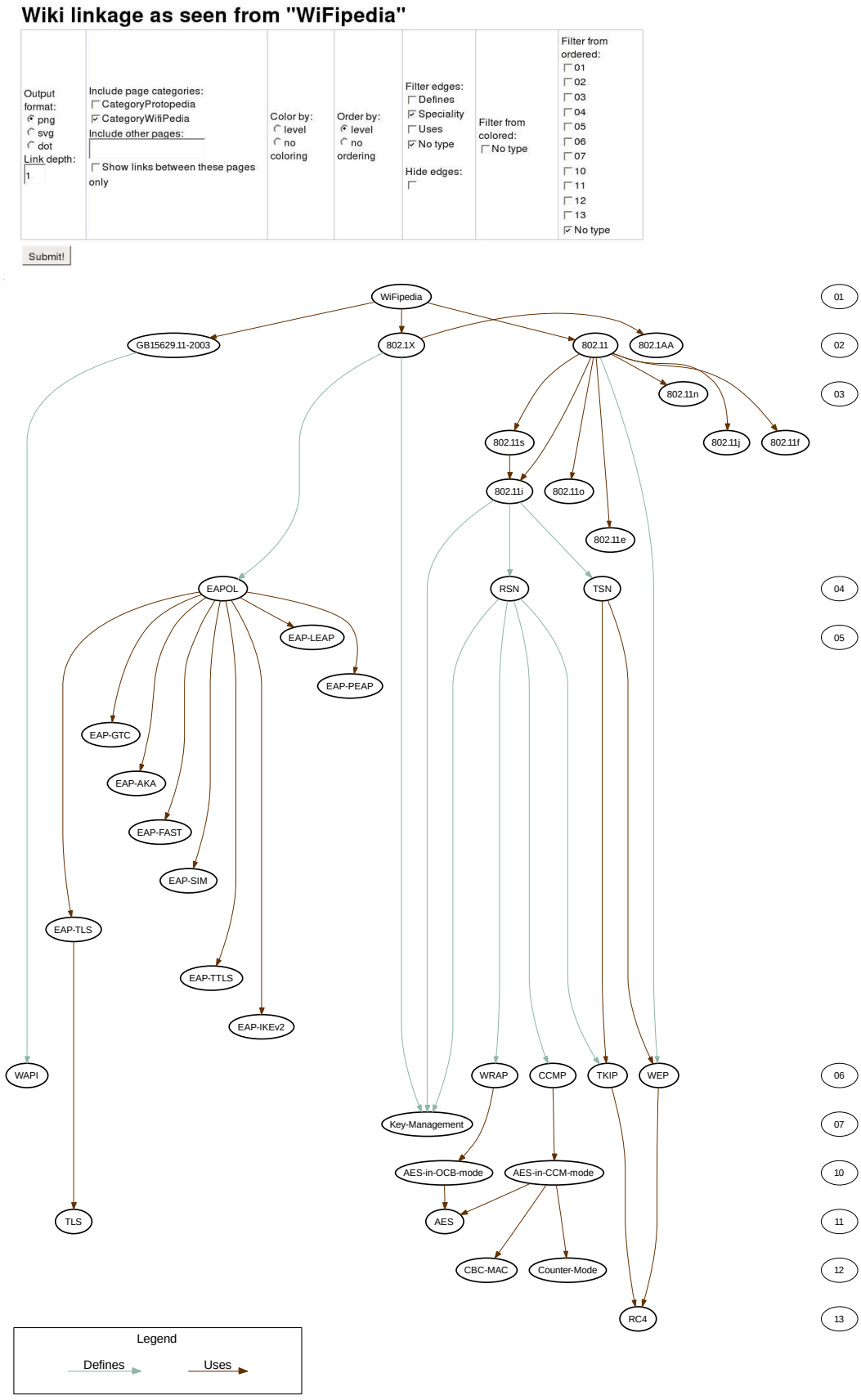

Figure 21. Visualisations of wireless networking standards using data from WiFipedia.

Table 6. Requirements from the third cycle

Requirement #	Requirement	Priority
1	Wiki markup for semantic data	Done
2	Parent/child visualisation of Wiki page data	Done
3	Ordering, colouring, and filtering visualisations	Done
4	Path visualisation	Done
5	Improved user interface	Done
6	Variable starting points and depths for visualisations	Done
7	Rule-added queries by inference	Done
8	Visualisation of inference results	3
9	N3 export of semantic data	Done
10	Compression of visualisations	1
11	Testing and optimisation of the inference engine	2
12	Functional additions to the inference engine	3
13	Datatypes for semantic data	3
14	Page specific visualisation parameters	3

Requirement 10, compression of visualisations, represents the main limitation of the third prototype. Wider usage of the prototype quite expectedly revealed that some visualisations can be too large to be grasped, even with the means of filtering out irrelevant data. Suggested improvement methods for the compression of visualisations include adding more efficient node grouping and folding away unimportant parts of the graph to make the resulting graphs smaller and clearer. However, it is clear that any major improvements on the visualisations requires additional analysis.

The inference engine of the third prototype is a relatively fresh implementation that has undergone only basic testing. It will yet require much testing and refactoring to become mature enough for general use. An important part of this work is the analysis of methods for making computationally feasible reasoning despite the threat of combinatorial explosion. The inference engine of the current prototype is also expected to suffer from performance issues, especially with increases in the size and the degree of the nexus of a Wiki. These bounds can be especially salient with inference queries that affect a major part of the Wiki data. However, there are various well established optimisation techniques for unifier based inference engines. The testing and optimisation work for overcoming the stated limitations constitutes requirement 11.

The inference engine in Graphingwiki will also require some work to be fully operational in a practical manner. Requirement 12, functional additions to the inference engine, was introduced to address this need. A major part of this work includes creating the basic queries representing the common use cases of the inference engine. It also includes queries with additional functionality such as “find all of the links from the Wiki that point to non-existing pages”.

Requirement 13, datatypes for semantic data, aims for a practical way for adding datatype support to Graphingwiki. Early analysis for the implementation of datatypes includes examining the input semantic data for adherence with specified datatype patterns and saving the matched semantic data with this knowledge. Together with re-

quirement 12 this facility would greatly increase the expressive and operational power of semantic data in a Wiki.

