2013 5th International Conference on Cyber Conflict
K. Podins, J. Stinissen, M. Maybaum (Eds.)

A Cyber Attack Modeling and Impact Assessment Framework

Igor Kotenko
Laboratory of Computer Security Problems
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
Saint-Petersburg, Russia
Email: ivkote @comsec.spb.ru

Andrey Chechulin
Laboratory of Computer Security Problems
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
Saint-Petersburg, Russia
Email: chechulin@comsec.spb.ru

Abstract: The paper suggests a framework for cyber attack modeling and impact assessment. It is supposed that the common approach to attack modeling and impact assessment is based on representing malefactors' behavior, generating attack graphs, calculating security metrics and providing risk analysis procedures. The main aspects outlined are achieving near-real time mode, event analysis and prognosis mechanisms, security and impact assessment. To optimize the attack graph generation and security evaluation we apply an anytime approach to have the result at any time by applying a set of algorithms with different timelines and precision. The architecture of the Cyber Attack Modeling and Impact Assessment Component (CAMIAC) is proposed. We present the prototype of the component, the results of experiments carried out, and comparative analysis of the techniques used.

Keywords: *attack modeling, attack graphs, security metrics, impact assessment, anytime algorithms*

1. INTRODUCTION

Currently, computer networks are playing an important role in many areas. The increasing size and complexity of networks lead to the growth of complexity of their security analysis. Possible financial, political, and other benefits, which can be gained by cyber attacks, lead to substantial increase of the number of potential malefactors. Despite these facts, the existing security analysis is a process which still dependents mainly on the experience of security administrators. All these problems define the importance of the research and developments in the area of automated security analysis of computer networks.

There are many approaches to the network security analysis. One of the promising approaches consists in cyber attack modeling and impact assessment based on attack graphs (or trees). Studies related to building and analysis of attack graphs have been conducted for over 20 years. *Attack graph* is a graph which represents all possible sequences of the malefactor actions that lead him/her to the goals established. These action sequences are also called *attack traces*. The main disadvantage of this approach is its *computational complexity*. Building a complete attack graph for a malefactor (and especially for a set of malefactors) is a computationally complex problem and usually takes a long time. For instance, for a small size network the attack graph can be formed quickly, however, when the graph must be built for a network which includes hundreds or even thousands of hosts and the result should be obtained in a limited time (or even in real time), the graph based algorithms require a very large amount of computational resources. Moreover, over time the composition of hosts and links between them can be changed, and the attack graphs will require reconstruction. Thus, at the present time, the usage of attack graphs in systems operating in near real-time mode, for example, in Security Information and Event Management (SIEM) systems, is very complicated.

This paper suggests a framework for designing the Cyber Attack Modeling and Impact Assessment Component (CAMIAC) which implements the attack graph generation, real-time event analysis techniques, prognosis of future malefactor steps, attack impact assessment, and anytime approach for attack graph building and analysis. In contrast to the existing works of the authors [17-19], the paper describes the attack modeling and impact assessment solutions directed to optimization of attack graph building and analysis process with the goal to enable their usage in the systems operating in near real time. The main contributions of the paper are as follows: two stages based algorithm for attack graph building, the basic principles of real-time event analysis, the approach to identify possible malefactors by analyzing the compliance between security events and attack graphs, the application of anytime approach for the attack graphs analysis.

The paper is organized as follows. Chapter 2 analyzes the state-of-the-art in attack modeling. Chapter 3 outlines briefly key elements of suggested techniques for attack modeling and impact assessment. Chapter 4 presents a prototype of the CAMIAC. The architecture and implementation are described. Chapter 5 describes a case study and an example of the experiments with the prototype of the CAMIAC. It is shown how suggested approaches could be applied to a real use case. Also the comparison with existing tools is presented. Conclusion surveys main results and next steps of the research and development activities.

2. RELATED WORK

There are a lot of papers, which consider different approaches to attack modeling and security evaluation taking into account various classes of attacks. We analyze briefly current state-of-the-art in representation of attack scenarios and malefactors, generation of attack graphs, determining security metrics, combining service dependency graphs with attack graphs, representing zero day attacks and specification of platforms, vulnerabilities, vulnerability scorings, attacks, weaknesses and configurations.

In [1, 20, 30] *attacks are represented* in a structured and reusable tree-based form. In [30] a high-level conceptual model of attacks based on the intruder's intent (attack strategy) is suggested. The comprehensive work using the so-called tree based approach is proposed in [1]. This paper describes means for documenting attacks in a form of attack trees. One of the most important problems in security analysis is the malefactors' classification and model construction. In [38] the task of modeling and simulation of intelligent, reactive attackers is described. The suggested computer network attack model uses an action representation based on the GOLOG situation calculus [13] and goal-directed procedure invocation.

Different approaches, which use *attack graphs and trees for security analysis*, have been suggested. S. Hariri et al. [40] calculate global metrics to analyze and proactively manage the effects of complex network faults and attacks. S. Noel, S. Jajodia et al. [25, 41] propose a technique based on determining the minimum-cost network hardening via exploit dependency graphs. I. Kotenko and M. Stepashkin [14-16] are focused on security metrics computations based on attack graph representation of malefactor behavior. R. Lippmann and K. Ingols [35] propose to use attack graphs to detect firewall configuration defects and host critical vulnerabilities. This approach was extended by taking into modern network attacks threats (zero-day exploits and client-side attacks) and countermeasures (intrusion prevention systems, personal firewalls, and host-based vulnerability scanners) [23]. J. Ryan and D. Ryan [21] suggest calculating metrics based on failure-time analysis. L. Wang, S. Jajodia et al.

[24, 26] propose to calculate attack resistance metrics based on probabilistic scores by combining Common Vulnerability Scoring System (CVSS) scores [11]. N. Kheir et al. [32] suggest an implementation of confidentiality, integrity and availability metrics using the notion of privilege, which is inspired by access permissions within access control policies.

There is a new trend of research in attack modeling, which is to *combine attack graph models and service dependency models*. In their essence, attack graphs represent possible attacker actions in the light of current system configuration. Meanwhile, they do not represent service dependencies and their underlying connection requirements. N. Kheir et al. [31] propose to extend the use of CVSS metrics in the context of intrusion response, by supplying this metric with dynamic information about system configuration and service dependencies structured within dependency graphs. The dependency graph is further used to evaluate the overall impact of an attack, thus replacing the informal environmental parameters in the CVSS vector. Nonetheless, the problem with this approach is that it does not provide clear evidence on how to interface service dependency graphs with attack graph models.

