Defence Research and Development Canada
Recherche et développement pour la défense Canada

Network defensive posture demonstrator

System description and work plan

Glen Henderson, Eugen Bacic and Larry Tremblay

Defence R&D Canada – Ottawa

Canada

Contract Report
DRDC Ottawa CR 2009-133
August 2009

Network defensive posture demonstrator

System description and work plan

Glen Henderson
Bell Canada

Eugen Bacic
Bell Canada

Larry Tremblay
Bell Canada

Prepared by:

Bell Canada
333 Preston St., Suite 1100, Ottawa, Ontario, K1S 5N4

Project Manager: Joanne Treurniet
Contract Number: W7714-071028
Contract Scientific Authority: Joanne Treurniet

Defence R&D Canada – Ottawa

Contract Report

DRDC Ottawa CR 2009-133

August 2009

Scientific Authority

Original signed by Joanne Treurniet

Joanne Treurniet

Approved by

Original signed by J. Lefebvre

J. Lefebvre
Head/NIO Section

Approved for release by

Original signed by B. Eatock

B. Eatock
Head/Document Review Panel

Abstract

The Network Information Operations (NIO) section at DRDC Ottawa is performing work under the Dynamic Computer Network Defence (CND) Applied Research Project, the goal of which is to provide network operators with situational awareness of their network. Crucial to this awareness is the knowledge of what assets residing on their network are critical to their operations, and what assets on their networks are exposed, that is, have a vulnerability that would allow an adversary to violate the confidentiality, integrity or availability of the asset. The NIO section has defined the combination of these two elements as network defensive posture: the set of exposed, critical resources on the network. Further, the defensive posture of a network is dynamic. The network critical resources may change with time in response to changing missions and operational priorities. At the same time, the network state can be altered by new software installations, the discovery of new vulnerabilities in existing software, changes to firewall rules, and other network events. Both types of changes affect the defensive posture.

A milestone in the Dynamic CND project is to create a demonstrator of a network defensive posture system. This contract addresses this requirement, and will provide an environment in which researchers at DRDC Ottawa can continue their work. Through previous investigation, we have determined that the MulVAL open-source software package is appropriate for our needs, and as such will be used in the development of the defensive posture analysis system. MulVAL is a logic-based network security analyzer powered by the XSB Datalog interpreter. It is capable of performing a multi-stage, multi-host vulnerability assessment of a computer network, determining all potential attack paths in a given network.

Résumé

La section des Opérations d'information de réseau (OIR) de RDDC Ottawa effectue des travaux dans le cadre du projet de recherche appliquée Défense dynamique des réseaux informatiques (DDRI), dont l'objectif est de permettre aux opérateurs de réseaux de connaître la situation dans leurs réseaux. Ce qu'il faut absolument savoir pour connaître l'état d'un réseau : les biens qui sont installés dans le réseau ou qui y sont reliés et qui sont critiques pour les opérations, ainsi que les biens du réseau qui sont exposés, c'est à dire, les biens ayant une vulnérabilité qui pourrait permettre à un adversaire de violer la confidentialité, l'intégrité ou la disponibilité du bien. La section des OIR a défini la combinaison de ces deux éléments comme étant la position défensive d'un réseau : l'ensemble de ressources exposées et critiques dans le réseau. De plus, la position défensive d'un réseau est dynamique. Les ressources critiques d'un réseau peuvent changer avec le temps, en fonction des missions et des priorités opérationnelles changeantes. Parallèlement, l'état d'un réseau peut être modifié par l'installation de nouveaux logiciels, la découverte de nouvelles vulnérabilités dans des logiciels existants, des changements dans les règles des pare-feu et

autres événements relatifs au réseau. Les deux types de changements touchent la position défensive.

Un jalon important dans le projet de DDRI est de créer un démonstrateur d'un système de position défensive d'un réseau. Avec ce contrat, on vise à répondre à cette exigence et à créer un environnement dans lequel les chercheurs de RDDC Ottawa peuvent continuer leurs travaux. Des enquêtes antérieures nous ont permis de déterminer que le logiciel à source ouverte MulVAL répond à nos besoins et, par conséquent, il sera utilisé dans l'élaboration du système d'analyse de la position défensive. MulVAL est un analyseur de sécurité de réseau à base logique qui fonctionne grâce à l'interpréteur XSB Datalog. Il est capable d'effectuer une analyse des vulnérabilités à étapes et hôtes multiples d'un réseau informatique, afin de déterminer tous les chemins d'attaques potentiels d'un réseau donné.

Executive summary

Network defensive posture demonstrator

Glen Henderson, Eugen Bacic, Larry Tremblay; DRDC Ottawa CR 2009-133; Defence R&D Canada – Ottawa; August 2009.

Background: This document describes the vision and architecture of an application to analyse network defensive posture. The application's function is comprised of several stages. To begin, the user defines and manipulates a network model using a common descriptor language. This model contains all network elements and contains information describing each element's configuration, such as operating system, software, patch level, services available, connections to other elements, and so on. Once defined, the application will validate that the network definition is correct and well formed. Next, the defensive posture analysis of the network is performed, including preprocessing, model analysis (via MulVAL), and prioritization (via AssetRank or other scheme). When analysis is complete, the application displays the results of the analysis in a way that allows the user to alter the view of the results to highlight whichever elements of the analysis are most relevant. With this, users may use the analysis as a diagnostic tool for a network, as well as using the application as a "what-if" tool for iterative changes to the network without changing the network itself.

Principal results: This document presents an architecture for the application which may be used to form the basis of a system design for implementation of the application. Included in this document are:

1. The results of interviews with DRDC staff to establish the requirements for the system;
2. Discussion of XML schema for describing networks, and a decision as to which schema to use;
3. A high level system architecture describing the system, its components, and their interaction;
4. High level UML diagrams of the system;
5. Selection of software development language and any libraries to be used for development;
6. Descriptions of system component application programming interfaces (APIs) used for interactions between the components;
7. Mockups of the application user interfaces; and

8. A proposed development work plan.

Future work: As a follow-on to this task authorization, design and development of the system is proposed to show the proof-of-concept. Assuming the concept is proven, future steps include design and implementation of a full-fledged application.

Sommaire

Network defensive posture demonstrator

Glen Henderson, Eugen Bacic, Larry Tremblay ; DRDC Ottawa CR 2009-133 ;
R & D pour la défense Canada – Ottawa ; août 2009.

Renseignements généraux : Ce document décrit la vision et l'architecture d'une application servant à analyser la position défensive d'un réseau. La fonction de l'application compte plusieurs étapes. Pour commencer, l'utilisateur définit et manipule un modèle de réseau à l'aide d'un langage descripteur commun. Ce modèle contient tous les éléments du réseau et de l'information décrivant la configuration de chaque élément, comme le système d'exploitation, les logiciels, les correctifs installés, les services disponibles, les connexions à d'autres éléments, etc. Une fois ce modèle défini, l'application valide la définition du réseau pour s'assurer qu'elle est exacte et bien formée. Ensuite, le système analyse la position défensive du réseau, y compris le prétraitement, l'analyse du modèle (avec MulVAL) et l'établissement de l'ordre de priorité (à l'aide d'AssetRank ou d'un autre système). Une fois l'analyse terminée, l'application affiche les résultats de l'analyse d'une façon qui permet à l'utilisateur de modifier la vue des résultats pour mettre en évidence les éléments de l'analyse qui sont les plus pertinents. Ainsi, l'utilisateur peut utiliser l'analyse comme outil diagnostic pour gérer un réseau ou utiliser l'application comme un outil permettant de simuler des changements itératifs au réseau sans modifier le réseau proprement dit.

Résultats principaux : Ce document présente une architecture pour l'application, qui pourrait être utilisée afin de constituer la base d'une conception de système en vue de la mise en œuvre de l'application. Le document comprend :

1. les résultats des entrevues avec le personnel de RDDC afin d'établir les exigences du système ;
2. une discussion du schéma XML permettant de décrire les réseaux et une décision quant au schéma à utiliser ;
3. une architecture de système de haut niveau décrivant le système, ses composantes et leur interaction ;
4. des diagrammes UML de haut niveau du système ;
5. la sélection du langage de développement de logiciels et des bibliothèques à utiliser pour le développement ;
6. des descriptions d'interfaces API ("application programming interface") de composantes du système utilisées pour les interactions entre les composantes ;
7. des modèles des interfaces utilisateur de l'application ;
8. un plan de travail de développement proposé.

Recherches futures : Pour faire suite à l'autorisation de cette tâche, on propose de concevoir et de développer le système pour en valider le principe. Si le principe du système est validé, les étapes futures comprennent la conception et la mise en œuvre d'une application complète.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

The Network Information Operations (NIO) section at DRDC Ottawa is performing work under the Dynamic Computer Network Defence (CND) Applied Research Project, the goal of which is to provide network operators with situational awareness of their network. Crucial to this awareness is the knowledge of what assets residing on their network are critical to their operations, and what assets on their networks are exposed - i.e. have a vulnerability that would allow an adversary to violate the confidentiality, integrity or availability of the asset. The NIO section has defined the combination of these two elements as network defensive posture: the set of exposed, critical resources on the network. Further, the defensive posture of a network is dynamic. The network critical resources may change with time in response to changing missions and operational priorities. At the same time, the network state can be altered by new software installations, the discovery of new vulnerabilities in existing software, changes to firewall rules, and other network events. Both types of changes affect the defensive posture.