As the user base of Graphingwiki has grown, the need for using its visualisations for presentation purposes has emerged. Requirement 14, page specific visualisation parameters, has been requested by users desiring to customise the outlook of their views. The requirement states that visual layout parameters, such as colours and shapes, are to be presented on Wiki pages for easy modification. This feature increases the clarity of the visualisations while minimising the efforts needed for modifying them with external tools for presentation purposes.

4.5. Prototyping Conclusions

The prototyping cycles have shown how the MoinMoin Wiki can be extended to include some semantic capabilities. Graphingwiki uses the MoinMoin plugin mechanism along with its existing capabilities of linking and category pages to create a simple and lightweight semantic tagging scheme. The tagging scheme was further used to provide for the visualisation of semantic data and making reasoning upon it.

Most of the components of Graphingwiki experienced extensive development during the prototyping cycles. The functions used to interpret and store semantic data were the ones enduring the greatest mutation. The initial plain hash table of attributes with predefined sets of keys and values was evolved into graphs and indices of user-defined data that could be freely worked upon. The storage format experienced subtle changes throughout prototyping.

The visualisation style evolved very little during the development cycles. The initial style proposal developed in the first cycle was deemed clear and sufficient for the purposes of Graphingwiki. The unordered view implemented during the second cycle became the only major style-related addition. It was found illustrative for some data sets. Other visualisation related changes in the development cycles involved the parameters for the views.

Graphingwiki has been tested with Wikis containing up to 4000-5000 pages. The operation of the tool has not suffered greatly from problems related to scalability or performance with these data sets. In some cases, non-trivial inference queries suffered from performance constraints that rendered them ineffective. Rework on parts of the implementation is expected to remove most of these problems. The main limitation of the visualisations was their expansion with massive data sets. Some of these visualisations became too large to be effectively handled by the current functions of Graphingwiki.

The requirements of Graphingwiki evolved among the prototyping cycles as follows: the first prototyping cycle implemented the vague requirements for simple visualisations using a predefined data set. As these visualisations were found effective, requirements were laid out for transferring the experiences of the first cycle to the Wiki environment. The second prototyping cycle was mainly concerned with building a Wiki integrated framework to provide functionality, parts of which were identified as requirements. The third prototyping cycle then went on to implement this functionality that was mainly related to user options and inference.

Table 7. Application features compared to Graphingwiki requirements

Applications	R1	R2	R3	R4	R5	R6
mSpace	-	-	X	X	-	-
Gzz	-	X	-	-	X	-
Gnowsis	-	-	X	X	-	-
SemperWiki	-	-	X	X	-	X
Fenfire	X	X	-	-	X	-
Rhizome	X	-	X	-	-	X
Makna	X	-	X	X	-	X
KendraBase	X	-	-	X	-	X
Semantic Mediawiki	X	-	X	X	X	-
WikSAR	X	-	-	X	X	X
MoinMoin Wiki with Graphingwiki	X	X	X	X	X	X

The Graphingwiki tool that resulted from the prototyping cycles includes the basic features needed for collaborative management of semantic knowledge. Table 7, a new version of Table 1 on Page 24, illustrates how the identified requirements are implemented by different tools. The requirements were: R1) supports the iterative collaboration of a large body of experts, R2) creates visualisations which can be interactively adapted to the needs of the user, R3) is widely available, enabling diffusion to a wide user base, R4) is easy to use and does not require any substantial training, R5) enables the creation of a data model based on the content, and R6) has advanced semantic querying or rudimentary inferencing abilities. The MoinMoin Wiki enhanced with the Graphingwiki extension is included in the comparison. The MoinMoin Wiki implements the requirements R1, R3, and R4, which Graphingwiki augments by implementing the requirements R2 and R5. Consequentially, Graphingwiki implements all of the requirements for collaborative knowledge management identified in the scope of this thesis.

5. DISCUSSION

The objective for this thesis lies within the introduced concept of protocol dependency in the context of critical information infrastructures. The main goal of the work is to provide the risk assessment tools necessary for the management of these dependencies and the vulnerabilities they evoke.

The most important contributions of this work include the PROTOS-MATINE method for the management of protocol dependencies and the Graphingwiki tool to support the use of the method. Collaborative knowledge management with interactive visualisations is the main process used to fulfil the objective of assessing dependency related risks and vulnerabilities.

5.1. Implications of this Research

Visualising the relations of protocols has proved to be an effective method for understanding the scope of a single protocol in application and network contexts. Visualisation of protocol data gathered from standards has been used in various stages of protocol-related vulnerability work, such as giving direction to communications related to an existing vulnerability, making cost-benefit analysis on a protocol test suite and shedding light on the scope of applications that should be included in the testing process.

Graphingwiki was used throughout its development to produce protocol views for the PROTOS-MATINE project. The data was gathered from specifications and similar data sources both by hand and by automated methods. Some of this data was also refined by domain experts.

The gathering and visualisation of information was found straightforward using the Wiki, and the visualisations successfully disclosed hidden dependencies in the data set. The views highlighted problem areas, such as protocols that are abundantly depended upon, baroque relations between protocol families, and the inherent complexity of modern networks.

This is exemplified by Figure 22, a visualisation on the Label Distribution Protocol (LDP) used by the Multi-Protocol Label Switching (MPLS) family of protocols. The view is apt in illustrating the point that becoming familiar with a technology by reading all related specifications might sometimes not be a desirable option. Half of the standards related to the LDP protocol are not displayed in the view due to paper size limitations.

Figure 23 is a visualisation created from CVE vulnerability data on virus scanners. The links in the view point to the vendors whose products the vulnerabilities have affected and the file formats whose handling has exhibited the vulnerabilities. Surface inspection on the view shows error prone file formats and vendors whose products have suffered from vulnerabilities.