The analysis of network security against unknown zero day attacks is also a relatively new topic of research. Zero day attacks can be defined as attacks which use unknown vulnerabilities. E. Bursztein [12] extends the security analysis approach, based on game theory, by taking into account zero day exploits. L. Williams [27] presents a practical realization of the approach to calculate the possible number of zero day vulnerabilities. M. McQueen et al. [29] attempt to evaluate the total number of possible zero day vulnerabilities for one day. K. Ingols et al. [23] suggest ordering different applications by the seriousness of consequences of having a single zero day vulnerability. L. Wang et al. [24] propose a security metric called k-zero day safety. It is based on how many unknown vulnerabilities are required to compromise a network asset, regardless of the type of vulnerabilities.

Very important relevant work is connected with research and developments in coherent *description of vulnerabilities, attacks, weaknesses, security policies and configurations, lists of the software/hardware installed on each platform, events, countermeasures*, etc. Common Vulnerabilities and Exposures (CVE) [9] contains the list of known information security vulnerabilities and exposures. Usage of the National Vulnerability Database (NVD) [33] based on CVE dictionary is the basis for constructing of attack graph via known vulnerabilities. Common Vulnerability Scoring System (CVSS) [11] is an open and standardized vulnerability scoring system for vulnerabilities rating. Common Weakness Enumeration (CWE) [10] contains a unified, measurable set of software weaknesses. Usage of the database of weaknesses can improve the quality of the zero-day based attack graph generator

module. Common Platform Enumeration (CPE) [7] provides a unified description language for information technology systems, platforms, and packages. Common Configuration Enumeration (CCE) [6] gives common identifiers to system configuration issues. Common Attack Pattern Enumeration and Classification (CAPEC) [5] helps to capture and use the attacker's perspective. Usage of attack patterns allows applying sequences of known and zero-day vulnerabilities in one attack action. Common Remediation Enumeration (CRE) [8] defines a security-related set of actions that result in a change to a computer's configuration. Remediations may be motivated by discovered vulnerabilities or misconfigurations.

3. PROPOSED TECHNIQUES

Let us consider the main techniques that are suggested in the CAMIAC: attack graph generation, real-time event analysis, prognosis of future malefactor's steps, attack impact assessment, and anytime approach. These techniques are based on using a comprehensive security repository, efficient attack graph (tree) generation techniques, taking into account known and new attacks based on zero-day vulnerabilities, stochastic analytical modeling, and interactive decision support to choose preferred security solutions (countermeasures). Not all these aspect are outlined in the paper due to limited volume.

A. *ATTACK GRAPH GENERATION*

Let us consider at first the basic definitions needed for attack graph generation.

Basic objects define the graph *vertexes*. They are linked to each other by *edges* to form different sequences of malefactor's actions. Basic objects and the links between them are included in the *network model* which is used for attack graph generation. Basic elements can belong to two types: "host" and "attack action". The objects of the "*host*" type describe hosts discovered and attacked by malefactors, while the objects of the type "*attack action*" describe all distinguishable actions of malefactors.

The algorithm of generating common attack graph is based on implementation of the following sequence of actions: (1) preparatory actions which allow a malefactor to move from one host to another; (2) reconnaissance actions for detection of "live" hosts; (3) reconnaissance scenarios for detected hosts; (4) attack actions based on vulnerabilities and (5) auxiliary actions.

As the result, all *attack actions* are divided into the following classes: (1) reconnaissance actions; (2) preparatory actions within the limits of

malefactor's privileges (these actions are used for creation of conditions needed for implementation other attack actions; (3) actions for gaining the privileges of local user and of administrator; (4) confidentiality, integrity and availability violation.

As in [45], we use three necessary conditions for adding a potential attack in the attack graph: (1) the protected system has vulnerabilities; (2) an attacker needs knowledge and resources to perform attacking activities; (3) an attack accomplishment facilitates the achievement of the malefactor goal. The first condition is determined completely by properties of the protected system. The second one is defined by both the system and the malefactor model properties. The third one is determined by the malefactor goals.

Respectively, at the first stage - the *stage of preparation and construction of attack trees* - a *3-dimension matrix* is formed for each host according to the following information:

(1) class of attacks (data gathering, preparation activities, escalation of privileges, attack goal realization);

(2) needed access type (remote source without access rights, remote user of the system, local user of the system, administrator);

(3) restriction for malefactors (by malefactor knowledge, zero-day vulnerabilities, etc.).

As a result for each host a set of corteges (attack action class, access type, and malefactor knowledge level) is formed, for each cortege in its turn a list of particular attacks and vulnerabilities needed for these attacks implementation is generated. The total list of vulnerabilities is formed on the base of host software and hardware description using CPE [7] and public vulnerability databases such as NVD [33]. Additional data sources for the detected vulnerabilities are the reports of security scanners such as Nessus, MaxPatrol, etc. In the CAMIAC the vulnerabilities are stored in the CVE format [10].

When constructing an attack graph, particular attack patterns described in the CAPEC format [5] are used. The CAMIAC uses these patterns not only as input information, but also allows producing new ones. They can correspond to the most often used sequences of vulnerability exploitations and other actions of the attacker. The patterns also contain attack descriptions that do not use vulnerabilities, for example at the initial stage of an attack the malefactor could gather information on available hosts. To specify in this case the attacker actions, the CAPEC-292 (Host Discovery) entry is used. It describes a group of various ways of scanning hosts and ports. This group contains, for example, such entries as CAPEC-285 (ICMP Echo

Request Ping), CAPEC-296 (ICMP Information Request), CAPEC-299 (TCP SYN Ping), etc.

The second stage is a *search of vulnerable software*. The examples of patterns used to describe malefactor actions are as follows: CAPEC-310 (Scanning for Vulnerable Software), CAPEC-311 (Fingerprinting Remote Operating Systems), CAPEC-300 (Port Scanning), etc.

On the third stage of *attack graph generation*, both particular vulnerabilities from the CVE dictionary and patterns like CAPEC-233 (Privilege Escalation) are used.