A milestone in the Dynamic CND project is to create a demonstrator of a network defensive posture system. This contract addresses this requirement, and will provide an environment in which researchers at DRDC Ottawa can continue their work. Through previous investigation, we have determined that the MulVAL open-source software package is appropriate for our needs, and as such will be used in the development of the defensive posture analysis system. MulVAL is a logic-based network security analyzer powered by the XSB Datalog interpreter. It is capable of performing a multi-stage, multi-host vulnerability assessment of a computer network, determining all potential attack paths in a given network.

2 Requirements Gathering

As part of the effort to define the requirements for the network defensive posture demonstrator, information gathering interviews were held with 6 DRDC stakeholders and potential system users. After consideration and consultation with the project authority, the desired design and functionality requests were reduced to a working set of requirements that will define the solution capabilities.

Each of these requirements was given a rank on a priority scale that indicates the relative necessity of each requirement in the system design and architecture. The scale is described as follows:

Rank	Description
1	The requirement is an absolute necessity for the solution and **must** be reflected in the architecture and design
2	The requirement reflects a significant functional aspect of the solution and **should** be included in the design unless there is a clear rationale for not including the capability represented by the requirement.
3	The requirement reflects functionality that **is desired** for the solution where the design, implementation and maintenance of the capability does not represent an effort that would be detrimental to other more important requirements.
4	The requirement reflects functionality that is desired for later versions of the solution and, as such, the design **should not preclude** support for the functionality represented by the stated requirement.
5	The requirement reflects potential long term capability for the solution and **should be considered** during the initial design and architecture unless there is a rationale for omitting the stated functionality.

Table 1: Requirements Ranking

Note that requirements without rankings represent requirements that were collected from interview personnel but have been dropped from the solution design in subsequent discussions with the project authority. Individual requirements have been eliminated from the solution design because:

- the perceived benefit was negligible,
- the requirement was out of scope for the project, or
- the requirement was not in line with the project authority's research priorities.

These eliminated requirements have been included in this document for completeness.

2.1 General Requirements

These requirements describe needed functionality that reflects the overall intent of the solution and goals to be achieved in the architectural design of the application.

No.	Description	Priority
GR-1	The system must be modular and extensible.	1
GR-2	The individual solution modules must be uniquely identifiable for configuration purposes.	1
GR-3	The modules must *plug* into a dynamic and configurable framework to meet the needs of researchers and end-users alike.	1
GR-4	Definitions for a given "type" of module must provide for the means of swapping one module out for another. For example, replacing "AssetRank" with another asset ranking module.	1
GR-5	Standards based formats such as XML must be used and they must be extensible with an open and well-defined format.	1
GR-6	All assets being examined must be stored in a Microsoft SQL Server database with an open XML interface whose output is human-readable and manipulable. The structures must be sufficiently dynamic to allow for inclusion of new asset values and variables.	1
GR-7	For the purposes of generating models, the system must allow assets (physical, logical) to be evaluated individually or as groups. That is, is should be possible to cut-and-paste or template systems and groups of systems.	1
GR-8	The system must allow users to observe the state of the analysis as it unfolds, such as profiling and breakpointing. Breakpoints should be able to be set at the module-level.	1
GR-9	The system must allow for flexible grouping of assets according to predefined /*values*, such as time, location, and type. The XML schema must support this capability.	1
GR-10	The system must allow for the prioritization of assets.	1

No.	Description	Priority
GR-11	The system must allow for the creation of “templates and baselines” so as to allow for rapid creation of massive networks from basic components.	1
GR-12	Users must be able to specify needed details associated with an attack, specifically the attacker’s node location and initial access privileges.	1
GR-13	The system should attempt to re-use existing technologies so as to minimize the development effort.	2
GR-14	The system should be as simple as possible while still providing the needed analytical capabilities in support of security research.	2
GR-15	The system, though composed of multiple independent modules, must behave and be called like a single application. An approach that uses multiple executables, for example one for loading/manipulating network models and one that integrates with a solution such as RapidMiner would be acceptable.	2
GR-16	Modules should be able to be authored in any language so long as they link together to the larger system whole. The primary purpose of this demonstrator is to easily enable further research extensions, therefore, modules written under this contract should be written in Python, Java, C++, Visual Basic and Datalog as these represent the languages used by DRDC research staff.	2
GR-17	Several robust and documented APIs should be in place to support modularity and external connections	2
GR-18	Well-documented and simple APIs should be provided that are compatible with in-use technologies such as C/C++, Python, Java, MatLab, and Prolog.	2
GR-19	The system should allow multiple analysis modes. For example, the user community may define the need for such analysis modules as sensitivity analysis, trajectory analysis, and what-if analysis.	2
GR-20	The client should be a desktop application rather than a web-only application.	2
GR-21	The system should support result filtering (e.g. only show attack graphs that are affecting Internet Explorer) for ease of use and to support need-to-know.	2
GR-22	Within the model, it should be possible to define an asset as anything in the network environment, including files, devices, machines, and network nodes.	2

No.	Description	Priority
GR-23	The system should provide details regarding a vulnerability once prioritized, for example, what is the vulnerability and what remedial measures are appropriate to counter it.	2
GR-24	Virtual machines may be used for snapshot ability and transportability of the system. As such, the solution should be delivered and installable as a virtual machine.	2
GR-25	The system must be able to take a predefined network definition and generate an equivalent network model.	2
GR-26	The topologies must be extendable and initially annotated to the degree required by the algorithms with information on assets, safeguards, connectivity, addresses, applications, etc.	2
GR-27	The system must support import of data from multiple external sources, reusing rather than creating entirely new data sources such as JNDMS, IDS, BPEL, and NVD.	3
GR-28	External applications should be able to use the output from the system, that is, output should be in XML where users can transform/analyze it independently.	3
GR-29	The system may be aware of business processes as policies to be enforced and use them in the context of prioritization.	3
GR-30	These components must be stored within a database for later reuse, extension, or instantiation.	3
GR-31	The solution should support release as open source software by both following and restating open source licensing restrictions.	3
GR-32	The system must be able to run in a distributed fashion, wherein the modules can operate within a single computer system or across a network.	4
GR-33	The system should provide extended details regarding a vulnerability once prioritized and reported, for example, how is the vulnerability exposed and what policy violations apply that lead to the vulnerability.	4
GR-34	The system should be able to prioritize and correlate vulnerability/event data (i.e. examine events, extract attributes, correlate to vulnerabilities, prioritize the vulnerabilities). As this system is pro-active only, this level of vulnerability analysis is out of scope for this project.	5
GR-35	The system should be able to auto-discover and create the necessary topology models.	5
GR-36	The system must have an algorithm that defines the coverage provided by existing safeguards, preferably with a ranking of effectiveness.	5

No.	Description	Priority
GR-37	The system must fit into the JNDMS architecture.	
GR-38	The system and the XML structures and formats must comprehend the passage of time, allowing for temporal logic to be embedded into the asset description, that is work factor, as well as the various structures that define how the system functions.	

Table 2: General Requirements

2.2 Operational Requirements

These requirements describe elements of the application pertaining to the operational capabilities that will be available to the analyst.

No.	Description	Priority
OR-1	A robust XML schema for defining network and node elements must be found or created.	1
OR-2	The system must create detailed logs of the analysis steps and status (metadata) to facilitate correctness verification of models.	1
OR-3	The system should allow users to replay models, including what-if scenarios, to ensure same results for same inputs.	1
OR-4	It must be possible to replace network models and analysis modules in support of analysis.	1
OR-5	The system should allow the end-user to indicate which variables are considered important, especially for data import from external sources or for output and pre-filtering.	1
OR-6	The system must allow for the creation of multihop analysis.	1
OR-7	The system must allow for the data to be provided in an open format (XML) so as to allow for end-user unique ways of accessing, assessing, and navigating the results.	1
OR-8	The system must be able to perform internal logging of all relevant actions performed by the end-user.	2
OR-9	The system must allow invocation from external applications for data exchange and manipulation.	2
OR-10	The system should allow for algorithm unfolding traceability that allows the user to watch how an algorithm works against a given set of data. The system should allow for full inspection of the data structures pre/post algorithm execution, for example, to support debugging efforts.	2

OR-11	The system should be able to indicate whether or not an asset is accessible from a given node and the facts which enable the access to occur.	2
OR-12	Exposures (or risks) must be highlighted as to where they originate and how they impact the network model.	2
OR-13	The system should be available across various platforms (PC, Mac, Unix) where there is support for the tools that will define the solution components such as RapidMiner.	3
OR-14	The system must maximize the use of the mouse, for example, the ability to right click to bring up contextual menus.	3
OR-15	The architecture must allow for efficient comparison with known facts. i.e.: if the system states that a machine is not vulnerable, but it is known to be vulnerable, that should be easily compared.	3
OR-16	The system should be able to provide Course of Action (COA) results whenever it discovers a threat.	3
OR-17	Reports must be creatable, editable, and exist that sufficiently reduce the amount of information flowing back to the operational end-user (i.e., there is simply too much information, it needs to be reduced into either digestible chunks or distilled down to relevant data only)	3
OR-18	The system must allow invocation of external applications for data exchange and manipulation.	4
OR-19	The system should allow choice as to its scan range/depth (e.g. backbone, WAN, Internet exposed systems, servers, workstations)	4
OR-20	There should be a way of linking the results of this system back into JNDMS.	4
OR-21	The system must be able to perform discovery (topology, nodes, content) of a network to which it is introduced.	5
OR-22	The system may be capable of performing diagnostics (e.g. detect misconfiguration in firewall, detect deviance in router configuration from NSA hardening guidelines)	5
OR-23	The system may be capable of observing historical inputs to make better judgements regarding events. For example. 29 of 30 of the same event in the past four hours were false positives, then lower the priority of the 31st event unless some other parameter is also changed.	
OR-24	The system should allow for data and reports to be dumped to a web page.	
OR-25	The system must be able to determine a threat against an asset and provide the likelihood.	