It can be readily noted from the view that the handling of Roshal Archive (RAR) file format has resulted in most of the publically disclosed vulnerabilities. This further suggests that archive file formats may be among the foremost to feature vulnerabilities, which makes them prime targets for malicious attack. On the other hand, the lack of exposure for file format may also suggest that it may possess latent bugs. Similarly, the

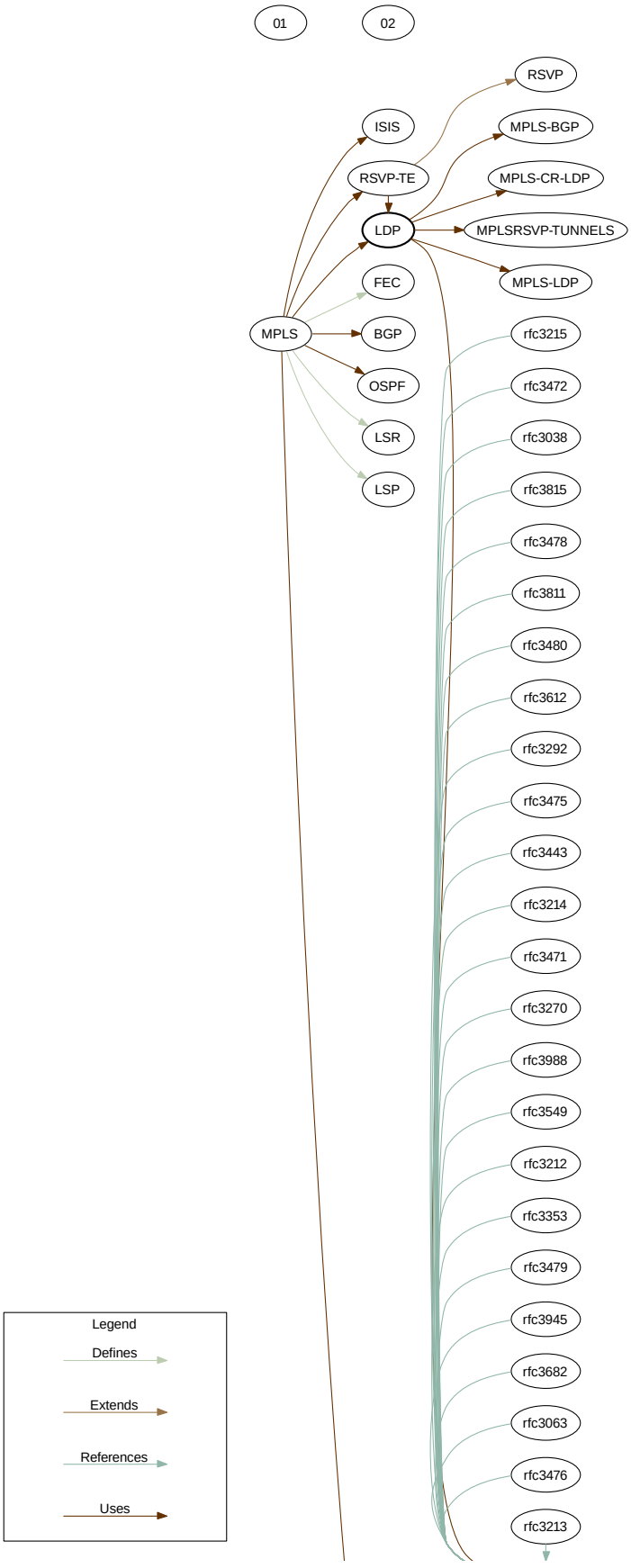

Figure 22. A visualisation of the LDP protocol exemplifying complexity of standards.

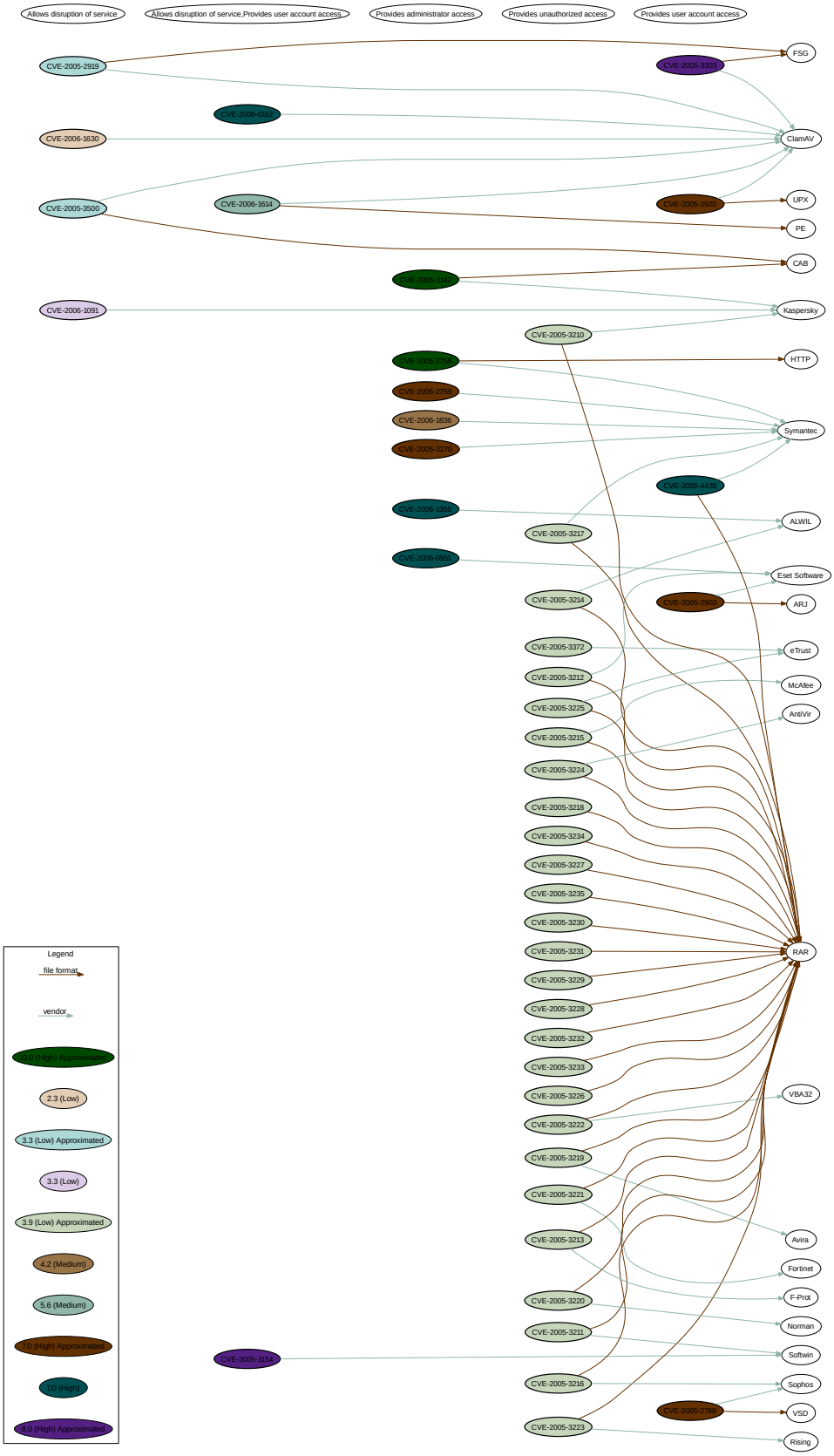

Figure 23. A visualisation on CVE vulnerability data.

lack of disclosed vulnerabilities in an implementation may suggest that its code has not been subjected to strenuous testing. Analysis on the view presents thus a viable testing strategy for antivirus software. It also sheds light on the impacts of the displayed vulnerabilities.