After forming matrixes of potential attacks, for each host the *possible malefactor type and his/her initial location* are chosen for the analyzed network.

The examples of the malefactor type are as follows:

(1) *External hacker*, a user having significant knowledge in information security field, but lacking any direct possibility to connect to the internal network; possible intrusion points, for example, are servers which can be accessed via the Internet (web servers, mail servers, etc.);

(2) *Internal user*, a user having basic knowledge in information security field with local user or administrator rights;

(3) *Worm/virus/botnet*, a program that can use a set of vulnerabilities specified in advance. It is supposed that in this case a part of internal network can be already infected.

The full *malefactor model* includes following parameters: type (internal, external, complex); initial privileges for each host of the network (none, remote user, local user, administrator); possible access points in the network; knowledge level (defines possible attack actions).

Further for each chosen malefactor model a list of possible goals is generated. For example, for the internal user it could be a revenge (causing maximum damage for the company). The goal of the external hacker could be the access to confidential information located on a server inside the network. For a worm at the first stage a goal can be its distribution, while at the second one it could be carrying out DDoS attacks.

Therefore, the malefactor in the CAMIAC is presented by a pair “malefactor model, goal”, which determines constraints on the usage of attacking actions and possible initial intrusion point into the network.

The *key elements of the suggested approach* are as follows: (1) for all malefactor

models the attack graph is formed in the same time on the basis of information gathered; (2) for each malefactor model the security metrics are evaluated; (3) for each malefactor who can successfully realize attacks, the list of possible countermeasures is formed.

Due to the fact that the attack modeling cannot be often fulfilled in real-time, its usage in real-time processes is limited. However, the generated attack graphs keep their actuality for a certain period of time (until significant changes in the security policy or physical network topology occur).

Thanks to this in the frame of the general system of event analysis it is suggested to use the attack graphs constructed in advance. These attack graphs can be used when solving two main tasks: (1) predicting subsequent malefactor actions and (2) analysis and detection of their past actions which led the system into its current state.

However, in some cases the attack modeling system needs to *update attack graphs*. For example, this necessity occurs when host characteristics (software and hardware, criticality, etc.), network topology and a list of possible malefactors are changed, as in these cases key objects (malefactor models, matrixes of host properties, etc.) are changed.

However in this case attack graphs are updated partly as the changes are calculated only for particular elements of matrixes. Due to this fact the computational complexity of the update decreases significantly.

B. REAL-TIME EVENT ANALYSIS

Attack graphs produced by the attack modeling component allow us to specify the ways of system security violation using different attacks. For a given malefactor, the attack graph corresponds to an attack tree, where the root represents an initial location of the malefactor and the leaves depict conditions which allow achieving the malefactor goals. That means that all paths from the root to the leaves are the sets of potential attacks. An attack scenario in its turn represents a minimum range of conditions the malefactor should meet to achieve the goal.

Depending on the kind of the graph, a scenario can represent a sum of some leaves or a subset of the graph elements including at least the root and one terminal leaf. Therefore, the graphs and the attack scenarios are supplementary notions. While the former allows getting some knowledge on potential attacks, the latter represent detailed information on particular attack type which can be included in a great number of graphs.

In fact the *task of real-time event analysis* consists in detection of a set of attack trees this event can belong to. In the end, an ideal result of the CAMIAC operation should be an attack scenario (i.e. a path from the root to its leaf) completely determining the malefactor model (i.e. goals and possibilities of the malefactor) and its possible subsequent actions.

In existing literature a range of approaches to find out the most probable attack scenario on the base of analysis of particular security events is proposed.

W. Lee et al. [48] suggest an approach allowing determining a scenario, which lower level events belong to. The paper also describes a technique for evaluation of probability of revealed malefactor goals and its subsequent steps. In the CAMIAC we apply this approach.

Three potential attack conditions described in [45] allow us evaluating a probability the attacker chooses exactly a given attack (proceeding from a fact that the malefactor uses a way of maximally quick achievement of his/her goal). On the basis of the probability of vulnerability exploitation by a particular malefactor model, it is possible to detect what generated attack tree the detected event relates to.

C. PROGNOSIS OF FUTURE MALEFACTOR'S STEPS

The most part of existing attack detection systems revealing current attacks cannot predict subsequent malefactor actions. It is obvious that the prediction of subsequent malefactor steps allows increasing the protection level of the system against malefactors. Thus, the main function of the security system becomes the detection of particular malefactors and generation of targeted protection measures rather than the discovery of particular attacks.

Let us consider the following example. If a malefactor conducts an attack against some host in a network, he/she can have the goal (1) to capture the control over the host for carrying out further attacks or (2) to assess data on the host.

To implement the system protection correctly, it is necessary to define the malefactor goals and predict its future actions. Let us consider these variants in detail.

The first goal supposes the malefactor will use the attacked host as an intermediate one, thus, it is not the object of the final goal. Therefore, we could predict the next actions of the malefactor: he/she will concurrently look for other intermediate hosts. If the malefactor captures the attacked host, the attack will be progressing from it. In this case it is worth to increase the sensitivity of rules for attack detection of the attacked host as a protection mean in order to detect the capture of a given host. It allows gathering additional information on the malefactor and his/her methods.

The second goal supposes the attacked host is the final goal of the malefactor and contains valuable information. It is likely that all hosts in the network controlled by the malefactor participate in the attack. In this case it is reasonable to block temporarily all suspicious or all available connections to this host as all hosts connected to the attacked host should be considered as potentially infected.

Thus, depending on the malefactor goal the response of the security system on one event detected in the network (attack) can differ. In fact the wrong malefactor goal detection can even assist the malefactor in the goal achievement. For example, when the attack goal is detected as one of the second type, the protection mechanism will allow the malefactor conducting DDoS attack, which may lead to successful implementation of some other attack, for instance IP spoofing.

We propose an approach that allows determining the future malefactor actions on the basis of the reports from an event correlation subsystem or an intrusion detection system (IDS) and the analysis of attack graphs created in advance for different models of potential malefactors. This approach also facilitates conduction of a retrospective analysis of events in the network, which allows revealing of unknown vulnerabilities (0-day).