OR-26	Detected events must be ranked as to relevance and reliability of event based on known safeguards, etc. within the system being modeled.	
OR-27	False positive and false negative issues should be addressed in a logical and consistent manner.	
OR-28	Discovered dangers/threats should result in programmable measures that are passive (email the security officer) or active (shutdown systems).	
OR-29	Correlation with historic precedence must be maintained so as to retain long-term state and leverage temporal logic.	

Table 3: Operational Requirements

2.3 Configuration Requirements

These requirements describe elements of the application that pertain to configuration. This includes configuration of the application itself, as well as definition of network models in the chosen descriptor language.

No.	Description	Priority
CR-1	System configuration files must be human readable and modifiable.	1
CR-2	Configuration files must allow for configuration of system, indicating which modules to load and in which order including pre- and post-processing.	1
CR-3	Changes made via GUI must be reflected in configuration files.	1
CR-4	The system should have facilities to make model creation fast and simple (e.g. templating of nodes, cut and paste, etc.)	1
CR-5	The system should reflect changes made to the model via the GUI in configuration files as the changes are made. There should be file consistency checks on save. That is, if the original configuration file has been changed the user should be warned before results are saved. We do not need try to merge the configuration file changes with the online changes, just identify when to potential source of change (GUI and config file) may impact each other.	2
CR-6	Changes made to configuration files should be detected in an active session.	2

CR-7	Reporting formats should be defined in configuration files that may be copied and modified.	3
CR-8	The system may allow diff type comparisons on model files and configuration files to provide ease of comparison to see what is different between scenarios.	3

Table 4: Configuration Requirements

2.4 User Interface Requirements

These requirements describe elements of the application that pertain to presentation. This includes what is displayed to the user, as well as how the user interacts with the application.

No.	Description	Priority
IR-1	The GUI presented to the user must be clean and manipulable, with the ability to drill down into detail for individual elements.	1
IR-2	User interface must allow outputs to show results in multiple useful formats (tables, graphs, charts, etc.).	1
IR-3	The GUI must be able to provide analysts with a user-friendly interaction method.	1
IR-4	The system must provide text-based and graphical interfaces so as to allow calls from scripts that allow it to be an element in a complete simulation.	2
IR-5	The system should allow facilities for model creation via GUI in addition to configuration files.	2
IR-6	The system should allow access (read, modify, save, load) to AssetRank parameters.	2
IR-7	The system should allow access (read, modify, save, load) to MulVAL predicates.	2
IR-8	The system may allow the user to change the model, that is, by halting processing at a module level and allowing the manipulation of the execution parameters of subsequent modules.	2
IR-9	The system should allow, for appropriate parameters, the iteration over a range of values.	2
IR-10	Text descriptions, similar to Visio, along graph edges should be supported.	3

IR-11	User interface should allow outputs to show results in multiple contextual formats, including network view, asset view, and operations view..	4

Table 5: Interface Requirements

2.5 Persistence Requirements

These requirements describe elements of the application that pertain to how data is kept in non-volatile forms (i.e. files written to disk, database, etc.) This includes what is stored, the formats data is stored in, and manipulation of those storage formats.

No.	Description	Priority
PR-1	Data persistence must be provided by MSSQL.	1
PR-2	Interactions with the data service must be via open standards such as XML/XQuery.	2
PR-3	The system should allow saving model states: resume where you left off, reload an earlier version, define a common start point for what-if scenario modelling, etc.	2
PR-4	The system should allow data export in multiple formats (e.g. CSV, XML, SQL).	2

Table 6: Persistence Requirements

2.6 Documentation Requirements

These requirements describe elements of the application's documentation. This includes architecture and design documentation, installation and maintenance documentation, test data, test guides, and user guides.

No.	Description	Priority
DR-1	The system must be delivered with sufficient documentation to support the installation, operation and maintenance of the solution post-delivery.	1

DR-2	Documentation must make integration by external parties as simple as possible.	2

Table 7: Documentation Requirements

2.7 Performance Goals

During the interview and project discussions, several design goals pertaining to NDPD solution performance were recognized. While not formal solution requirements, these goals will aid in the definition of an NDPD solution that is acceptable from a performance perspective. That is, these guidelines will define what is deemed acceptable performance and provide suggestion on how this performance can be achieved.

- Using the system as a means to prioritise allocation of IT resources on a day-to-day basis, “fast enough” may mean an hour or less to evaluate a network for vulnerabilities
- Using the system as a what-if simulator, or as an element in a larger simulation exercise, “fast enough” may mean within 24 hours
- The system should be able to operate on a laptop in a “lite” version of the system The system should be “lite enough” to run on a single machine (inside a single VM) but otherwise, there are no performance/size requirements
- During the design and implementation phases, performance optimization techniques that leverage memory-based objects over data service objects should be utilized where appropriate.

3 Network Description XML Schema

A requirement is that network models be stored in a human readable and modifiable format. To this end, an XML based description language to define models been given as a requirement of the system. This description language must allow users to describe the elements of a network in a manner that is complete for the purposes of model analysis.

3.1 Network Element Description

DRDC has defined a set of attributes for network elements under analysis that must be supported by the description language used. These attributes cover several areas, including:

- Vulnerability rules
- Exploit rules
- File access
- Trojan horse programs
- NFS semantics
- User credentials
- Network topology
- Policy specification

Within each of these areas, several attribute types are defined that are required for the analysis of the network's defensive posture.

3.2 Description Languages

Several existing XML based languages exist for describing networks. Each of these is limited in its application in that they are each mainly meant to describe network topology, and do not extend to some of the other attributes defined by DRDC. Selection of an existing language is then an exercise in determining which is the best fit that covers the greatest portion of the defined attributes. Once a selection is made, it must either be extended to include all of the defined attributes, or a secondary schema must be defined that is specifically tailored to include the missing attributes.

Existing languages that are most suitable are described in the following sections.

3.2.1 Network Description Language

The Network Description Language (NDL) is developed by the University of Amsterdam. Its stated goal is to provide a way to describe computer networks in a meaningful way. NDL is an ontology for computer networks that is based on the W3C Resource Description Framework (RDF).

3.2.1.1 Overview

NDL is divided into five separate schema for describing different aspects of a network:

- Topology
- Layer
- Capability
- Domain
- Physical

Technology schema are also available as specific implementations of the layer schema that is defined as an abstraction.

3.2.1.2 Advantages

NDL is a small and compact language that is strong for detailing network topologies. A number of tools including validators and generators are available for NDL in several languages. As with any XML-based language, NDL is extensible or will coexist with alternate schema that define other required attributes.

3.2.1.3 Disadvantages

The NDL is designed to describe network topologies and does not include support for any of the other required attributes from section 3.1. Further work would be required to create extensions to NDL to cover required data elements that are not currently included in the specification, or to create an adjunct XML schema to cover those elements. While created and backed by an academic institution, NDL is not embraced to date by any industry players. As such it remains an academic exercise.

3.2.1.4 Summary

Given its narrowness of scope requiring extra work to bring full support of the required attributes, and lack of industry support, NDL is not a strong choice for the demonstrator.

3.2.2 System Description Language and Security Policy Language

The System Description Language (SDL) and Security Policy Language (SPL) are developed by the POSITIF project, which is a standards body funded by the European Commission. Its stated goal is to describe systems to be configured and managed by framework tools. The SDL language is developed to describe networked systems and applications with the degree of detail needed by individual users.

3.2.2.1 Overview

The SDL schema is used for describing different aspects of a network:

- Topology, including physical (nodes) and logical (vlan) elements
- Configuration of nodes
- Security, such as packet filtering, operating system controls, application level access controls

The SPL schema is used for describing different aspects of the security aspects of a network:

- Authentication policies, describing rules which define authentication criteria for a subject with a network element as the target.
- Authorisation policies, describing the rules which define authorisation based on privileges.
- Filtering policies, describing filtering criteria of a network element such as address masks, port ranges, protocol types, etc.
- Channel protection policies, describing security associations for a channel such as IPSec, IKE, SSL, etc.

3.2.2.2 Advantages

SDL+SPL is published by a body backed by the European Commission with support from several industry players within Europe. The defined schema cover several areas of the defined attributes from section 3.1, including topology, policy, and credentials.

3.2.2.3 Disadvantages

The POSITIF website indicates that the effort was to complete on 31 January 2007. At this time, the POSITIF website does not appear to have been updated since late 2006. The SDL+SPL are designed to use a data model based on the Common Information Model (CIM). As such, translation of SDL+SPL into the CIM format is required for the use of tools provided by POSITIF for validation. The schema as defined contain a significant amount of vendor- and technology-specific reference – the schema are intended to be used as a tool for configuration and provisioning of network devices, rather than a tool for modelling networks. Further work would be required to create extensions to SDL+SPL to cover required data elements that are not currently included in the specification, or to create an adjunct XML schema to cover those elements.