Based on the experiences of the PROTOS-MATINE project group, it can be stated that Graphingwiki performs well in its task of composing the protocol views of the PROTOS-MATINE method. In previous research, the same kinds of visualisations have been laboriously crafted by hand. Hand-made visualisations are often difficult to change, and they do not necessarily include any direct reference to their source data. This work is enhanced by Graphingwiki as the visualisation task is automated and the views are always based on accurate and available source data, which is accessible through the view. Therefore, the attention of researchers can remain on the essentials tasks of gathering and maintaining research data.

5.2. Limitations

In practice, the PROTOS-MATINE method introduced in this thesis consists of faceted views for the analysis of protocol related dependencies. However, only the formation of the protocol view has been handled in the scope of this work. This represents the main limitation of Graphingwiki, and further analysis is needed for the applicability of the tool for forming other views of the method.

During testing and preliminary usage, Graphingwiki has been employed in Wikis that have had up to 4000-5000 pages of data. As there are no usage experiences with larger data sets, no guarantees of scalability to immense bodies of data can be made. Particularly the inference capabilities may be susceptible to this limitation.

A related constraint is the handling of visualisations on extensive data sets. The existing usage scenarios on such sets have exposed cases where visualisations have become either overly expansive or cluttered beyond recognition. Thus, the effectivity of current functionality for handling visualisations on larger sets of data is suspect. Furthermore, this may be taken as an indication of inherent vulnerability of the system under inspection due to excessive protocol dependency.

The possible uses of Graphingwiki may be further limited due to discrepancies between user expectations and the functionality provided by the user interface. Some of the usage to date exhibits this phenomenon as users have contrived elaborate workarounds to achieve views that had not been provided for by the GUI. Ironically, Graphingwiki has entered the domain of creative features as described in Section 2.1.3.

5.3. Future Research

The main focus of future research is to improve the applicability of Graphingwiki for the other views of the PROTOS-MATINE method. This task includes gathering the data to be used in the formation of these views and analysis on the requirements for the efficient automated visualisation of this data. In effect, the efforts required for this task may surpass those of this thesis, amounting to several man-months. More accurate estimates require further analysis. However, the benefits of using Graphingwiki

as a tool to mitigate the pitfalls of the PROTOS-MATINE method appear promising and warrant this effort. The greatest benefits to be gained using the tool are related to expert interviews. They include the reduction of the transcription overhead and minimising the laborious feedback loop between forming views and receiving expert input on them.

Graphingwiki itself could be enhanced in a variety of ways to increase its efficiency and expressiveness, and to make it more approachable for users. The most important identified areas of enhancement are listed below.

1. *Increased ontology support.* A full support for different levels of ontology formalisation would be an obvious benefit, along with mechanisms that check the page's adherence to a specified ontology [8][59]. RDF schema to manipulate typed data could be added, as well as some OWL features. Many of the implicit Wiki relations, such as being part of a certain category or being made with a specific template, could be formed explicitly with these facilities. Importing RDF data related to instances of other namespaces would also increase the application scope of Graphingwiki.

2. *Visualisation and navigation enhancements.* The visualisation style and the GUI would benefit from user interaction studies and research on other visualisation styles. Different dimensional views such as Zzstructures and Polyarchies could be used to produce more data-compact views [47]. Wiki pages could include navigation section of related links created with the help of faceted classification [38] [19].

3. *Automatic generation of semantic data.* Some of the semantic data in a Wiki could also be automatically generated from the knowledge of who created and modified the page, creation date, data on referring page given by the browser, and so on [59]. Similarly, page categories could be automatically suggested to the user by comparing the page with representatives from existing categories using Bayesian classification.

4. *Tagging scheme enhancements.* The creation of ontologies might be easier and more scalable if users could first use the augmented link syntax to denote all statements, shifting to use the MetaData-macro only when it has been ascertained that the values of the link tags do not have further structure and can be considered to be mere tag value data.

5. *Improvements in user interaction.* Users could be greatly aided by the creation of semantic data macros specific to their domains of knowledge. Further, the user interface could include tag word suggestions to help converge the tagging scheme, similarly as in the del.icio.us service and the Makna semantic Wiki. Another aid for the tagging scheme would be the use of synonym-declaring relations. However, experiences from Wikipedia suggest that problems regarding the selection of tags are not critical, and that the situation is further ameliorated by the Wiki pages describing the tags [18].

6. *Visual content creation.* Currently, the semantic data in Graphingwiki is created by editing the Wiki pages and only displayed by the visualisations. Users of

the tool would be greatly aided by enabling the modification and creation of semantic content also by the visualisation user interface.

7. *Visualisation-aided versioning.* Many common use cases of Wikis, such as systems documentation and contracts, can encompass a smorgasbord of pages while placing great demands for the trustworthiness of the included data. As Wiki pages are by nature under constant revision and refinement, these use cases require facilities for specifying the set of page versions that constitute the desired state of the concept. Visualisations that are bound to specific page revisions could be used to facilitate version control of such concepts while making their structure easier to apprehend.

   Encapsulating the revision state of Wiki pages with visualisations in this manner is comparable to the transition of software version control from the per file Revision Control System (RCS) into the set oriented Concurrent Versions System (CVS). Whereas in software development module hierarchy facilitates revision tagging, in non-hierarchical Wikis the visualisations can provide for a one click snapshot of a larger concept. Following the evolution of these visualisations could give insight into the development of the concept and the processes involved.

Items 3, 4, and 5 should be implementable within a man-month. Estimate for the efforts of implementing items 1 and 7 should be no more than 2-3 man-months, although more accurate estimation would require proper analysis and design. Estimating the effort needed by items 2 and 6 is rather difficult, as they include additional research and implementation methods that have not yet been analysed.