The proposed approach includes the following steps:

(1) An attack graph(s) is (are) formed on the basis of the network and malefactor models;

(2) In a real network the network of connected sensors is formed. These sensors allow detection of particular attacking actions. A monitoring system allows producing a general picture of events occurred in the network on the base of information gathered by sensors. The events should be normalized, prioritized and correlated by an event correlation subsystem (or an intrusion detection system);

(3) Further the management subsystem finds correspondences between attack graphs and events in the real network. Thus, according to the analysis of incidents and data received from the attack modeling subsystem it is possible to conclude that there exists a sufficient probability that an incident “scanning of host C by host B” is followed by some undetected incident “host B was attacked by host A” and the next action of the malefactor is “host C is subjected to attack from host B”.

In the literature a number of approaches to determine a current attack scenario on the base of security events is described. For example in [3] the PHATT (Probabilistic Hostile Agent Task Tracker) algorithm using Bayes models is suggested. When revealing a scenario, this algorithm uses probabilities of attack fulfillment, contradictions in scenarios and works with a range of malefactors having different goals.

We use the following admissions simplifying the work of an algorithm are assumed:

(1) the malefactor uses only a single scenario at a particular moment;

(2) elements of a scenario have no strict time limitation;

(3) if the malefactor uses several scenarios, he/she runs them sequentially.

D. SECURITY AND IMPACT ASSESSMENT

Attack modeling needs to represent not only the sequences of actions, but also the attack consequences (in terms of impacts), as well as how countermeasures can mitigate these impacts and for which cost [32]. According to this principle we realize five main groups of security and impact assessment metrics.

The first group includes *metrics that are connected with topology, criticality and vulnerabilities of the analyzed system (hosts)*: the level of the host vulnerability which is defined on the base of the known vulnerabilities (we suggest to exploit CVE and NVD for its assessment); the level of the host criticality which is defined by its position and functions in the system (we suggest to define it on the base of CVSS); and the vulnerability of host to the zero-day attacks. The last metric allows considering the potential to compromise the system via unknown vulnerabilities. Here we use CWE to detect "weak places" of the system, and on the base of the approach in [23], depending from the malefactor type, we suppose the existence of zero-day vulnerabilities on the hosts where they lead to the maximum impact. By extension of the attack graph and taking into account weak places we define and use for impact assessment a set of possible unknown vulnerabilities.

The second group includes *metrics characterizing the attack, for example, attack potentiality.* When we define the *attack potentiality* (probability), the attack graph is used. These metrics are based on CVSS and the metrics of the first group, and allow calculating the integrated complexity and severity of the sequence of steps that are necessary to compromise the system assets.

The metrics of *the third group* characterize *the malefactor's potential* and are intended to define possibilities of the attack development. As the basis for such calculation we consider the malefactors position in the system and his/her skills (malefactor profile).

The metrics of *the fourth group* are *response efficiency and response collateral damage*. Response efficiency measures the response ability to reduce attack impacts. Response collateral damage evaluates negative influence of countermeasures on the system efficiency. To evaluate response collateral damage we use both attack graphs and service dependencies.

The last group includes *integral spatial characteristics of the system security* and a score of the system risk level. For example, the approach of qualitative express assessment of network security level uses the following basic metrics: *Criticality*(h) - criticality level of host h; *Severity*(a) - criticality level of attack action a; *Mortality*(a,h) - damage level caused by attack action, taking into account the criticality level of host; *Mortality*(S) and *Mortality*(T) - damage level of route S and threat T; *AccessComplexity*(a), *AccessComplexity*(S), *AccessComplexity*(T) - "access complexity" of attack action a, route S and threat T; *Realization*(T) - admissibility of threat realization; *RiskLevel*(T) - risk level of threat T; *SecurityLevel* – general security level of the computer network.

E. ANYTIME APPROACH

To improve the efficiency of construction, modification and analysis of attack graphs the usage of the anytime approach is proposed. The main goal of the anytime approach is to have the result at any time by applying a set of algorithms with different timelines and precision. Summarizing, the anytime algorithms suppose the following peculiarities: a opportunity to obtain the solutions, possibly not precise, as soon as they are needed during the process of solving the problem; a solution found is to be of a sufficient level of adequacy (but it may be either incomplete or approximate); with the lapse of time, the obtained solutions are getting closer to the final result (i.e. improving the precision).

Application of the anytime algorithms for the cyber attack modeling and impact assessment includes the procedures for constructing and analyzing attack trees, including calculating security metrics. Such application enables continuous security monitoring and decision-making.

In the suggested approach the security evaluation is conducted with the use of security metrics, which can be obtained by means of complex analysis including the detection of hosts, network interfaces, operating systems, taking into account different kinds of communication, etc. Such analysis can take a lot of time and hence exact values of the integral metrics (e.g. express evaluation of the protection) can be available in due time.

For security level evaluation by anytime approach, the following groups of algorithms were selected (they are sorted in the order of time expenses and precision increase):

(1) analysis of security level on the base of lists of vulnerabilities detected on hosts without determination of a particular malefactor model (or taking into account a simplest model – a level of complexity of vulnerabilities the malefactor can use) and without considering an attack graph;

(2) construction of an attack graph, where all sub-networks are grouped according to the criticality level, meanwhile the vulnerability groups are computed as a sum of vulnerabilities of individual hosts in them;

(3) construction of an attack graph for the complete network;

(4) dynamic attack simulation as a technique that allows obtaining more precise attack modeling.

Also, in order to construct an anytime algorithm for integral security metrics computation, one can conduct their calculations for some subnet available at the given moment and for hosts for which all needed information is already known or for subnet which is changed.

4. ARCHITECTURE AND IMPLEMENTATION

The task for this stage of research and development was to develop a version of the CAMIAC prototype, which uses the techniques suggested in the paper. In this section we describe the CAMIAC prototype architecture and its current implementation.

The CAMIAC prototype is aimed to demonstrate the approach to modeling attacks at various levels. To do this the prototype has to implement the following functionality:

(1) generation of basic attack trees (not in real-time mode);

(2) construction of the dynamical simulation model (not in real-time mode) imitating stochastically different attacks and countermeasures;

(3) computation of the security level of the network, determining the possible bottlenecks and defining other network security metrics (anytime mode);

(4) updating attack trees taking into account changes in input data (near real-time mode);

(5) detection of scenarios of the current attack (near real-time mode).

The input data for the prototype is as follows: network (system) configuration and the list of hosts' software and hardware; list of existing vulnerabilities from External database (DB) - National Vulnerability Database (NVD); security scanners reports, that can consist information about network configuration and vulnerabilities; real-time events from the external correlation engine.