3.2.2.4 Summary

The SDL+SPL schema appears as though it may have been abandoned, and as such is not a strong choice for the demonstrator.

3.2.3 Common Information Model

The Common Information Model (CIM) provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. CIM's common definitions enable vendors to exchange semantically rich management information between systems throughout the network.

3.2.3.1 Overview

CIM is divided into sixteen separate schema for describing different aspects of a network:

- Application
- Application - J2EE
- Core
- Databases
- Devices
- Events
- Interoperability
- IPsecPolicy

- Metrics
- Network
- Physical
- Policy
- Security
- Support
- System
- User

3.2.3.2 Advantages

The Distributed Management Task Force is the governing body for CIM. The DMTF is composed of members from hundreds of organisations within industry and academia. CIM is a very robust and extensive specification with definitions for describing topology, devices, policies, file systems, credentials - the bulk of the attributes defined. It is designed to be extensible to service the needs of users, and a stated goal of CIM is for future releases to remain backwards compatible with previous releases.

3.2.3.3 Disadvantages

CIM is a very large and complex specification with definitions for elements that are irrelevant to the needs of the system. As such, a fairly significant learning period would be required to become familiar with CIM to the point of deciding which schema are mandatory, which are optional, and which schema are relevant. Further work would be required to create extensions to CIM to cover required data elements that are not currently included in the specification, or to create an adjunct XML schema to cover those elements.

3.2.3.4 Summary

CIM represents the most comprehensive network description language currently available, covering more of the required attributes than any other schema. In addition to this is a provision for extensibility, allowing coverage of the remaining attributes through extensions to the schema. CIM is a strong choice for the demonstrator.

3.2.4 Nessus XML Schema

The Nessus XML Schema is maintained by Tenable Security. It defines the format of the output generated by the Nessus scanner after analysing a network and the elements it contains.

3.2.4.1 Overview

The Nessus XML schema can describe several different aspects of a network:

- Topology
- Vulnerability
- Service

3.2.4.2 Advantages

Nessus XML is a small and compact schema. As with any XML-based language, it is extensible or will coexist with alternate schema that define other required attributes.

3.2.4.3 Disadvantages

The Nessus XML schema is designed to briefly describe network elements that have been found to have vulnerabilities present during the Nessus scan. It is not designed to be a general and comprehensive schema, and only includes the minimum necessary to describe the results of a scan. Significant work would be required to create extensions to this schema to cover required data elements that are not currently included, or to create an adjunct XML schema to cover those elements.

3.2.4.4 Summary

Given its narrowness of scope requiring extra work to bring full support of the required attributes, the Nessus XML schema is not a strong choice for the demonstrator.

3.3 Conclusion

Of the three XML-based languages for describing networks, none will be able to model all of the required attributes that have been defined. In all cases, some sort of extension to the XML schema will be required, or the creation of a separate XML schema to describe the missing attributes. Network description language is considered the least useful for the demonstrator as it covers the fewest of the required attributes. System Description Language + Security Policy Language cover more than NDL, but still falls well short of the required attribute definitions, and appears to be abandoned. The Nessus XML schema also falls short of the required attribute definitions, although it likely requires less extension than the previous candidates to become useful. CIM is the likely best candidate, even though it will require the most learning of the three before it becomes useful. CIM is also designed to be extensible, allowing the missing attribute definitions to be described in a separate schema that does not affect the base CIM schema.

4 Demonstrator Architecture

This section describes the architecture of the system, the operation of which is described in section 1. This section presents the architecture as a set of views: use-case view, logical view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model.

The demonstrator function is comprised of several stages:

- The user defines and manipulates a network model using a common descriptor language.
- This model contains all network elements and contains information describing each element's configuration, such as operating system, software, patch level, services available, connections to other elements, and so on.
- The application will validate that the network definition is correct and well formed.
- The defensive posture analysis of the network is performed, including preprocessing, model analysis, and prioritization.
- After analysis is complete, the application displays the results of the analysis in a way that allows the user to alter the view of the results to highlight whichever elements of the analysis are most relevant.

With this, users may use the analysis as a diagnostic tool for a network, as well as using the application as a ”what-if” tool for iterative changes to the network without changing the network itself. Several architecture models were considered before settling on the chosen architecture. The system will be split into two components: an application for manipulating network models, and a plugin for the RapidMiner application.

Figure 1 shows an overview of the system architecture. The architecture is described in views to illustrate the operation and composition of the system:

4.1 Architecture Goals and Constraints

During the requirement gathering phase, several constraints and design goals were identified that must be reflected in the architecture. It must be noted that the requirements collected describe the completed system at its end state. The result of this TA is an architecture for a demonstrator of the system. To that end some requirements must be ignored either in the interest of time required to develop the demonstrator, or because the requirement was deemed of lesser priority in the overall system. In general, requirements rated as priority 1 or 2 (see section 2) are to be incorporated in the demonstrator, while requirements

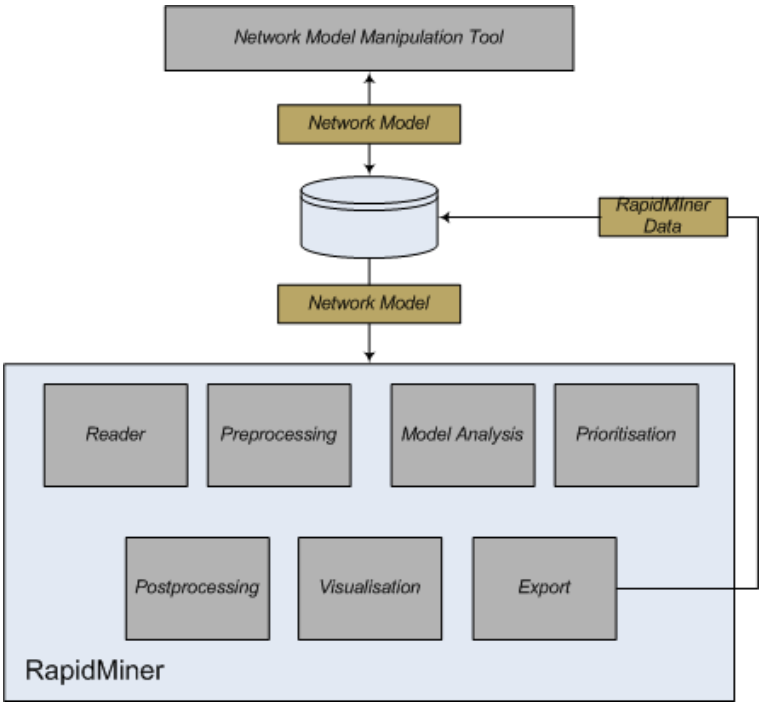

Figure 1: *Architecture Overview*

of priority 3 to 5 are included only if their exclusion would lead to a loss of functionality thought to be central to the application.

The decision to split the demonstrator into two components was made after consideration of requirements with the project authority, most notably the requirement that DRDC staff be able to create new modules to perform analysis. The RapidMiner application provides a framework that is robust and useful, and for this reason was chosen as the framework for the analysis capability.

RapidMiner is an open-source application for data mining and machine learning experiments. Central to RapidMiner is the operator, which is a module that performs a task such as input/output, preprocessing, or visualization. Users create sequences of operators to perform operations on their data sets. Scripts may be made up of an arbitrary number operators. The RapidMiner framework is extensible via the use of a plugin system, whereby custom operators may be created for any purpose to be used in the application. RapidMiner exposes a well-defined application programming interface (API) that allows users to create custom operators for use within the RapidMiner application with a minimal learning curve. Multiple operators may be combined into one container, called a plugin. Once created, a plugin is simply placed into the appropriate directory and RapidMiner loads and makes available all operators in the plugin.

The decision to create a separate application for network model manipulation stems from the decision to not modify the basic RapidMiner application. As delivered, a plugin cannot be used to provide the desired ability to graphically view and manipulate network models. Integration of this functionality into RapidMiner would require modification of its source code. As this would require re-modification of each new release of RapidMiner and runs the risk of a future version being incompatible with the changes, the choice to perform network model manipulation in an external application was made.

4.2 Use-Case View

A description of the use-case view of the software architecture. The use-case View is important input to the selection of the set of scenarios and/or use-cases that are the focus of an iteration. It describes the set of scenarios and/or use-cases that represent some significant, central functionality. It also describes the set of scenarios and/or use-cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The use-cases are:

- View model
- Modify model
- Save model
- Fetch model
- Validate model
- Preprocess model
- Postprocess model
- Model analysis
- Prioritization
- View results
- Export
- Iterate

These use-cases are initiated by the user or by the RapidMiner application. In addition to these use-cases, two wrapping use-cases are defined: manipulate model and model analysis.