5.4. Further Applications

The visualisation of information is a difficult task for which a variety of solutions have been implemented. These solutions range from general purpose tools to highly application specific frameworks. Drawing tools such as Dia[1], Visio[2], and SmartDraw[3] are among the most generic of these tools. They include facilities for creating various types of visualisations for diverse purposes.

There are also domain specific tools for visualisations. Mind mapping tools facilitate ideation by visualising an entirety related to a keyword or a concept. Technical planning applications use various forms of visual modeling, such as UML, SA/SD, flowgraphs, and network diagrams, to depict a system in an understandable manner. Project planning tools employ various visual aids such as organisational charts, gantt charts and timelines. In short, all these tools aim to aid the management of domain specific knowledge with the help of visualisation.

Semantic Wikis are a natural placeholder for various kinds of domain-specific data that are produced in normal course of work, enabling collaboration and groupwork. It has been claimed that semantic tools also have applications in learning by evaluating,

[1] http://www.gnome.org/projects/dia/
[2] http://www.microsoft.com/office/visio/
[3] http://www.smartdraw.com/

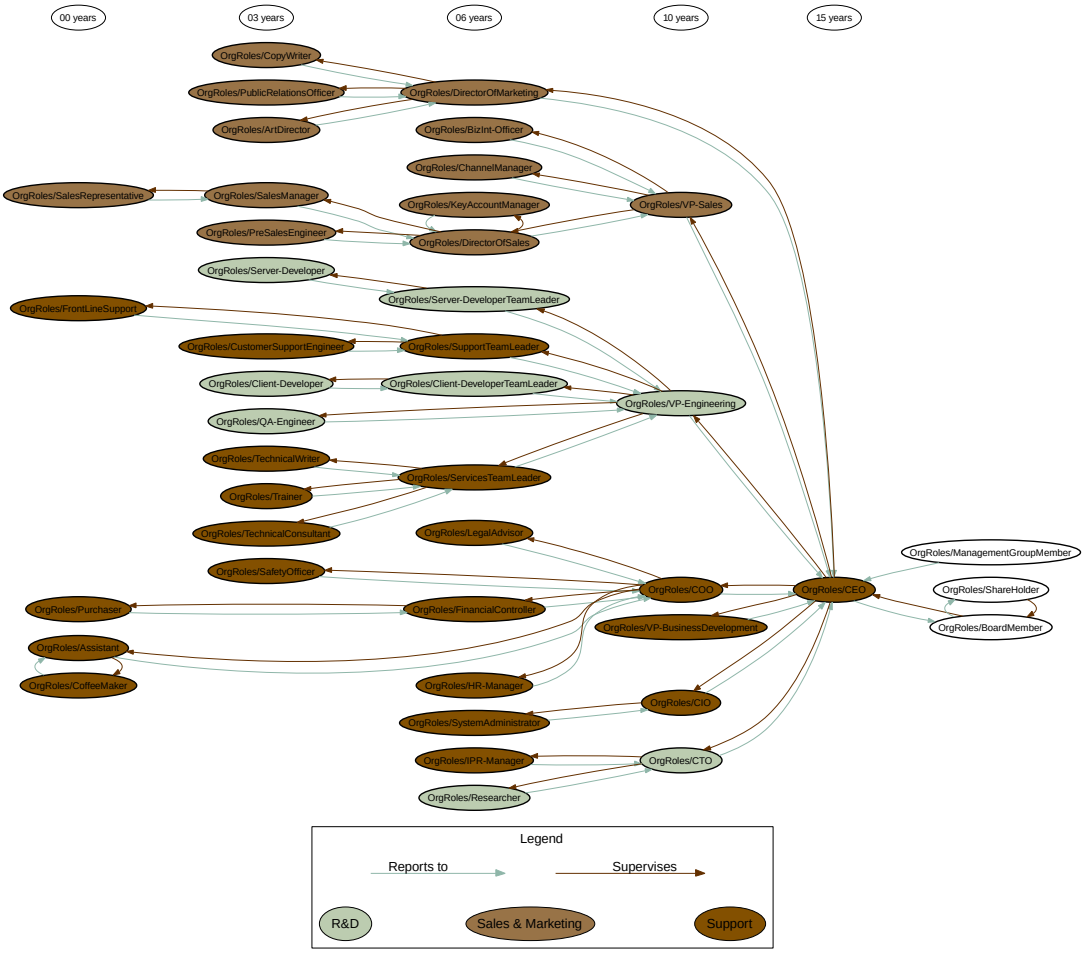

Figure 24. An organisational chart created with Graphingwiki.

manipulating, and presenting data in new ways [57]. Visualising this data according to the requirements of a given domain presents an effective method for making its contents easier to grasp by humans.

Consequently it is no surprise that in addition to fulfilling its intended purpose for creating the protocol view of the PROTOS-MATINE project, Graphingwiki has proved to be useful for a variety of other tasks. New application areas emerged at a constant rate during its development, indicating that there is a great need for lightweight information visualisation facilities. Some of the application areas are illustrated by the examples in the following paragraphs. Also the architecture diagrams of Graphingwiki presented in this work (see Figures 16 and 17) represent a suitable application domain.

Figure 24 is a organisational chart of a company that has been created with Graphingwiki. The nodes of the graph represent the roles of different employees while edges report the reporting and management chains between the roles. The roles are ordered by their required experience and colored according to the departments they belong to. Similarly, Wiki pages containing data on employee responsibilities and fields of know-how could enable efficient resource management and aid in problem resolution. Social network mapping techniques could be used on this data even further, for example to identify communities and communication bottlenecks.

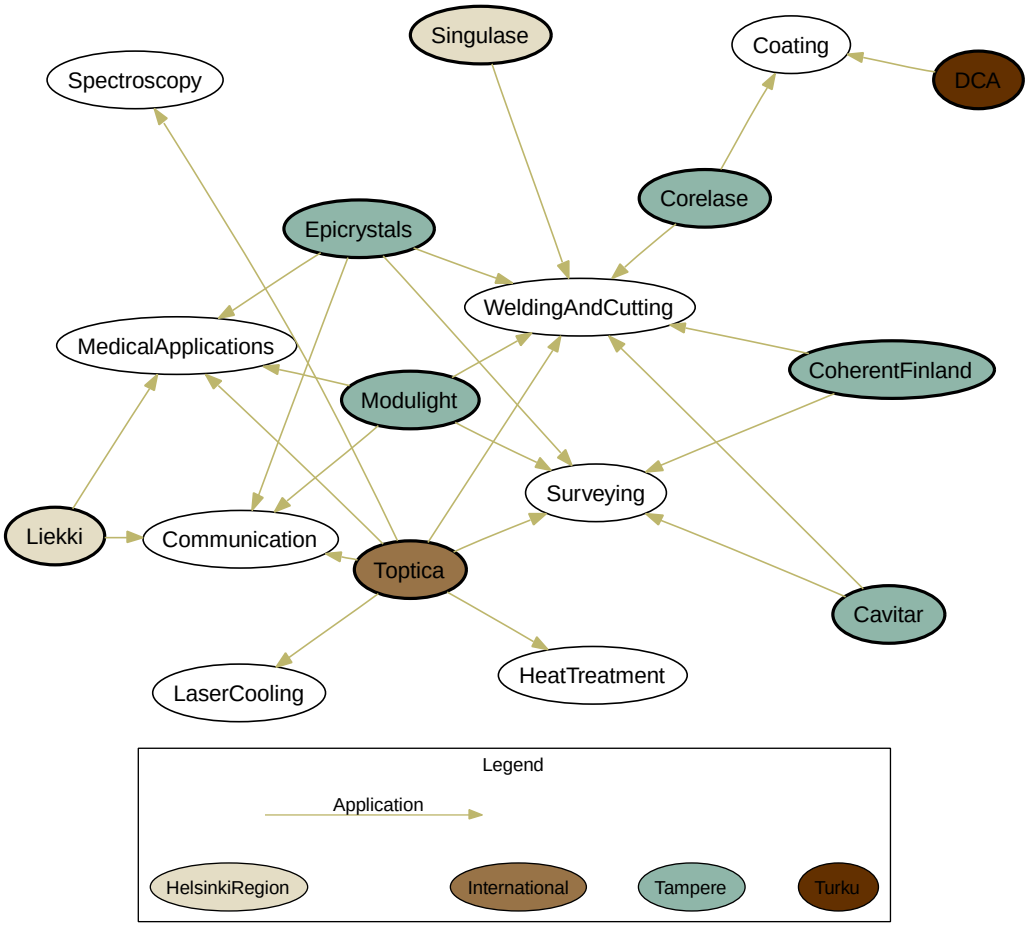

Figure 25. Different laser applications, their producers, and locations.

Figure 25 represents a survey on the research on laser technologies and on the manufacturers of laser products. Data on different actors of the field was inserted to a Wiki, along with their relations. This view on the Wiki data depicts the Finnish laser product vendors by location, with links to the application areas of their products.

6. CONCLUSIONS

This work introduced the PROTOS-MATINE method for handling dependencies between protocols, and the Graphingwiki tool to be used in the method for the handling of protocol related data and visualisation of dependencies among protocol specifications.

Graphingwiki was used by the PROTOS-MATINE project group for knowledge engineering in the domain of network protocols. The protocol views created by Graphingwiki have proven to be an effective aid in discovering dependencies between protocols, while the reasoning capabilities showed promise for shedding light on complex relationships of the semantic data. The visualisations have been used in various stages of protocol-related vulnerability work.

Future research on Graphingwiki include analyses on the visualisation style and user interaction methods in the tool. This research could result in more compact and easily manageable views. Another future direction is the inclusion of more sopisticated semantic features, the lack of which currently limits the use of Graphingwiki with other semantic tools and data sources.

During the development of Graphingwiki, a great demand was noted for the management and visualisation of data from diverse domains. Usage of the tool was then attempted in a number of application areas. Initial experiences on the applicability of Graphinwiki for purposes outside its intended domain of application were very encouraging.

Therefore, a similar approach to handling, visualising, and inferring on data would probably be of much use in many other domains, including enterprise resource management and social network mapping. Organisational human resources related skill and social network mapping and documenting information systems from deployment level to strategy view with dimensions on security policy and system interdependencies are examples of envisioned use cases.

7. REFERENCES

[1] Hagelstam A. (2005) CIP - kriittisen infrastruktuurin turvaaminen. Käsiteanalyysi ja kansainvälinen vertailu. HVK Julkaisuja 1/2005, Huoltovarmuuskeskus, Helsinki. Electronic version: http://www.huoltovarmuus.fi/documents/3/CIP-raportti_final.pdf.

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

[3] Beizer B. (1990) Software Testing Techniques. International Thomson Computer Press, 550 p.

[4] Koubatis A. & Schönberger J.Y. (2005) Risk management of complex critical systems. Int. J. Critical Infrastructures 1, pp. 195–215.

[5] Wenger A., Metzger J. & Dunn M. (2002) International CIIP Handbook: An Inventory of Protection Policies in Eight Countries. Center for Security Studies and Conflict Research, Zurich, Switzerland, 217 p. Electronic version: http://www.isn.ethz.ch/crn/_docs/CIIP_Handbook_2002_bw.pdf.

[6] Dunn M., Wigert I., Wenger A. & Metzger J. (2004) International CIIP Handbook 2004 An Inventory and Analysis of Protection Policies in Fourteen Countries. Center for Security Studies and Conflict Research, Zurich, Switzerland, 403 p. Electronic version: http://www.isn.ethz.ch/crn/_docs/CIIP_Handbook_2004_web.pdf.

[7] Gheorghe A.V. & Vamanu D.V. (2004) Complexity induced vulnerability. Int. J. Critical Infrastructures 1, pp. 76–85.

[8] Schaffert S., Gruber A. & Westenthaler R. (2004) A semantic wiki for collaborative knowledge formation. In: 13th World Wide Web Conference, May 17-22, New York, US. Electronic version: http://www.salzburgresearch.at/research/gfx/SemWikiForCollKnowForm_20060120.pdf.

[9] Shirky C. (accessed 24.4.2006), The semantic web, syllogism, and worldview. URL: http://www.shirky.com/writings/semantic_syllogism.html.

[10] IWCIP2005 (accessed 24.4.2006), Call for papers. URL: http://www.iwcip.org/2005/CfP_WS2005.html.

[11] CERT/CC (accessed 24.4.2006), CERT/CC statistics 1988-2006. URL: http://www.cert.org/stats/cert_stats.html.

[12] Masera M. & Wilikens M. (2001) Interdependencies with the information infrastructure: Dependability and complexity issues. In: 5th International Conference on Technology, Policy, and Innovation, 26-29 June, Ispra, Italy. Electronic version: http://www.delft2001.tudelft.nl/paper%20files/paper1168.doc.

[13] Rinaldi S.M., Peerenboom J.P. & Kelly T.K. (2001) Identifying, understanding, and analyzing critical infrastructure interdependencies. IEEE Control Systems Magazine 21, pp. 111–25.

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

[15] Mock R. & Corvo M. (2005) Risk analysis of information systems by event process chains. Int. J. Critical Infrastructures 1, pp. 247–257.

[16] Engelbart D.C. (1962) Augmenting human intellect: A conceptual framework. Summary Report AFOSR-3223 - Contract AF49(638)-1024, SRI Project 3578, Air Force Office of Scientific Research, Stanford Research Institute, Menlo Park, US. Electronic version: http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html.

[17] Berners-Lee T., Hendler J. & Lassila O. (2001) The Semantic Web. Scientific American 284. Electronic version: http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2.

[18] Völkel M., Krötzsch M., Vrandecic D., Haller H. & Studer R. (2006) Semantic wikipedia. In: Proceedings of the 15th international conference on World Wide Web, WWW 2006, May 23-26, Edinburgh, Scotland. Electronic version: http://www.aifb.uni-karlsruhe.de/WBS/hha/papers/SemanticWikipedia.pdf.

[19] Aumueller D. (2005) SHAWN: Structure Helps a Wiki Navigate. In: W. Mueller & R. Schenkel (eds.) Proceedings of the BTW-Workshop "WebDB Meets IR", March 1, Karlsruhe, Germany. Electronic version: http://dbs.uni-leipzig.de/~david/2005/aumueller05shawn.pdf.

[20] Manola F. & Miller E. (accessed 24.4.2006), Resource Description Framework (RDF) primer. URL: http://www.w3.org/TR/rdf-primer/.

[21] Stoneburner G., Goguen A. & Feringa A. (2002) Risk Management Guide for Information Technology Systems. National Institute of Standards and Technology. Electronic version: http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

[22] PITAC (2005) Cyber Security: A Crisis of Priorisation. National Coordination Office for Information Technology Research and Development, Arlington, US, 58 p. Electronic version: http://www.nitrd.gov/pitac/reports/20050301_cybersecurity/cybersecurity.pdf.

[23] PCCIP (1997) Critical Foundations: Protecting America's Infrastructures. President's Commission on Critical Infrastructure Protection, Washington DC, US., 192 p. Electronic version: http://www.cshs-us.

org/cshs/cshs.nsf/6d0a0b623aab586b85256af500674bbb/e52acb6daaa5d2d785256d340068365b/$FILE/PCCIPreport.pdf.

[24] CVE (accessed 24.4.2006), CVE abstraction content decisions: Rationale and application. URL: http://www.cve.mitre.org/cve/cd_rationale_application.html#id_bugs.

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

[26] CERT/CC (accessed 24.4.2006), CERT/CC overview: Incident and vulnerability trends. URL: http://www.cert.org/present/cert-overview-trends/module-2.pdf.

[27] Röning J. & Eronen J. (2002) Software considered harmful: Why software is insecure. In: Corporate Security, May 29, Helsinki, Finland. URL: http://www.ee.oulu.fi/research/ouspg/protos/sota/CorpSec2002/.

[28] Kaksonen R., Laakso M. & Takanen A. (accessed 24.4.2006), Vulnerability analysis of software through syntax testing. URL: http://www.ee.oulu.fi/research/ouspg/protos/analysis/WP2000-robustness.

[29] CERT/CC (accessed June 24.4.2006), CERT advisory CA-2002-03 multiple vulnerabilities in many implementations of the Simple Network Management Protocol (SNMP). URL: http://www.cert.org/advisories/CA-2002-03.html.

[30] NISCC (accessed 24.4.2006), NISCC vulnerability advisory 006489/H323 vulnerability issues in implementations of the H.323 protocol. URL: http://www.niscc.gov.uk/niscc/docs/re-20040113-00387.pdf.

[31] Clarke R. (accessed 24.4.2006), Looking at vulnerability issues in addressing cyber security. URL: http://www.ncs.gov/nstac/march2002/nsatc_cybersecurity.html.

[32] Harrop M. (2002) Creating trust in critical network infrastructures: Canadian case study. In: ITU Workshop on Creating Trust in Critical Network Infrastructures, 20-22 May, Seoul, Republic of Korea. Electronic version: http://www.itu.int/osg/spu/ni/security/docs/cni.07.doc.

[33] OUSPG (accessed 24.4.2006), PROTOS test-suite: c07-h2250v4. URL: http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/h2250v4/.

[34] OWL Web Ontology Language Overview. (accessed 24.4.2006) URL: http://www.w3.org/TR/owl-features/.

[35] Berners-Lee T. (accessed 24.4.2006), An RDF language for the Semantic Web - Notation 3. URL: http://www.w3.org/DesignIssues/Notation3.html.

[36] Cunningham W. (accessed 24.4.2006), Wiki design principles. URL: http://c2.com/cgi/wiki?WikiDesignPrinciples.

[37] Völkel M., Kiesel M., Schaffert S., Decker B. & Oren E. (accessed 24.4.2006), Semantic wiki state of the art paper. URL: http://wiki.ontoworld.org/index.php/Semantic_Wiki_State_of_The_Art_Paper.

[38] Völkel M. (accessed 24.4.2006), Personal knowledge management with semantic wikis. URL: http://www.xam.de/2005/12_voelkel_oren_SPKM_submission_eswc2006.pdf.

[39] Do the simplest thing that could possibly work. (accessed 24.4.2006) URL: http://c2.com/cgi/wiki?DoTheSimplestThingThatCouldPossiblyWork.

[40] Rocha L.M. & Bollen J. (2001) Biologically motivated distributed designs for adaptive knowledge management. In: Segel L. & Cohen I. (eds) Design Principles for the Immune System and other Distributed Autonomous Systems. Santa Fe Institute Series in the Sciences of Complexity. Oxford University Press, Oxford, UK, pp. 305–334.