The output of the prototype consists of a list of security metrics calculated for the network and a recommendation for increasing its security level.

The general architecture of the implemented CAMIAC is shown in Figure 1.

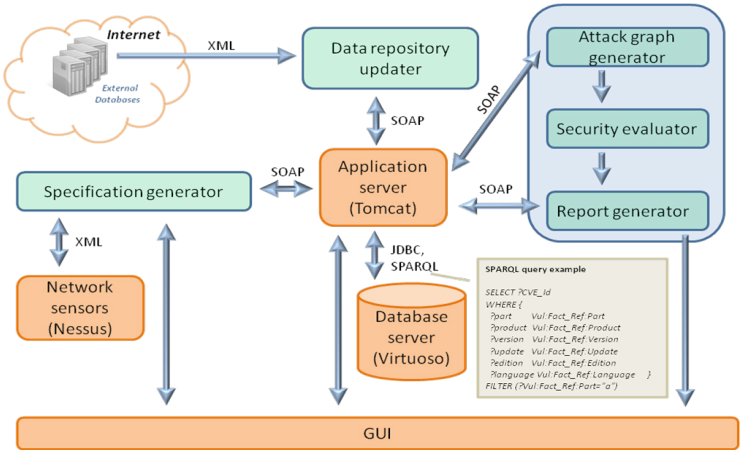

Figure 1. CAMIAC prototype architecture

This figure contains three main components of the CAMIAC: (1) Data repository updater; (2) Specification generator; (3) Attack graph generator and analyzer. Additionally the prototype includes the database and the files for storing the tested network elements and links between them. The first component allows updating the internal database of vulnerabilities, using information obtained from NVD (National Vulnerability Database) [33]. The second one mainly aims to create and modify models of computer networks and helps operator to form all input values like malefactor models and security requirements. The third component demonstrates the suggested approach for attack graph building and analysis.

To organize the fast interaction with the database, the ontology for representation of the CAMIAC data has been developed. The model is based on the SCAP protocol. The approach to the vulnerability presentation for the CAMIAC allows getting significantly lesser amount of data from the database and getting rid of a necessity of program processing, delegating analysis task to a logical reasoning system. The Virtuoso server [47] is used as the storage for ontologies. Interaction with the main CAMIAC module is carried out through the Repository Application Server (RAS), representing web services for demands to the logical reasoning system. The client part is written in Java. When implementing RAS, the Jena framework is used. The logical reasoning system is embedded into the Virtuoso server.

For the interaction between CAMIAC components, SOAP v1.2 [43] is used. The client generates a query in the form of XML document, which is transported using HTTP. Apache Tomcat [2] is chosen as a servlet container as it satisfies all requirements. The web-services are implemented in Java programming language [37]. The service implementation level uses the Hibernate library [36], which supports Java Persistence API v. 2.0 (JPA) [46].

The example of the CAMIAC dashboard showed in Figure 2 is used to setup initial data.

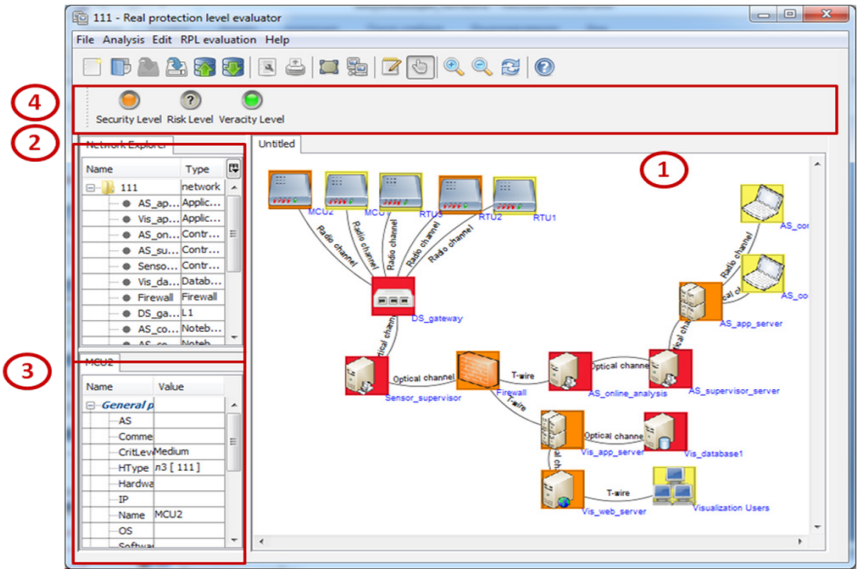

Figure 2. Example of CAMIAC dashboard

This dashboard can be divided into four subviews. The main view 1 shows the topology of the studied network, while view 2 reflects the hierarchical structure of the network, depicting domains or specified networks zones. The graph based techniques are used to represent the network topology. Each network object is represented by an icon. The user has possibility to define icons for each type of the network objects. The background color of the icon is used to encode values of the security metrics calculated for the given host, such as Criticality, Mortality, Risk Level. These metrics are chosen by the user from the predefined list. The brief information about each host is available a via tool tip which appears when mouse hovers over the network object.

The user can configure each host and network using the property view 3. It can specify predefined properties of the host such as IP address, host type (web server,

ftp server, database server, router, firewall, etc.), installed software and hardware, user-defined host criticality. These properties are necessary for attack graph generation. There is also a possibility to define user properties. This property view is updated whenever a particular state node is selected. Thus user always has details at hand.

The view 4 shows the security metrics calculated for the network itself: Security level, Risk Level, and Veracity level. As these metrics can have value from the predefined set of values {Low, Medium, Above Medium, High, Undefined}, they are presented in a form of the semaphore signal. We think that such dashboard design gives a general overview about security analysis of the network and communicate a lot of information in a glance. Thus, the user can analyze calculated host security metrics in the context of initial host configuration; all information is available in different views, but on one dashboard panel.

5. CASE STUDY

We performed the following experiments with the prototype implemented to show the advantages of the proposed CAMIAC framework:

(1) Formation of the attack graph for a computer network;

(2) Evaluation of network security by the attack graph analysis;

(3) Import of the report containing the IDS security events;

(4) Analysis of the security events related to the changes in the source data (changes in the list of installed software, host list and links between hosts) to show the modification of attack graphs;

(5) Analysis of the security events for detection of attack actions in order to recognize possible malefactor models.

The network of a small company was selected for experiments. This network includes the hosts of several types: user computers, a database, a web server and network equipment. For each host in the network the software and hardware were defined.

Several malefactor models were selected:

(1) An *external malefactor with medium knowledge in security area*. His/her knowledge enables usage of the attack action with low and medium complexity. The external type means that he/she is located in the Internet and has access only to the web server. The aim of this model is to collect information about the company

network. This malefactor does not know any zero-day vulnerability and does not have any rights in internal network;

(2) An *external malefactor with high knowledge in security ar*ea. His/her knowledge enables usage of the attack actions with any complexity. The external type means that he/she is located in the Internet and has access only to the web server. The aim of this model is to destroy all information in the internal database of the company. This malefactor knows several zero-day vulnerabilities, but initially does not have any rights in the internal network;

(3) The *internal malefactor with low knowledge in security area*. His/her knowledge limited with the list of attack actions which can be performed by some security tools. The aim of this model is to modify some information in the internal database of the company. This malefactor does not know any zero-day vulnerability, but has initial access to the users' computers in the internal network and has user and remote user rights for several hosts.

To make clearer the illustration of the CAMIAC prototype possibilities, let us consider a case with the following software for network hosts: operating system (OS) Windows Server 2003 is installed on all hosts, DBMS MySQL 5.0 - on the host Database, Apache HTTP Server 1.3.6 - on the Web Server host.

After constructing the attack graph, the CAMIAC provides the following information: the malefactor knowledge after all possible attacks, the attack tree in the graphic form and the log of the malefactor's actions.

For the malefactor 2, the attack graph example is depicted in Figure 3.

The malefactor, carrying out attack actions, is located on the top of the graph. The other icons are as follows: "A" – an attack action, "S" – a scenario which does not use vulnerabilities (for example, host discovery (PING)), "V" – an attack action which exploits some vulnerability.

According to the attack graph the chain of malefactor's actions and their results are as follows:

(1) Detection of nodes connected with the initial malefactor host. Web Server host is detected; (2) Detection of the software installed on the Web Server host. Windows Server 2003 is detected; (3) Usage of the vulnerability CVE-2007-0214. Malefactor compromises of the Control Web Server; (4) Detection of the nodes connected with the Web Server. Application Server host is detected, etc.

Thus, the malefactor starts to perform attack actions from the host "External Web Server Users". This host is a starting point because the malefactor has all privileges

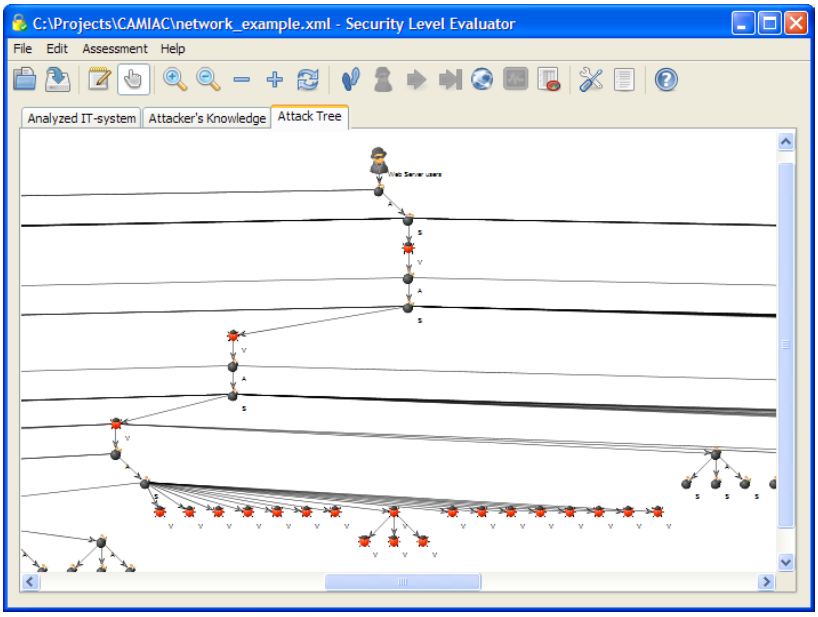

Figure 3. Attack graph example

in the host according to the specified malefactor model. The selected malefactor is an external for the network, and he/she can connect only to the "Web Server".

Firstly the malefactor gathers the information about the host "Web Server" and performs attack actions without any privileges on this host. After several attack actions the malefactor obtains the remote and local users privileges and continues the information gathering. The final step of the malefactor on this host is to obtain administrators privileges. Then the malefactor scans for accessible hosts and starts new attack actions for a new host.

According to the suggested metrics the security level of the tested network is evaluated. For each node the criticality level is determined, for example for the nodes "Web Server Users" and "Application Server" it is LOW while for the node "Firewall" it is HIGH.

There are 12 hosts in the attack graph. For these hosts eight different successful attack actions were discovered and modeled. The attack graph contains 207 different attack routes. These routes contain 31 security violations (confidentiality, integrity and availability) for different host.

For each attacker's action and each possible attack route the security metrics Access Complexity (AC) and Mortality (M) are calculated.

Thus, the Route parameters Access Complexity and Mortality equal LOW (the minimal values for each host in the route). These metrics form the basis for the general network level evaluation. In this use case the Security Level is ORANGE, what means that countermeasures need to be implemented. The weak place in the analyzed network for the selected malefactor model is the host "Web Server" – all 207 routes passed through it. The main recommendation for the system administrator is to increase the protection of the host "Web Server".

The next stage of the experiment is the analysis of security events that are received from the analyzed network. To do this a simple report is created. This report contains two types of the security events – the events describing changes in the network and the events including the recognized attacking actions. The following rows are the examples of the events content:

(1) 192.168.0.107	192.168.0.2	SCAN nmap TCP {tcp}
(2) 192.168.0.2	installed	cpe:/a:mysql:mysql:5.1.33

Event 1 contains information about the detected scanning process. The Database host (192.168.0.2) was scanned by the Nmap tool from some user's host in the network (192.168.0.108). If there are no other attack actions in the security event report, then the CAMIAC make a decision that with high probability the malefactor of type 3 was detected in the network. The reason of this decision is the fact that the malefactors of type 1 and 2 should firstly perform attacks on Web Server, after that they can attack some user host, and only then, they are able to attack database host.

Event 2 specifies new software installed to the host Web Server. It contains the CPE description of the MySQL server (DBMS MySQL 5.1.33 was installed instead of MySQL 5.0). This event leads to the fact that some vulnerabilities that are specific to the previous version of the database may disappear. For instance, this stipulates that the malefactor of type 3 will not be able to modify data in the database, and thus the evaluation of network protection for the malefactor of this type is changed to the GREEN.

6. COMPARISON WITH RELATED COMMERCIAL SYSTEMS

As it was mentioned the number of the implemented security evaluation systems based on attack graph analysis is very small in contrast to amount of the theoretical papers. In this section, several related systems of different classes, which can fulfill security analysis functions, are analyzed: COMNET III [4], OPNET [36], Amenaza SecurITree [42] and Nessus [34]. There are other related systems, but they have

similar disadvantages which we are trying to overcome in the CAMIAC solutions.

Stochastic discrete event simulation systems like COMNET III [4] and OPNET [36] allow creating the detailed model of computer networks. The results of simulation are the evaluation of network protection against a variety of attacks including resource depletion. Disadvantage of these systems is the high resources needed for development. Detailed simulation of the network activity of all services and hosts requires a long time and, therefore, the use of such systems for security analysis is very complicated. In addition, after the changes of network topology and services, it is necessary to fulfill repeated simulation. Thus, taking into account the requirements of operative near real time security analysis, these systems are worse than the CAMIAC by efficiency and resource consumption parameters.

Amenaza SecurITree [42] is an example of commercial software which uses attack trees for security analysis. This tool is designed for attack tree building and analyzing, it has a friendly interface and very detailed documentation. The disadvantage of this system in comparison with the CAMIAC is the lack of possibility to investigate specific malefactors with his/her capabilities and goals. Also there is no support for real-time event analysis in this tool.

Nessus security scanner [34] interacts with the real network and during the scanning cannot penetrate internal network from the external network, if some security system is installed. That is why it usually recognizes only a small number of vulnerabilities. The approach based on malefactor modeling and attack graphs analysis, implemented in the CAMIAC, allows detecting all currently known vulnerabilities in the network, regardless of the original location of the malefactor. Also Nessus can detect changes in the analyzed network, but it requires full rescanning for that.

7. CONCLUSIONS

The paper suggests a framework which allows using attack graphs to evaluate security and provide impact analysis for detection of malefactors and determining the countermeasures in near real time. To achieve this goal the graph generation process is divided on two stages. On the first stage it is suggested to generate the graph of potential attacks for a general malefactor model. This stage should be performed at the time of network deployment or as an offline process, when there are no severe time limitations. On the second stage the attack graph is modified according to the changes in the analyzed computer network. During this stage, modification and analysis of the attack graph should allow to obtain the results in a limited time. The detection of malefactors by their attack action is also performed during the second stage.

This paper gives consideration to the state-of-the-art in attack modeling, the essence of the approach to analytical attack modeling and impact analysis, as well as the architecture of Cyber Attack Modeling and Impact Assessment Component. The techniques suggested are based on the attack graph generation which represents possible attack scenarios taking into account the current security situation, including network configuration, security policy, events and alerts, probable malefactor's location, knowledge level and strategy, known and possible new vulnerabilities.

The developed prototype of the CAMIAC is described. The main difference between the proposed approaches and the existing ones is the possibility of the work in near real-time. Thus, the new results obtained in this investigation are the algorithms and methods of attack graph constructing and analyzing that excel in performance existing ones. All elements of attack modeling, described in the paper, will be extended and detailed in the next steps of research and development.

Acknowledgement

This research is being supported by grant # 13-01-00843 of the Russian Foundation of Basic Research, Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (contract #2.2), State contract #11.519.11.4008 and partly funded by the EU as part of the SecFutur and MASSIF projects.

REFERENCES

[1] A.P. Moore, R.J. Ellison, R.C. Linger. Attack Modeling for Information Security and Survivability. Technical Note CMU/SEI-2001-TN-001. *Survivable Systems*, 2001.

[2] "Apache Tomcat." http://tomcat.apache.org/. [Dec. 28, 2012]

[3] C. Geib, R. Goldman. "A Probabilistic Plan Recognition Algorithm Based on Plan Tree Grammars", *Artificial Intelligence.* vol. 173(11), 2009, pp. 1101-1132.

[4] "CACI Products Company." http://www.caciasl.com/, [Dec. 28, 2012]

[5] "Common Attack Pattern Enumeration and Classification (CAPEC)." http://capec.mitre.org/. [Dec. 28, 2012]

[6] "Common Configuration Enumeration (CCE)." http://cce.mitre.org/. [Dec. 28, 2012].

[7] "Common Platform Enumeration (CPE)." http://cpe.mitre.org/. [Dec. 28, 2012]

[8] "Common Remediation Enumeration (CRE)." http://scap.nist.gov/events/2010/saddw/presentations/remediation.pdf. [Dec. 28, 2012].

[9] "Common Vulnerabilities and Exposures (CVE)." http://cve.mitre.org/. [Dec. 28, 2012]

[10] "Common Weakness Enumeration (CWE)." http://cwe.mitre.org/. [Dec. 28, 2012].

[11] “Common Vulnerability Scoring System (CVSS).” http://www.first.org/cvss/. [Dec. 28, 2012]

[12] E. Bursztein. “Extending Anticipation Games with Location, Penalty and Timeline.” LSV, ENS Cachan, CNRS, INRIA, France, 2008, pp. 272-286.

[13] H.J. Levesque, R. Reiter, Y. Lesperance, F. Lin, R.B. Scherl. “GOLOG: A Logic Programming Language for Dynamic Domains.” *Journal of Logic Programming*, vol. 31, 1997, pp. 59-83.

[14] I. Kotenko, M. Stepashkin. “Attack Graph based Evaluation of Network Security.” *Lecture Notes in Computer Science*, vol. 4237, 2006, pp. 216-227.

[15] I. Kotenko, M. Stepashkin, E. Doynikova. “Security Analysis of Computer-aided Systems taking into account Social Engineering Attacks.”, in *Proc. of PDP 2011*, Los Alamitos, California. IEEE Computer Society, 2011, pp. 611-618.

[16] I. Kotenko, M. Stepashkin. “Network Security Evaluation based on Simulation of Malefactor’s Behavior.” *In Proc. of International Conference on Security and Cryptography (SECRYPT-2006)*, Portugal, 2006, pp. 339-344.

[17] I. Kotenko, A. Chechulin, E. Novikova. “Attack Modelling and Security Evaluation for Security Information and Event Management.” *In Proc. of International Conference on Security and Cryptography. Proceedings (SECRYPT 2012),* 2012, pp. 391-394.

[18] I. Kotenko, A. Chechulin. “Common Framework for Attack Modeling and Security Evaluation in SIEM Systems.” *In Proc. of 2012 IEEE International Conference on Green Computing and Communications, Conference on Internet of Things, and Conference on Cyber, Physical and Social Computing*. Los Alamitos, California. IEEE Computer Society, 2012, pp. 94-101.

[19] I. Kotenko, A. Chechulin. “Attack Modeling and Security Evaluation in SIEM Systems.” *International Transactions on Systems Science and Applications*, vol.8, December 2012, pp.129-147.

[20] J. Dawkins, C. Campbell, J. Hale. “Modeling network attacks: Extending the attack tree paradigm.” In *Proc. of the Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, Johns Hopkins University, 2002.

[21] J. Ryan, D. Ryan. “Performance metrics for information security risk management.” *IEEE Security and Privacy*, vol 6, 2008, pp. 38-44.

[22] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. “Modeling modern network attacks and countermeasures using attack graphs” In *Proc. of the 2009 Annual Computer Security Applications Conference*, Washington, DC, USA, 2009, pp. 117–126.

[23] K. Ingols, M. Chu, R. Lippmann, S. Webster, S. Boyer. “Modeling modern network attacks and countermeasures using attack graphs.” In *Proc of Annual Computer Security Applications Conference (ACSAC ’09)*, Washington, D.C., USA, IEEE Computer Society, 2009, pp. 117-126.

[24] L. Wang, S. Jajodia, A. Singhal, S. Noel. “k-Zero Day Safety: Measuring the Security Risk of Networks against Unknown Attacks.” In *Proc. of ESORICS’10*, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 573-587.

[25] L. Wang, S. Noel, S. Jajodia. “Minimum-cost network hardening using attack graphs.” *Computer Communications*, vol. 29, 2006, pp. 3812-3824.

[26] L. Wang, T. Islam, T. Long, A. Singhal, S. Jajodia. “An attack graph-based probabilistic security metric.” In *Proc. of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security*. Springer-Verlag Berlin, pp. 283-296, 2008.

[27] L. Williams. “GARNET: A Graphical Attack Graph and Reachability Network Evaluation Tool.” In *Proc. of the 5th international workshop on Visualization for Computer Security*, Springer-Verlag Berlin, 2008, pp. 44-59.

[28] M.M. Gamal, D. Hasan, A.F. Hegazy. “A. Security Analysis Framework Powered by an Expert System.” *International Journal of Computer Science and Security*, vol. 4(6), pp.505–526, 2011.

[29] M. McQueen, T. McQueen, W. Boyer, M. Chaffin. “Empirical estimates and observations of 0-day vulnerabilities.” In *Proc. of Hawaii International Conference on System Sciences*, 2009.

[30] M.Y. Huang, T.M. Wicks. “A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis.” *Computer Networks*, vol. 31, New York, NY, USA, pp. 2465-2475, 1999.

[31] N. Kheir, H. Debar, N. Cuppens-Boulahia, F. Cuppens, J. Viinikka. “Cost evaluation for intrusion response using dependency graphs.” In *Proc. of IFIP International Conference on Network and Service Security (N2S)*, IEEE, Paris, France, 2009, pp. 1-6.

[32] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, H. Debar. “A service dependency model for cost-sensitive intrusion response.” In *Proc. of ESORICS 2010*, Athens, Greece, 2010, pp. 626-642.

[33] “National Vulnerability Database (NVD).” http://nvd.nist.gov/. [Dec. 28, 2012]

[34] “Nessus scanner software.” http://www.tenable.com/products/nessus/nessus-product-overview, [Dec. 28, 2012]

[35] O. Sheyner, J. Haines, S. Jha. “Automated generation and analysis of attack graphs.” In *Proc. of IEEE Symposium on Security and Privacy*, Berkeley, California, 2002, pp. 273.

[36] “OPNET Technologies, Inc.” http://www.opnet.com/. [Dec. 28, 2012]

[37] “Oracle Java SE.” http://www.oracle.com/technetwork/java/javase/downloads/index.html. [Dec. 28, 2012]

[38] R.P. Goldman. “A Stochastic Model for Intrusions.” *Lecture Notes in Computer Science*, vol. 2516. Springer Verlag, 2002, pp. 199-218.

[39] “Relational Persistence for Java and .NET.” http://www.hibernate.org/. [Dec. 28, 2012]

[40] S. Hariri, G. Qu, T. Dharmagadda, M. Ramkishore, C.S. Raghavendra. “Impact Analysis of Faults and Attacks in Large-Scale Networks.” *IEEE Security and Privacy*, vol. 1, pp. 49-54, 2003.

[41] S. Noel, S. Jajodia, B. O'Berry, M. Jacobs. "Efficient minimum-cost network hardening via exploit dependency graphs." In *Proc. of ACSAC'03*, 2003, pp. 86-95.

[42] "SecurITree – Attack graph analysis software. Amenaza Technologies Limited." http://www.amenaza.com/. [Dec. 28, 2012]

[43] "SOAP." http://www.w3.org/TR/soap/. [Dec. 28, 2012]

[44] "Symantec Enterprise Security Manager." https://www.symantec.com, [Dec. 28, 2012]

[45] T.L. Amenaza. "Fundamentals of Capabilities-based Attack Tree Analysis." Calgary, Canada, November, 2005.

[46] "The Java Persistence API." http://glassfish.java.net/javaee5/persistence/. [Dec. 28, 2012]

[47] "Virtuoso universal server." http://virtuoso.openlinksw.com/. [Dec. 28, 2012]

[48] W. Lee, X. Qin. "Attack Plan Recognition and Prediction Using Causal Networks." In *Proc. of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, Tucson, Arizona, December, 2004.