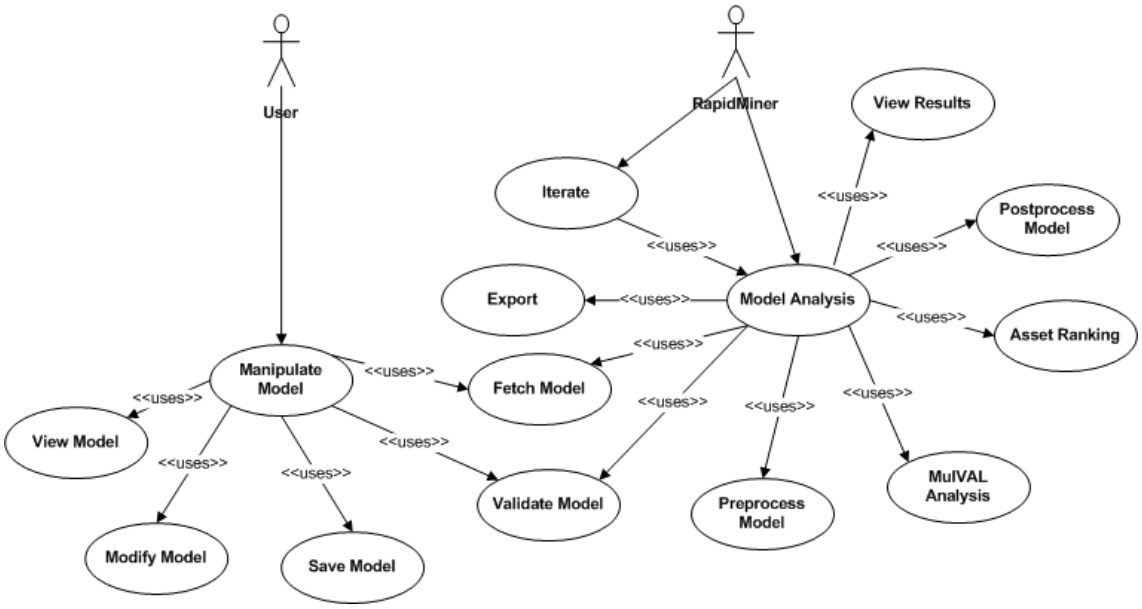

Figure 2: *Significant Use-Cases*

4.2.1 Architecturally Significant Use-Cases

Figure 2 shows the significant use-cases of the demonstrator. In this use-case, both the user and the RapidMiner application are actors using the defined use-cases. The user actor is implied to be a user who is using the model manipulation application, and the RapidMiner actor is implied to be activating use-cases in the execution of an analysis. As illustrated, the wrapper use-cases, Manipulate Model and Model Analysis, indicate the flow of use-case application within the demonstrator.

4.2.1.1 Fetch Model

This use-case retrieves a model from persistent storage. Persistent storage is not limited to the host system, but may be a network-accessible location such as a remote system or SAN. The actors starting this use-case include the user and RapidMiner application.

4.2.1.2 Validate Model

This use-case performs an XML schema validation on a model that has been loaded. This includes verifying that the model is correctly formed, but does not extend to attempting to guess whether the model is correct (e.g. that a particular element is connected to the correct router or switch as in the actual network.) It may detect that an element is orphaned, or that a logical error exists in the model (e.g. a router is connected to itself). The actors starting this use-case include the user and RapidMiner application.

4.2.1.3 View Model

This use-case describes how a model that has been loaded to be displayed in a visual representation on-screen. The user may scroll and zoom the visualised model to focus on specific details. The actor starting this use-case is the user.

4.2.1.4 Manipulate Model

This use-case describes how the user can manipulate and modify a network model, including the creation of a model from scratch. The user may add elements (servers, computers, firewalls, routers, etc.) to the model and create connections between elements to model an actual network for analysis. Each element may have attributes defined per the schema described in section 3.1.

4.2.1.5 Save Model

This use-case writes a model to persistent storage. Persistent storage is not limited to the host system, but may be a network-accessible location such as a remote system or SAN. Storage may include the act of cloning an existing model. The actor starting this use-case is the user.

4.2.1.6 Preprocess Model

This use-case allows a model to be processed to modify elements or attributes in preparation for analysis. The actor starting this use-case is the RapidMiner application.

4.2.1.7 Model Analysis

This use-case invokes a model analysis operator to analyse a network model, retrieves the results of the analysis, and transforms the results retrieved from that analysis into the normalised format used within RapidMiner. The actor starting this use-case is the RapidMiner application

4.2.1.8 Prioritization

This use-case invokes a prioritization operator to analyse the results retrieved from a model analysis, and transforms the results retrieved from that analysis into the normalised format used within RapidMiner. The actor starting this use-case is the RapidMiner application

4.2.1.9 Postprocess Model

This use-case allows the user to define filtering and sorting options for visualization, prior to invoking visualization. The actor starting this use-case is the RapidMiner application.

4.2.1.10 View Results

This use-case describes how the processed results of an prioritization analysis are displayed for the user. The user may apply further sorting or filtering criteria to display the data in the desired format. The actor starting this use-case is the RapidMiner application.

4.2.1.11 Export

This use-case describes how data under analysis is written to persistent storage. The actor starting this use-case is the RapidMiner application.

4.2.1.12 Iterate

This use-case describes how analysis may be performed iteratively on the same model, where each iteration of the analysis is performed with one or more analysis parameters changed. The actor starting this use-case is the RapidMiner application.

4.3 Logical View

The system is logically broken into packages:

1. Model - Contains classes for the creation and manipulation of network models.
2. Shared - Contains classes for the tasks of validating and persisting network models.
3. Plugins - Each plugin represents one RapidMiner operator.
4. Storage - An abstraction for the database or file that data is written to/read from.

Figure 3 shows the logical view for the demonstrator. Each package and any subpackages are described in the following sections.

4.3.1 Model Package

The task of creating and manipulating network models is embodied in an application that is separate from the RapidMiner application. This model manipulation application (MMA) is dedicated to network modeling, with sub-packages designed to facilitate the effort. The Shared package is used by the MMA to re-use the classes for network model persistence and XML validation.

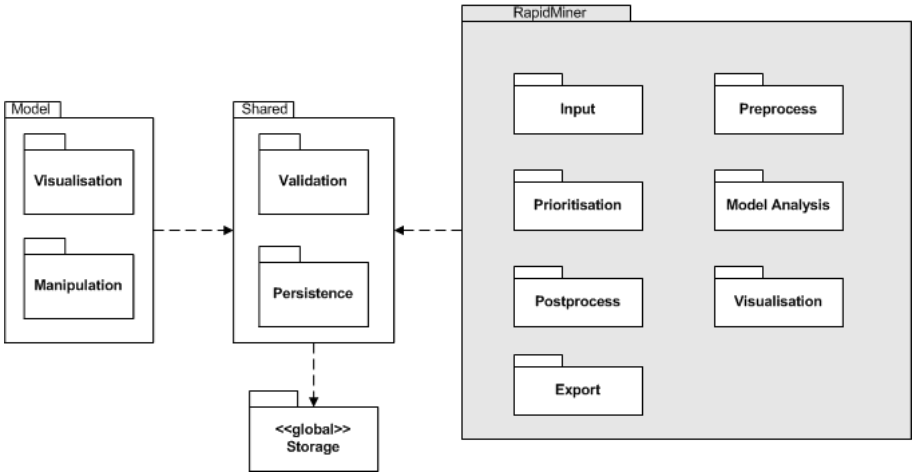

Figure 3: *Logical View*

4.3.1.1 Visualization

This package is responsible for rendering network models into a visual format. Major tasks include:

- Using the XML description of the network, the MMA must render a graph representation of the network showing network elements (e.g. servers, routers) and their interconnections.
- Allow the user to examine data attributes for an element in a logical way

4.3.1.2 Manipulation

Due to the use of an XML-based language to describe network models, they may be created in a simple text editor. This is, however, a very tedious and error-prone way to create XML documents, particularly when network complexity begins to rise. This package allows users to manipulate network models graphically. As the model is manipulated, the underlying XML is updated. Major tasks include:

- Create, delete, and edit elements in a network
- Define and change attributes for the element
- Define and change the connections between elements in the network
- Update the XML definition of the network model as elements are manipulated

4.3.2 Plugins

The analysis portion is comprised of several operators for RapidMiner compiled into plug-ins, each of which is defined as a package within this architecture. Note that these operators are defined as the baseline set necessary to produce a functional demonstrator. Efforts to develop other operators (e.g. a prioritisaion operator based on a different algorithm) are certainly feasible within the RapidMiner framework. Operators may be developed in any language the implementor desires, so long as the code can be invoked or otherwise called by an adapter that is loaded into RapidMiner. For example, the MulVAL software is written in Prolog. An adapter will be created that invokes MulVAL and passes its results into RapidMiner for prioritization or other further analysis. This adapter will be generic if possible so as to allow the use of the same adapter for all external operators.

4.3.2.1 Input Operator

The input operator provides a means to read network model files produced by the MMA into RapidMiner in preparation for analysis. Major tasks for this operator include:

- Read the specified XML file containing the network description
- Perform schema validation to ensure that the model is correctly formed
- Transform a valid model into a normalised form for input to other operators

4.3.2.2 Preprocessing Operator

The preprocessing operator performs any manipulations on the imported model prior to Model Analysis. The significant tasks for this operator are to be determined, but the capability to manipulate the model prior to analysis is seen as needed functionality for the solution.

4.3.2.3 Model Analysis Operator

A model analysis operator passes network model data to a network model analysis algorithm. Major tasks for this operator include:

- Process a normalised network model in whatever way is required to transform the data into suitable input to the model analysis algorithm
- Pass the network model to the algorithm for analysis
- Transform results into a normalised form for input to other operators

A MulVAL model analysis operator will be created that serves as an adapter for the MulVAL application.