[41] Shirky C. (accessed 24.4.2006), Ontology is overrated: Categories, links, and tags. URL: http://www.shirky.com/writings/ontology_overrated.html.

[42] Aumueller D. & Auer S. (2005) Towards a Semantic Wiki Experience: Desktop Integration and Interactivity in WikiSAR. In: 1st Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure, November 6th, Galway, Ireland. Electronic version: http://www.semanticdesktop.org/SemanticDesktopWS2005/final/22_aumueller_semanticwikiexperience_final.pdf.

[43] Diestel R. (2000) Graph Theory. Springer-Verlag, New York, US, 322 p.

[44] Sowa J.F. (1984) Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, 481 p.

[45] de Nooy W., Mrvar A., Batelgelj V. & Granovetter M. (2005) Exploratory Social Network Analysis with Pajek. Cambridge University Press, Cambridge, UK, 362 p.

[46] O'Leary D. (1998) Using AI in knowledge management: Knowledge bases and ontologies. IEEE Intelligent Systems 13, pp. 34–39.

[47] Mcguffin M.J. & Schraefel M.C. (2004) A comparison of hyperstructures: Zzstructures, mSpaces, and Polyarchies. In: 15th ACM conference on Hypertext and hypermedia, August 9-13, Santa Cruz, US, ACM Press, pp. 153–162. Electronic version: http://portal.acm.org/citation.cfm?id=1012852.

[48] Mann D.E. & Christey S.M. (accessed 24.4.2006.), Towards a common enumeration of vulnerabilities. URL: http://www.cve.mitre.org/docs/cerias.html.

[49] Banahan M. (accessed 24.4.2006), Impressions of using WAP/WML. URL: http://ebusiness.gbdirect.co.uk/case_studies/wapimpressions.html.

[50] Kenttälä J. (2005) Exploiting Communication Patterns in Complex Information Networks. Master's thesis, Department of Information Processing Science at University of Oulu, Oulu. Electronic version: http://www.ee.oulu.fi/research/ouspg/frontier/.

[51] Wihersaari J. (2000) Julkisiin lähteisiin perustuva tiedustelu (Open Source Intelligence, OSINT). In: Saarelainen J., Tynkkynen V., Aherto J., Hyytiäinen M. & Metteri J. (eds) Johtamissodankäynti. Taktiikan laitos, Helsinki, pp. 238–254.

[52] Mercado S. (2005) Reexamining the distinction between open information and secrets. Studies of Intelligence 49. Electronic version: http://www.cia.gov/csi/studies/Vol49no2/reexamining_the_distinction_3.htm.

[53] Mercado S. (2004) A venerable source in a new era: Sailing the sea of osint in the information age. Studies of Intelligence 48. Electronic version: http://www.cia.gov/csi/studies/vol48no3/article05.html.

[54] Stevens W.R. (2000) TCP/IP Illustrated, Volume 1, The Protocols. Addison-Wesley, 600 p.

[55] Hirsijärvi S. & Hurme H. (2000) Tutkimushaastattelu: teemahaastattelun teoria ja käytäntö. Helsinki University Press, Helsinki, 213 p.

[56] Berners-Lee T. (2004) WWW2004 Keynote speech. In: 13th World Wide Web Conference, May 17-22, New York, US. URL: http://www.w3.org/2004/Talks/0519-tbl-keynote/.

[57] Shadbolt N., Gibbins N., Glaser H., Harris S. & Schraefel M.C. (2004) CS AKTive Space, or how we learned to stop worrying and love the semantic Web. IEEE Intelligent Systems 19, pp. 41–47. Electronic version: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1315540.

[58] Schraefel M.C., Smith D.A., Owens A., Russell A., Harris C. & Wilson M. (2005) The evolving mSpace platform: leveraging the semantic web on the trail of the memex. In: 16th ACM conference on Hypertext and hypermedia, September 6-9,

Salzburg, Austria, ACM Press, New York, US, pp. 174–183. Electronic version: http://dx.doi.org/10.1145/1083356.1083391.

[59] Decker B., Ras E., Rech J., Klein B. & Höcht C. (2005) Self-organized reuse of software engineering knowledge supported by semantic wikis. In: Workshop on Semantic Web Enabled Software Engineering (SWESE), at the 4th International Semantic Web Conference (ISWC 2005), November 6, Galway, Ireland. Electronic version: http://www.salzburgresearch.at/research/gfx/SemWikiForCollKnowForm_20060120.pdf.

[60] Noel S., Jajodia S., O'Berry B. & Jacobs M. (2003) Efficient minimum-cost network hardening via exploit dependency graphs. In: 19th Annual Computer Security Applications Conference (ACSAC '03), December 8-12, Las Vegas, US, IEEE Computer Society, Washington, DC, US. Electronic version: http://www.acsa-admin.org/2003/papers/98.pdf.

[61] Govindavajhala S. & Appel A.W. (accessed 24.4.2006), Windows access control demystified. URL: http://www.cs.princeton.edu/~sudhakar/papers/winval.pdf.

[62] Sommerville I. (2001) Software Engineering. Pearson Education, 693 p.

[63] Pressman R.S. (1997) Software Engineering: A Practitioner's Approach. McGraw-Hill, 885 p.

[64] Brooks F.P. (1995) The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 322 p.

[65] Muljadi H., Takeda H., Kawamoto S., Kobayashi S., Mizuta Y., Demiya S.M., Suzuki S., Kitamoto A., Fujiyama A., Araki J., Shirai Y., Ichiyoshi N., Ito T., Abe T., Gojobori T., Sugawara H. & Miyazaki S. (2005) Semantic mediawiki: a user-oriented system for integrated content and metadata management system. In: IADIS International Conference WWW/Internet 2005, Oct 19-22, Lisbon, Spain. Electronic version: http://www-kasm.nii.ac.jp/papers/takeda/05/hendry05icwi.pdf.

[66] Hepp M., Bachlechner D. & Siorpaes K. (2005) Ontowiki: Community-driven ontology engineering and ontology usage based on wikis. In: 2005 International Symposium on Wikis (WikiSym 2005), Oct 16-18, San Diego, US. Electronic version: http://www.heppnetz.de/files/ontowikiDemo-short-camera-ready.pdf.

[67] Norvig P. (1992) Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp. Morgan Kaufmann Publishers, 946 p.

[68] Kreyszig E. (1993) Advanced Engineering Mathematics. John Wiley & Sons, 1271 p.