4.3.2.4 Prioritization Operator

A prioritization operator uses the output of a model analysis operator to perform sorting of those results based on whatever criteria are important to the prioritization algorithm. Major tasks for this operator include:

- Process normalised model analysis data in whatever way is required to transform the data into suitable input to the prioritization algorithm
- Pass the data into the prioritization algorithm for analysis
- Transform results into a normalised form for input to other operators

An AssetRank prioritization operator will be created that serves as an adapter for the AssetRank algorithm. Operators encapsulating other prioritization algorithms may be created for the demonstrator, depending on availability of other algorithms during development.

4.3.2.5 Postprocessing Operator

This operator performs any tasks or transformations on data retrieved from another operator prior to visualization. Filtering and sorting of data may be specified in postprocessing. Major tasks for this operator include:

- Allow sorting and filtering of data prior to being passed to the next operator in the sequence
- Others TBD

4.3.2.6 Visualization Operator

The visualization operator displays the results of analysis for the user. For the purposes of the demonstrator, visualizations for viewing attack graphs and the network model will be developed. Major tasks for this operator include:

- Display the attack graph generated by model analysis
- Display the attack graph and network model overlaid on each other
- Another view type (TBD)
- Allow sorting and filtering of data to allow rendering of only certain parts of the attack graph and/or network model

The standard RapidMiner visualization operators are also useable, for example to simply show prioritization results in a tabular format.

4.3.2.7 Export Operator

This operator is used to write normalised data to persistent storage. Major tasks for this operator include:

- Write normalised data to persistent storage

The standard RapidMiner Writer operators may be useable, rather than writing a custom operator. Investigation during the design phase will determine whether a custom operator is required or not.

4.3.3 Shared Package

The Shared package contains those classes that are common to the Plugin and Model packages. Its purpose is to encapsulate these classes into a library that may be shared amongst all common users to promote code re-use.

4.3.3.1 Persistence

This package is responsible for persisting models created and modified in the MMA. Major tasks include:

- Read models from persistent storage (Per requirement PR-1, models will be stored in XML format within Microsoft SQL Server)
- Write models to persistent storage

4.3.3.2 Validation

This package is responsible for validating network models. Major tasks include:

- Validate models against the XML schema of the network definition language
- Create logs of errors and warnings found in a model

4.4 Implementation View

All software resides within either the RapidMiner plugin, the model manipulation application, or a library of shared classes. The model manipulation and RapidMiner applications provide the access layer for the application. Persistent data is stored either in flat files within a file system or in a Microsoft SQL Server database.

5 Development

Development of the demonstrator will be performed in two stages. The first stage will be the development of the model manipulation application, and the second the development of the RapidMiner plugins. Each development effort requires similar decisions be made:

- Which programming language(s) to use
- Which development tools to use
- What libraries are necessary to support development
- The development environment required

5.1 Development Environment

During the requirements gathering exercise, it was determined that the system be a monolithic application. This removes the necessity for multiple computers for development and testing. With that requirement, the development environment can be defined as a common environment for both tasks.

- A system with the development tools and libraries installed, for software development
- A system with no development tools or libraries installed, for testing
- Access to a revision control system with a backup strategy in place and functioning

5.2 RapidMiner Plugins

The choice of RapidMiner immediately dictates several choices for the development environment, detailed in the following sections.

5.2.1 Programming Language

The method and framework for RapidMiner plugins is well defined as Java-based. This meets the requirement specified in OR-1, that the application must be cross-platform (Windows, Mac, Unix).

5.2.2 Libraries

A number of libraries are included in the RapidMiner distribution. Of particular interest to the plugins will be a library for presenting visualizations such as attack graphs and possibly network models from within RapidMiner. RapidMiner is distributed with the JUNG (Java Universal Network/Graph) Framework, which is used by all existing RapidMiner operators for visualization. In the interest of compatibility and consistency, the RapidMiner plugins will use JUNG as well.

5.2.3 Development Tools

For Java development, the Eclipse development environment[1] is preferred. As noted in section 5.1, access to a revision control system is a required development tool.

5.3 Model Manipulation Application

The model manipulation application is brand new software, designed on a green field. With this in mind, the decisions may be made with an eye towards maximising productivity and maintainability, while minimising development time. This primarily affects the choice of language and libraries to use for development. A key selector is the ability to choose libraries that provide the best graphical display flexibility, specifically the ability to build and display network models. The following describe the choices made for the model manipulation application.

5.3.1 Programming Language

Per requirement OR-1, the application must be cross-platform (Windows, Mac, Unix). This narrows the choice of language to those languages that are truly platform independent and can be used to build a graphical user interface. This requirement effectively limits the choice of language to one of Java, Ruby, or Python.

5.3.2 Libraries

The choice of a library for the visualization of networks is of primary importance. The visualization library must allow for arbitrarily large networks, provide zooming, scaling and scrolling across large graphs, and be proficient at layout optimization. An examination of available visualization libraries for Java, Ruby, and Python reveals a great deal of choice for Java, and very little for Ruby and Python via their GUI libraries. Combined with the rich selection of other libraries and frameworks available for Java, it seems to be the clear choice for development of the model manipulation application. An additional benefit of

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

choosing Java is maintaining the code base between RapidMiner plugins and the model manipulation application in a common language.

Given the choice of Java as the development language, other libraries and frameworks present themselves as suitable for development of the model manipulation application:

- Spring Framework. Provides inversion of control, aspect-oriented programming, and model-view-controller functionality.
- JUnit. Provides a unit testing framework.
- Visualization. As the RapidMiner application ships with the JUNG Framework which will be used for the RapidMiner plugins, this same library will be used for the model manipulation application.

5.3.3 Development Tools

For Java development, the Eclipse development environment is preferred. As noted in section 5.1, access to a revision control system is a required development tool.

6 User Interface

This section presents mockups of the interfaces for the RapidMiner operators and model manipulation application. These mockups are intended to convey a sense of what the demonstrator's interfaces may present to the user, but are not intended to be complete. Full definition of the interfaces is a task within the design phase, which will be carried out with input from DRDC project stakeholders to ensure that all user interfaces are fulfill requirements and are signed off by project owners before development proceeds.

6.1 RapidMiner

The user interface for the demonstrator within RapidMiner will follow the existing Rapid-Miner conventions.

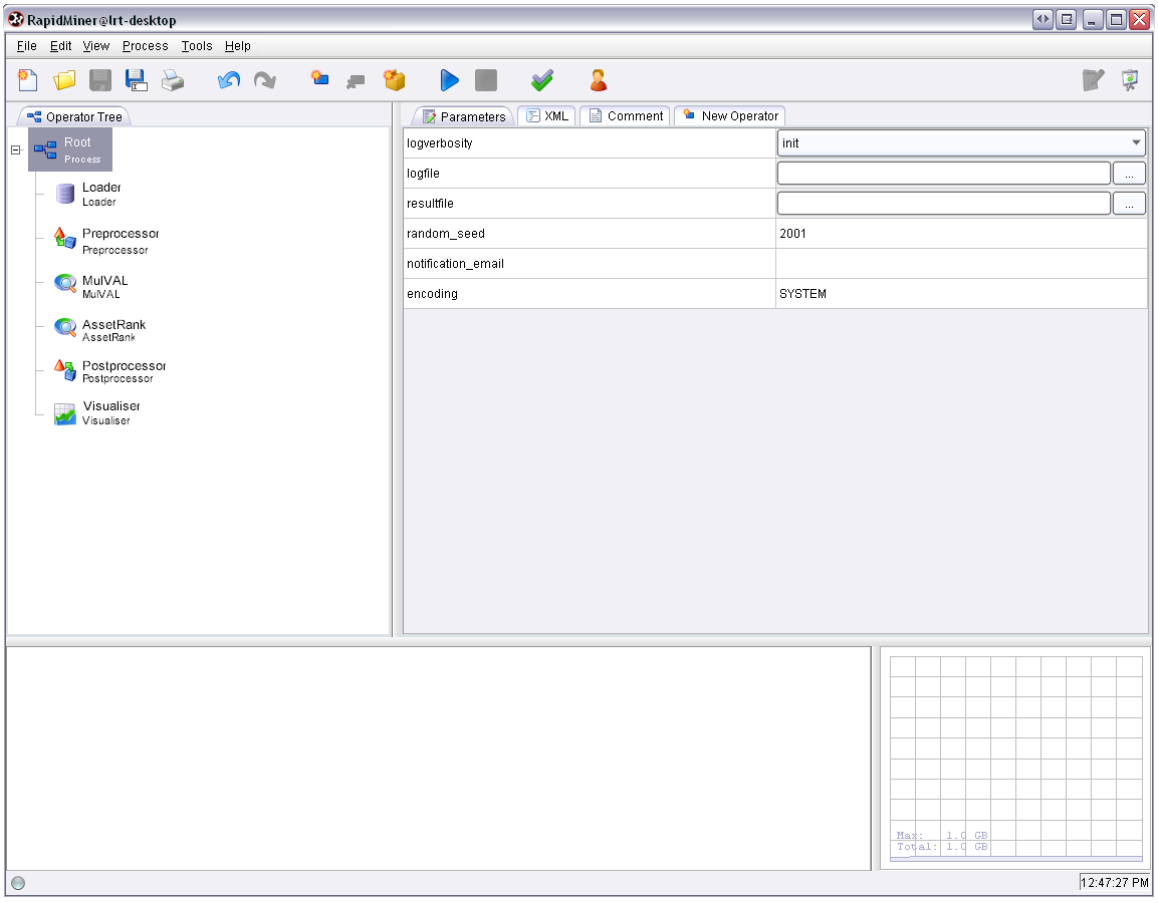

Figure 4: *RapidMiner UI*

Figure 4 shows a mockup of the demonstrator's appearance in RapidMiner. Operators will

appear in the execution list in the order they are added, though the order of operators is predefined and should not be always the same – that is, load, preprocess, model analysis, prioritization, postprocessing, visualization. As with all RapidMiner operators, each operator will have some number of variables that may be altered to influence the execution of the analysis.

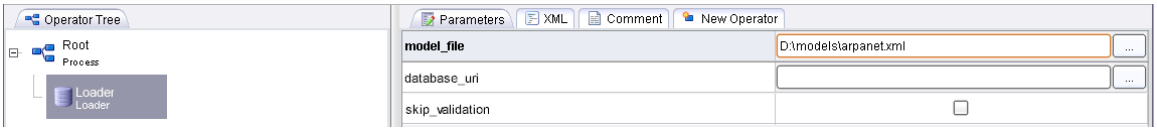

Figure 5: *Example of an Operator*

Figure 5 shows a mockup of the demonstrator's loader operator. This operator is responsible for loading the network model from persistent storage and validating its XML for correctness before passing it to preprocessing in preparation for model analysis. This operator indicates the ability to load network models from file storage or from a database, as well as the option to skip XML validation on the loaded model. The other operators will similarly have options and inputs relevant to their operation.

6.2 Model Manipulation Application

The user interface for the demonstrator model manipulation application is not bound by any existing convention, only by requirements gathered from DRDC staff and potential users.

Figure 6 shows a mockup of what the model manipulation application may look like during the creation and manipulation of a network model. Various nodes are arranged in the graph, with the interconnections made between them. In this example, each node is either a template of a Standard PC or a Cisco model 1801 router. In the case of these templates, all parameters would have been previously defined using the template facility, highlighted in the Model menu. The use of templates allows for an increase in the speed of creating models. As shown in the Generic portion of the Elements tree, numerous element types are predefined and sorted for ease of location during the model creation phase.

Figure 7 shows a mockup of the model manipulation application showing the attributes of an element in the network model. In this case, the previously-named ”node40” from Figure 6 has been renamed to ”desktop1”. Applications have been added to this node, namely Microsoft Internet Information Server version 6.0, and WS_FTP server. Security patches for the Windows XP operating system are shown, which may or may not have been configured as part of the template. A security patch for IIS is noted as well, which would have been added when the application was added to the system. Not shown is the

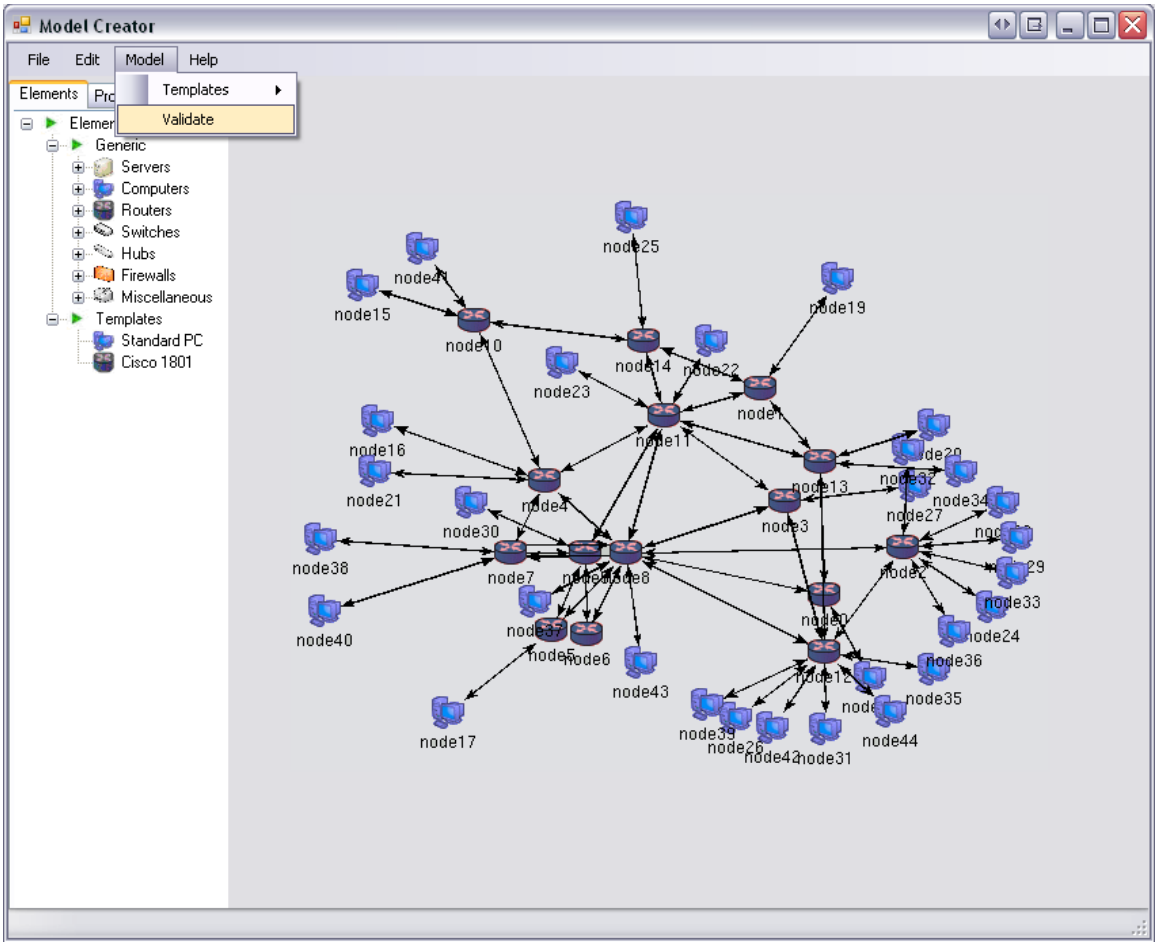

Figure 6: *Mockup of the MMA creating a model*

interface for defining attributes in an element. This is left to the design phase of the MMA, in consultation with DRDC to ensure that the proper functionality is achieved.

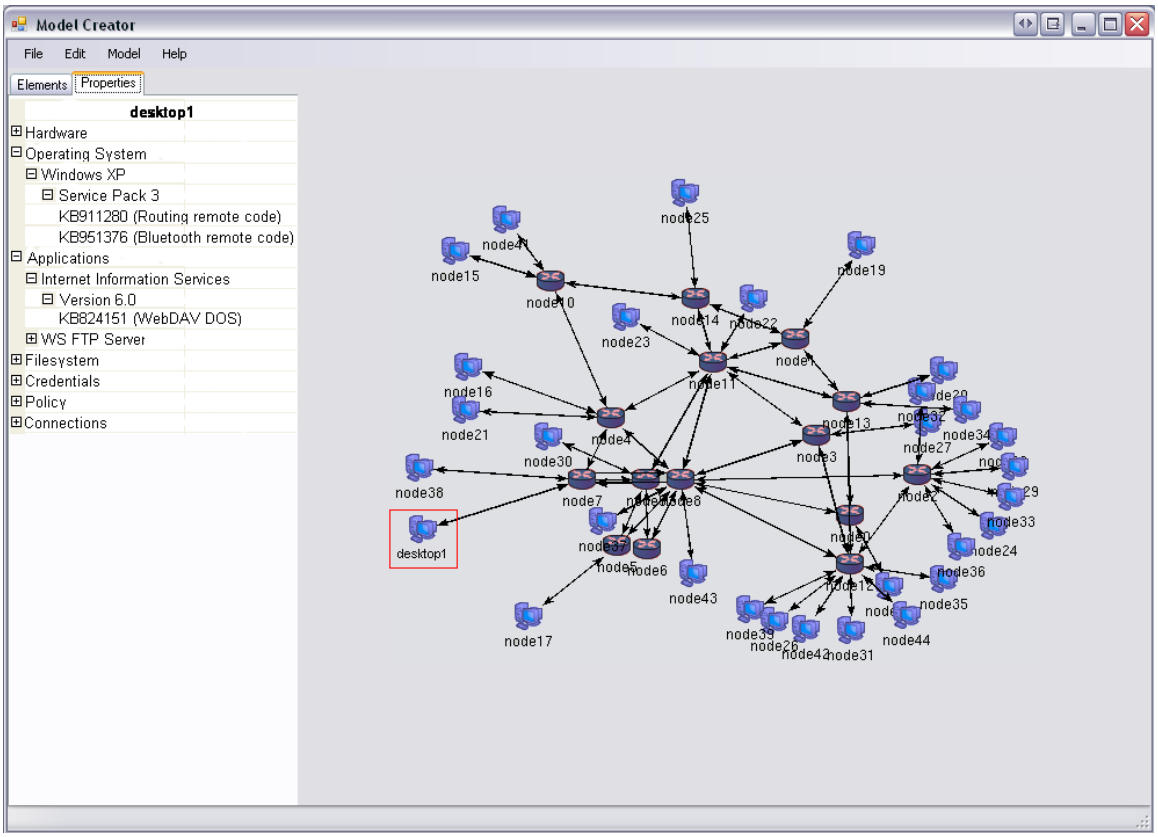

Figure 7: *Mockup of the MMA examining a model element*

7 Work Plan

Given the research nature of this project, the multi-level integration requirements for such items as network models, analytical modules and emerging technologies, and the need to work closely with the targeted end users to achieve a useful and supportive toolset, it is recommended that a cyclic or iterative methodology be used to develop the demonstrator. An iterative development approach such as Rapid Application Development (RAD) helps to steer the project through the development process in the context of a strong collaboration with the targeted end user to arrive at a more quickly at a solution that best reflects the user's needs. A RAD approach ensures that the development activities are focussed since there is regular interaction and correction on the part of the eventual solution owner. The owner thus has the ability to guide the effort through stages of prototyping, testing and overall solution validation. A RAD approach is very effective for projects of this nature where the end user and development teams are small and cohesive. At the beginning of each RAD cycle, it will be made clear, through a traceability matrix, exactly which requirements will be addressed during that cycle to ensure that the initially defined demonstrator design requirements are balanced with the design decisions that are arrived at during the collaboration phase.

The implementation cycle for such a project will therefore be adaptive allowing each technical track and cycle to be logically and effectively linked while allowing continuous feedback loops to ensure higher level of verification, validation and control. Continuous change and improvement form an integral part of the approach and is assumed. To succeed in this approach, strong documentation management is required to not only show the progression of the solution development, but also the rationale for design elements, justification for decisions made and lessons learned through the entire effort.

Our key findings observed on projects using the Cyclical Approach are:

- The necessity to involve end users at every track cycle
- The requirements to control the scope with sound change management practices
- The need to determine what is feasible for the next cycle(s)
- The obligation to instil project discipline to track interdependencies, progress, cost and risks

It is proposed that the demonstrator development effort be divided into four technical tracks.

Track1 Model Manipulation This track will create the supporting capability to create and modify network models that describe real-world or proposed network environments. This track will result in the creation of a tool that can import, from a local or centrally stored

repository, network models and present them visually. The model manipulation tool will also provide the output visualization for displaying the results of the analytical models layered onto the source network model for greater interpretation and understanding. As with all tracks, this solution component shall leverage open and extensible standards to communicate and store information through a set of well defined and published interfaces.

Track 2 Analytical Modules This track will create a set of modules based upon a framework that is integrated with RapidMiner. These module, namely MulVAL and AssetRank, will be made available through the RapidMiner interface and called upon to perform analysis against an established network model. The conditions under which the analysis will be made (e.g. location of the attacker in MulVAL) shall be specified by the operator and integrated with the analysis. The analysis results will be linked to the model and made available for interpretation by the Model Manipulation tool. Efforts will be made to simplify and standardize the module framework to support the development of future analytical modules.

Track 3 Model Storage This track will establish the representation and storage mechanism for all model elements, the interfaces needed to access these elements and any needed translations to modify the data from external source or between repositories. Model elements will include: the network data model, the vulnerability data model, data models to support the analytical modules (both pre and post condition), and any data related to the visualization of analytical results.

Track 4 Model Building and Quality Assurance This track will build representational network models suitable for validating the solution. A series of models will be created from simple (10 systems) to complex (200,000) with supporting vulnerabilities and attack path potential. The solutions from tracks 1 through 3 will leverage these models for unit testing, module testing and solution demonstration. More complex models will leverage templating, grouping and abstraction to allow a large model to be build from a smaller set of network configuration elements.

Within these 4 tracks, the following development cycles are proposed. For the project implementation phase, it is assumed that the first cycle will begin April 1, 2009 and the final cycle will conclude September 30, 2009. Each cycle represents a 2 week work cycle ending with a development review with all stakeholders where project progress is reviewed, existing cycle tasks are evaluated and the set of tasks for the following cycle are established.

Cycle	Track 1	Track 2	Track 3	Track 4
1	Create the model building tool framework from existing tools/libraries	Install and Configure RapidMiner for a development environment to support the creation of analysis modules	Begin CIM XML world modeling	Create the initial model 1 (M1 - small, 10 systems)
2	Create M1 in the tool framework	Create a generic analytical module shell in RapidMiner	Continue CIM and linked NVD modeling	Create model 2 (M2 - medium, 1000 systems)
3	Local loading/storage for network model	Create a MulVAL analytical model for RapidMiner	Populate M1 and M2 in storage	Begin model 3 (M3 - large, 200,000 systems)
4	Remote loading/storage for network model	MulVAL module to load local information	Model MulVAL data in storage	Complete M3 using groups, templates and abstraction
5	Advanced functions (copy/paste)	MulVAL module to load remote information	Model MulVAL external/configuration data in storage (NVD)	Develop test cases for M1 and M2
6	Advanced functions (templating)	MulVAL module to load external/configuration information		Develop test cases for M3
7	Layering of model data for visualization	Create an AssetRank analytical model for RapidMiner	Create model for AssetRank configuration and results	Full MulVAL test on M1
8	Technical investigation into a visualization module framework	Complete AssetRank analytical module for RapidMiner		Full MulVAL test on M2
9	Visual layer for MulVAL results	Module refinements		Full MulVAL test on M3

10	Visual layer for AssetRank results	Module refinements		Full AssetRank test on M1,M2 visualization of MulVAL results
11	Visual layer refinements			visualization of AssetRank Results
12	Visual layer refinements			Demo

Table 8: Work Plan

Note that the plan presented in the above table reflects an initial assessment of the tasks to be done to create a demonstrator and the list of tasks per cycle will be refined over the course of the project. As previously stated, strong documentation management will be enforced at each cycle review period. Each RAD cycle, comprised of a 2 week schedule, will require the following effort:

- IT Security Research Specialist: 4.5 days per cycle (50 days approx.)
- IT Security Engineer 1: 9 days per cycle (100 days approx.)
- IT Security Engineer 2: 3 days per cycle (33 days approx.)

The end product for the development cycle will be a demonstrator that includes module frameworks to support the development of additional analytical modules and visualization routines. The limited length of the full development cycle will support the creation of a limited set of visualization capabilities, specifically attack path visualization, asset rank prioritization and network element drill-down. Future development cycles will build upon the established framework to further refine the visualization capabilities of the solution to meet ongoing end user requirements.

Annex A: Interview Questionnaire

1. What do you see as your primary use of a defensive posture capability within your current capacity?
2. To what problem areas do you envision applying said capability?
3. What do you like about your current system?
4. What do you dislike about your current system?
5. How would you improve your current system?
6. How would you interact with a tool capable of automating defensive posture and providing dynamic feedback?
   (a) What would your graphical requirement be?
   (b) What would your configuration requirements be?
   (c) In which way would you prefer to be able to configure such a solution?
   (d) Is there a preferred way configurations should be stored, edited, archived, etc.?
   (e) What should the outputs of the system be (i.e.: what type of reports/information (printed and onscreen))?
   (f) How would you use the results?
7. What systems would need to be comprehensible by such a system for it to be maximally useful to you in your daily work?
8. What type of information from the systems being monitored do you consider to be paramount?
   (a) Do you rank physical assets or virtual (logical) assets (i.e., files/information) as a whole or individually?
   (b) How do you currently rank the value of a given asset?
   (c) Is your asset ranking system in any way automated or easily automatable?
9. What would an effective system do?
10. What are your concerns about such a system?

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Bell Canada
333 Preston St., Suite 1100, Ottawa, Ontario, K1S 5N4

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Network defensive posture demonstrator

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Henderson, G.; Bacic, E.; Tremblay, L.

5. DATE OF PUBLICATION (Month and year of publication of document.)

August 2009

6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)

56

6b. NO. OF REFS (Total cited in document.)

0

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Ottawa
3701 Carling Avenue, Ottawa ON K1A 0Z4, Canada

9a. PROJECT NO. (The applicable research and development project number under which the document was written. Please specify whether project or grant.)

15bo03

9b. GRANT OR CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

W7714-071028

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC Ottawa CR 2009-133

10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The Network Information Operations (NIO) section at DRDC Ottawa is performing work under the Dynamic Computer Network Defence (CND) Applied Research Project, the goal of which is to provide network operators with situational awareness of their network. Crucial to this awareness is the knowledge of what assets residing on their network are critical to their operations, and what assets on their networks are exposed, that is, have a vulnerability that would allow an adversary to violate the confidentiality, integrity or availability of the asset. The NIO section has defined the combination of these two elements as network defensive posture: the set of exposed, critical resources on the network. Further, the defensive posture of a network is dynamic. The network critical resources may change with time in response to changing missions and operational priorities. At the same time, the network state can be altered by new software installations, the discovery of new vulnerabilities in existing software, changes to firewall rules, and other network events. Both types of changes affect the defensive posture.

A milestone in the Dynamic CND project is to create a demonstrator of a network defensive posture system. This contract addresses this requirement, and will provide an environment in which researchers at DRDC Ottawa can continue their work. Through previous investigation, we have determined that the MulVAL open-source software package is appropriate for our needs, and as such will be used in the development of the defensive posture analysis system. MulVAL is a logic-based network security analyzer powered by the XSB Datalog interpreter. It is capable of performing a multi-stage, multi-host vulnerability assessment of a computer network, determining all potential attack paths in a given network.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

MulVAL
AssetRank
Attack path
computer network defence
CND
network security

Defence R&D Canada

Canada's leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca