PIK, Vol. 35, pp. 25–31 DOI 10.1515/pik-2012-0005

A Framework for Automated Identification of Attack Scenarios on IT Infrastructures

Seyit Ahmet Camtepe

DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin
Germany
camtepe@dai-labor.de

Dr. *Seyit Ahmet Camtepe* received the Ph.D. in computer science from Rensselaer Polytechnic Institute (2007, NY, USA), M.S. and B.S. in computer engineering from Bogazici University (2001 and 1996, Istanbul, Turkey). He worked as a network security engineer at Pamukbank (1996–2002, Istanbul, Turkey). He is currently a post-doctoral researcher at DAI-Labor, Technische Universitaet Berlin, leading a security research group, and seeking his habilitation degree. His research interests include security and privacy in pervasive computing and communication, applied cryptography, security in wireless sensor networks, attack modelling and security simulation.

Karsten Bsufka

DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin
Germany
karsten.bsufka@dai-labor.de

Dr.-Ing. *Karsten Bsufka* studied computer science at Technische Universitaet Berlin and received his Ph.D. degree in 2006. He currently works as a senior researcher at the DAI-Labor and coordinates research projects related to network security, and network simulation. He teaches classes on network simulation and intrusion detection. He was the lead developer for the security mechanisms in the JIAC IV (www.jiac.de) agent framework, and responsible for the Common Criteria evaluation of JIAC IV.

Leonhard Hennig

DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin
Germany
leonhard.hennig@dai-labor.de

Dr.-Ing. *Leonhard Hennig* studied computer science at Technische Universitaet Berlin and received his Ph.D. degree in 2011. He received his Magister Artium degree in computational linguistics/artificial intelligence, computer science, and linguistics from the University of Osnabrück in 2003. His main research interests are natural language processing, text summarization, and information extraction.

Cihan Simsek

Computer Science and Communication
Royal Institute of Technology
SE-100 44 Stockholm
Sweden
cihans@kth.se

Cihan Simsek received the B.S. in computer science & engineering at Sabanci University (2009, Istanbul, Turkey). He is currently seeking his M.Sc. degree in information and communication systems security at Royal Institute of Technology (KTH, Stockholm, Sweden). His research interests include attack modelling, intrusion detection, security risk management and secure software development.

Sahin Albayrak

DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin
Germany
sahin.albayrak@dai-labor.de

Prof. Dr.-Ing. habil. *Sahin Albayrak* holds a professorship as the chair for Agent Technologies in Business Applications & Telecommunication (AOT) at Technische Universitaet Berlin. He is the head of the DAI-Labor. He obtained his Dr.-Ing. degree in 1992, and the habilitation in 2002, both from TU Berlin. He is a member of the steering board of Deutsche Telekom Laboratories (T-Labs), and a member of the research initiative "Vernetzte Welten" of the incentive circle initiated by the German Government for cooperation between industry and academia "Partner für Innovation". His research interests include next generation telecommunication services, applications and network infrastructures. In particular service-centric architectures, service engineering, agent-oriented modelling, agent architectures, agent programming languages, telecommunication services, e-/m-commerce, mobility supporting services, 3G- and beyond-3G services, supply chain management, autonomous security, and smart systems.

Abstract

Due to increased complexity, scale, and functionality of information and telecommunication (IT) infrastructures, every day new exploits and vulnerabilities are discovered. These vulnerabilities are most of the time used by malicious peo-

ple to penetrate these IT infrastructures for mainly disrupting business or stealing intellectual properties. Current incidents prove that it is not sufficient anymore to perform manual security tests of the IT infrastructure based on sporadic security audits. Instead networks should be continuously tested against possible attacks. In this paper we present current results and challenges towards realizing automated and scalable solutions to identify possible attack scenarios in an IT infrastructure. Namely, we define an extensible framework which uses public vulnerability databases to identify probable multi-step attacks in an IT infrastructure, and provide recommendations in the form of patching strategies, topology changes, and configuration updates.

1 Introduction

Increased use of information and telecommunication infrastructures (IT) for monitoring and controlling critical infrastructures of modern societies mandates us to devote special attention on their security. Due to increased complexity, scale, and functionality, new vulnerabilities are discovered almost every day. These vulnerabilities are exploited by malicious people to penetrate the IT infrastructures for mainly disrupting business or stealing intellectual properties. This problem is intensified by modern attacks on the IT infrastructures, as represented by Stuxnet which combines several mechanisms to exploit vulnerabilities in a range of operating systems including industrial infrastructures [1], [2]. Stuxnet employs several steps in order to infiltrate parts of the networked infrastructure and in doing so gives way for new attack scenarios that previously might have been considered unlikely.

Cyber espionage and cyber warfare are considered to be two of the most important threats for IT infrastructures. Cyber espionage involves theft of intellectual properties and knowledge of a company or an institute. Attackers make use of vulnerabilities in an IT infrastructure to transfer large scale datasets to another location within a short time window without being detected [3]. Cyber warfare is another important threat mostly targeting critical infrastructures of modern societies. Everyday more such systems get interconnected. This trend changes the characteristics of military offences between countries or terrorist activities as in the example of the denial of service attack on Estonia [4].

The main challenge in analysis and identification of these threats is the asynchronous nature of the problem. For an attacker, it is sufficient to find a low cost attack scenario, while the defender has to identify all probable attack scenarios. The latter requires continuous analyses for vulnerabilities on all network nodes for the whole application inventory. Network security testing should not interrupt the daily business activities of an organization. Therefore it should be done in a virtual environment with simulation and distributed computation capabilities. Another drawback is the dimension of the problem which can be quite large due to a large number of vulnerabilities and exploits, firewall rules, address translations, proxies, interdependencies, intrusion detection rules, and the behavior of the clients. Even in a small scale networks with tens of nodes, the problem can become quite unmanageable. Additionally, standard vulnerability scanners are unable to determine all possible multi-step attacks, as they only provide analysis on a per node basis. Under these circumstances it is quite challenging to come up with an attack scenario that can test the security of the target network thoroughly. Moreover, the concurrent patching of all related vulnerabilities may not be feasible due to unavailability of patches, or due to application dependencies and operational constraints.

Due to the complexity, scale, and dynamic nature of the problem, it is not sufficient anymore to perform manual security analysis of an IT infrastructure based on sporadic security audits. Instead networks should be continuously tested against possible multi-step attacks. Automated and scalable security test solutions need to be developed so that security administrators can be warned against possible penetration points along with the associated risk values in their infrastructures. In addition, they can be supported with recommendations to mitigate these threats. Recommendations, such as patching strategies, topology changes, and configuration updates, can be useful to reduce IT related risks in the infrastructure.

There exist various activities [6], [7], [8] which address subsets of these issues, but, up to our knowledge, there is no unified open source solution. Existing solutions usually address a specific problem, such as vulnerability scanning, patch management, and IT security management. Moreover, existing vulnerability assessment tools (such as nessus, saint, openVAS, retine, GFI Languard) or GRCM (governance, risk and compliance management) are only useful for manual and sporadic security audits, and cannot analyze and identify multi-step attacks in large scale IT infrastructures.

In this paper we outline an open source software framework for automated security testing by using public vulnerability and exploit databases. Our framework provides unified testing opportunities with modules for information extraction, attack analysis, risk management, security recommendation, and security simulation. The framework is based on a multi-agent system. Therefore it can scale and permits the integration of new functionalities and data sources. We envision that the open source framework will be used by IT managers without releasing valuable information to external parties. Security audit companies usually scan an IT infrastructure once with very limited knowledge due to privacy and on a per node bases. Therefore they may not test the infrastructure thoroughly and may fail to cover all possible penetration points. Our unified solution can be used by IT specialists within and under control of the organization itself. It can provide a secure distributed platform to collaboratively analyze interdependent infrastructures belonging different administrative domains. As the result, the security of the IT infrastructure will be scored; identified weaknesses will be reported, and recommendations in terms of patching, configuration, and topology changes will be generated. These recommendations will be tested in the simulation component over up-to-date network and services settings, and finally mitigation steps may be taken to remove the important weaknesses.

Organization of the paper is as follows: in Section 2, we introduce our framework and describe its components. In Sections 3 to 8, we review current research and development

results to realize these components, and identify open problems to be addressed. Some final remarks in Section 9 conclude the paper.

2 Automated IT Security Testing Framework and Its Components

We propose a scalable and extensible framework interconnected with real network where attack analysis, risk analysis, and testing of mitigation strategies can be performed continuously. Figure 1 depicts our framework, its components, and their relations. Our automated security testing framework contains six basic components.

Our view of IT security test consists of identifying possible attack paths reflecting probable multi-step attack vectors within an IT infrastructure. This can be achieved with the help of an ontological approach reflecting device vulnerabilities, possible exploits, and the associated multi-step attacks. The **knowledge** component includes ontologies tailored for the vulnerability and threat domain. Knowledge repositories constructed using these ontologies can be populated by using the **information extracted** and enriched from public free-text vulnerability and threat databases as well as from data sources of target IT infrastructures, such as software inventory, topology, configuration, security policies, and firewall rules. Vulnerability and threat databases contain large number of unstructured free-text entries (NVD – National Vulnerability Databases, OSVD – Open Source Vulnerability Database, CVE – Common Vulnerabilities and Exposures, CPE – Common Platform Enumeration, CWE – Common Weakness Enumeration, etc.)[1] which usually need to be processed manually by security experts. Automatic or semi-automatic information extraction methods are required for populating ontologies designed for vulnerabilities and exploits.

With our framework, we would like to associate existing vulnerabilities and exploits with the network elements, servers, clients, applications, and services. Then, we will identify multi-step attacks by using distributed analysis techniques to decide IP, transport, or application level connectivity through gateways with firewall, and proxy capabilities. Scalability is one of the key points, since checking 45,000 vulnerabilities along with hundreds of firewall and IDS rules for even small scale networks are infeasible with existing approaches. The multi-step **attack analysis** component includes distributed accessibility analysis, multi-step attack analysis, and risk assessment functionalities. This component associates vulnerability knowledge to IT infrastructure elements based on their properties and capabilities (operating system, protocols, applications, services, patch levels, etc.). Distributed accessibility analysis processes firewall, proxy, access filter, address translation, and virtual private network rules and policies to decide which compromised node permits access to which other nodes. Multi-step attack analysis uses accessibility information to enumerate attack paths and reachable attack targets and sources. This information together with IT infrastructure specific knowledge is

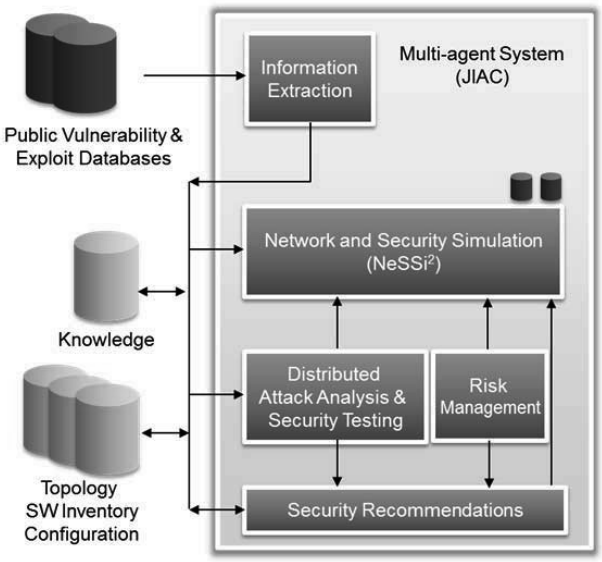

Figure 1 Automated IT Security testing framework.

used in **risk analysis** to assess the risks to critical IT elements and services.

The **recommendation** component generates best mitigation strategies under user specified constraints with respect to the placement of firewall and intrusion detection devices, patching plan, configuration change, software update, and topology change.

Assessment of the risk and impact of attack vectors by IT managers can be achieved by a novel simulation for security and networking capabilities. Our open source simulation framework (*NeSSi*^{2})[2] [9] operating on JIAC[3] [5] can handle large-scale network topologies, services, and vulnerabilities. The **simulation** component based on NeSSi^2 helps assess the impact of multi-step attacks and their mitigation strategies over the target IT infrastructure and its services. It includes methodologies for risk assessment, visualization, and what-if analysis.

The use of distributed algorithms for attack analysis within our open source multi-agent system JIAC ensures scalability. The **multi-agent** system is built upon the JIAC framework which provides scalability, and seamless service deployment and modification. It also enables IT security test scenarios which involve inter-dependent IT infrastructures which are operated by different administrative domains.

In the following sections we review and compare existing results, and identify open problems towards realizing these components.

3 Vulnerability Databases

Publicly available vulnerability databases (such as NVD, OSVD, CVE, CPE, CWE) are for human consumption, and

[1] nvd.nist.gov, osvdb.org, cve.mitre.org, cpe.mitre.org, cwe.mitre.org

[2] Network Security Simulator (*NeSSi*^{2}) – www.nessi2.de

[3] Java-based Intelligent Agent Componentware (JIAC) – www.jiac.de

they are unstructured. Thus, the available information is unsuitable for direct machine interpretation, and it creates a challenge for automatic security testing.

The Open Vulnerability and Assessment Language (OVAL[4]) is a community standard which partially addresses the non-machine readable nature of the current vulnerability databases. It defines a machine readable language using XML. One problem with OVAL is that the test results only indicate the presence of a vulnerability, not its exploitability. To successfully exploit a vulnerability a set of pre-conditions has to be fulfilled in the attacked IT infrastructure. Examples of these pre-conditions can be a particular software, the network connectivity, and the attacker's privilege level. After successful exploitation of the vulnerability, the network is exposed to a new set of conditions, called post-conditions, which can help an attacker to launch new attacks against the system. These post-conditions, as pointed out by Maggi et al. [10], have a higher layer of abstraction than pre-conditions and can be categorized as gained access (e.g. guest, user, root, etc.), read, write, execute, corrupt, exhaust, create, delete, crash, reboot, disrupt, and deny privileges. The partial pre- and post-conditions of vulnerabilities are given as human readable text in the title or description sections of an OVAL definition. We focus here on the mapping of this broad range of post-conditions onto pre-conditions of other vulnerabilities to model the chains of vulnerabilities and individual attacks which may result in a multi-step attack.

4 Information Extraction

Information extraction (IE) deals with the automatic extraction of semantically meaningful units of information from unstructured texts [29]. Two important subtasks of information extraction systems are entity and attribute extraction (e.g. software libraries, hardware products, services, version number, and patch level in the security context), and relation extraction, which is concerned with identifying relations between entities [28]. Storing and mapping extracted information to a knowledge base needs to consider the consistency and integrity of the collected information [30]. Challenges arise also from the variability of natural language with respect to the synonymy and ambiguity of words.

Information extraction methods can be applied to extract meaningful pre- and post-conditions from the free-text descriptions available in vulnerability and threat databases. A cursory analysis of these descriptions suggests that they follow semi-regularized patterns (vulnerability [X] in software [Y] allows [remote | local] attacker to [gain | execute | ...] ...) which can be learned, e.g. by applying a Conditional Random Field approach [31]. In this way, entities and their relations can be extracted and mapped onto instances of pre- and post-conditions predefined in a knowledge base. The application of word and entity disambiguation methods [30], [32] will ensure that natural language variation is adequately handled. It will also ensure that novel concepts can be identified and used for ontology extension.

[4] oval.mitre.org

5 Attack Modeling and Analysis

A significant part of a network security testing comprise defining the attack vectors which simply provide ordered lists of vulnerabilities and associated individual attacks. Attack modeling is an efficient way of representing the attack information in a structured and reusable form. These structures can be used in an automated solution to check security of an IT infrastructure.

The **attack tree** is a formal concept introduced in [11] to model the threats to a system in a structured tree representation given with an attacker's goal. In an attack tree the root node of the tree is the attacker's overall goal and all the leaf nodes below any parent node represent the ways the attacker can use to accomplish the parent goal. The parent nodes may have single or multiple leaf nodes. If the former is the case then the achievement of the leaf node directly implies the achievement of the parent node; if the latter is the case the achievement of the parent node depends on the result of the logical operation (i.e. AND/OR) bounded with the leaf nodes. After having logically constructed the attack tree, each node can be assigned a continuous or a Boolean value. Attack trees with the assigned attributes can be converted to machine readable languages like XML and can be fed to several analysis tools [12], [13], [14]. However, there is no standard way of creating and storing these attack trees today. The creation of the attack trees heavily depends on the domain knowledge of the security experts.

A **Petri net** is a bipartite graph consisting of places, transitions, and arcs in which places represent a state of the modeled system, transitions correspond to changes, and arcs determine the direction of the changes. Each place contains a set of tokens which moves among places when a transition state fires [15], [17]. An attack net is a disjunctive Petri net in which places represent the security states of a system and transitions correspond to attacker's actions to compromise the system. The place of the tokens in the attack net shows the compromised set of places by the attacker. The advantage of using attack nets for attack modeling is the ability to conveniently represent the attack concurrency and progress with the movement of tokens, attacker's actions as transitions, and the intermediate or final goals as places. Petri nets cannot scale well and building a well formed definite logic program with Petri nets has complexity of $O(M^3N)$ where M is the number of places and N is the number of transitions [16].

Model Checking is a validation method to check whether a model M is consistent with the given property p or not. The model checker performs an exhaustive search on the model to find a counter-example which violates the given property. Sheyner et al. [18] used model checking to generate attack graphs. In their approach the network is modeled as a finite state machine and each transition corresponds to an attacker's action. The security policy of the system is defined as a property and the violation of the defined policy implies the existence of an attack path. The model checker tool NuSMV[5] which stores each transition and state as a binary expression using Binary Decision Diagrams (BDD) is utilized to compute the attack graph. The severe drawback

[5] http://nusmv.fbk.eu/

of the model checking approach is the state explosion problem which makes it infeasible to use for large size enterprise networks. Host-centric model checking is introduced to overcome the state explosion problem [19]. By using the assumption that an attacker will never relinquish a state, attack graph generation is achieved in quadratic polynomial time.

MIT-Lincoln-Lab showed in [20], [21] that the full attack graph does not scale well. The number of the states in a full attack graph grows up to $O(N!)$. They have suggested two attack graphs namely *Predictive Graph* and *Multiple-Prerequisite* graph. The main idea behind these two attack graphs is to eliminate the redundant nodes. The assumption is that the attacker will exploit the vulnerability as long as it reaches to the vulnerable machine. The only two preconditions of a vulnerability considered in their approach is the locality (remote or local) of the attacker and the connectivity. The post-condition of vulnerability is categorized by four access levels: user, administrative, DOS, and other.

In the **predictive graph** a child node adds a vulnerability to the graph only if none of its ancestors already used the vulnerability to reach the state before. The predictive graph scales with $O(N^2 \log N)$ in a network with no filtering devices (e.g. firewall, router) and N hosts.

The **multiple-prerequisite (MP) graph** introduces three different node types which represent the attacker's access level, pre-conditions, and the vulnerabilities. The arcs have no content; they are simply associations between nodes. The three different arcs must point either from a state node to a condition, from a condition to a vulnerability instance, and from a vulnerability instance to a state node. Ingols et al. in [20], [21] suggested the use of reachability groups instead of a reachability matrix. There is no need to compute the reachability of the nodes in the same subnet, since there are no filtering devices between them. Therefore the reachability matrix for a single subnet can be collapsed into a single group. A reachability group dramatically reduces the time and memory needed to construct reachability information. The resulting multi-prerequisite graph is also further simplified. The state nodes whose pre- and post- requisites are identical are collapsed into a single state group. Then the vulnerability instances, which have identical pre-requisites, and the collapsed states forms a single vulnerability group. Given N nodes and E edges, the complexity of proposed graph simplification algorithm becomes $O(E + N \log N)$.

Client side vulnerabilities can be addressed by backwards reachability analysis. The assumption is that if there is a reverse path from the client side to the attack source then the client side vulnerability will be exploited. The modeling of host firewalls imposes a heavy burden on the reachability computation. This can be resolved by grouping the host firewalls which have similar rule sets.

Logic-based attack modeling reveals the causal relationship between attack steps. Ou et al.'s approach for a scalable attack generation is inspired by an early model checking of the attack graph [25]. The entire network state is represented as a Boolean expression in each node. Each network configuration and attacker's privileges are represented as a propositional formula. The technique is goal-oriented, i.e., the user has to specify a goal (e.g., *execCode*(*attacker,workStation, root*)) for the reasoning engine to generate the attack graph realizing the specified goal. Given a network with N machines the complexity of logic based attack graph is between $O(N^2)$ and $O(N^3)$.

6 Risk Management ans Security Recommendations

Ingols et al. [20], [21] suggest a pretty straightforward recommendation algorithm in which they try to identify the vulnerabilities which lead to the largest number of host compromises. They identify their weighting metrics as the ratio of the non-compromised hosts over all compromised hosts after deployment of a patch. The attack graph is generated from scratch each time a recommendation is tested. In addition, their recommendation algorithms do not take into account the organizational requirements, such as dependency between the services or the economical values of these services. They consider a worst case exploitation assumption and do not consider the CVSS[6] impact values or exploitability metrics, such as the likelihood or complexity of an attack.

Mehta et al. [26] use two different algorithms to rank and measure the probability of the existence of an attacker in each state over the attack graph. They adopt Google's PageRank algorithm and apply it with some minor changes. Sawilla et al. [27] applied Google's PageRank algorithm to identify most realistic and easiest attack vectors. The authors used a variety of weighting metrics to measure the criticality of attacks by considering the likelihood and complexity of an attack, the availability of an exploit, and the relative significance of a network asset. The former two metrics are taken from publicly available CVSS database, but the latter varies from one organization to another and the network administrators have to assign asset values.

The objective of the recommendation algorithms is to combat with the identified attack scenarios in the most cost–effective manner. For this purpose, complex dependencies between assets and processes in an IT infrastructure should be taken into consideration. We adapt a business process-oriented approach where risk assessment, vulnerability assessment, and risk mitigation are integrated into a quantitative framework [22]. By using the asset dependencies, business process values can be mapped onto IT hardware components in a hierarchical manner. These mappings should also consider the problem of *uncertainty* which means that there will never be perfect knowledge about the IT infrastructure, since neither inventory systems nor tools can reliably provide the current state. These mappings can be combined with the IT system vulnerability and attack analysis to derive risk scores on a network node level.

7 Security Simulation

Existing tools that support security evaluation with simulation are rare and usually focused on specific problems. Wei et al. [23] focus on epidemiological models of worm prop-

[6] http://www.first.org/cvss/

agation and developed for this a distributed simulation tool. Another example is Liljenstam et al. [24], they describe in their work the simulation tool RINSE. The focus of their tool is to create distributed simulation environment that is capable to interact with human during a simulation and allow them to issue commands. The intended use of the tool is security related education and security related exercises, like LÜKEX 2011. During this nation-wide crisis management exercise governmental organizations will test how well they are prepared against large-scale cyber-attacks against national IT infrastructures. Exercises like LÜKEX[7] 2011 are time consuming and costly. In the end they only reflect the state on certain point in time.

In our approach a data gathering agent exists that creates a model of the IT infrastructure to be tested. This agent combines several components for the actual data gathering. These components include: network scanners, database connectors, and correlation components. Network scanners actively try to collect information about the network topology, e.g. by wrapping *nmap*[8].Database connectors can connect to an inventory system of a company and retrieve the stored information. Finally the correlation component combines output and creates the topology. This can then be modified by a system administrator to resolve conflicts or to add missing components.

Based on the information about the network topology, the applications on network nodes, and the data available from vulnerability database, it is possible to construct a description of possible attack vectors on assets of the target organization. This is similar to work described by Foo et al. [7]. In contrast, it is not used for intrusion response purposes in running systems, but for calculating risks for the organization, planning and performing security tests, and evaluating different protection approaches, either preventive or responsive.

8 Multi-Agent Systems

Our approach is based on a multi-agent system. The main reason for this is the distributed nature of the problem and the requirement of continuous autonomous operation. Basically, our system will consist of multiple agents for the steps in an autonomous control loop (see, for example, Huebscher and McCann [33]). These agents all have their specific tasks and goals and contribute to the overall goal of continuous security testing and reporting. One main concern in developing the multi-agent system are the security requirements of the system itself, since it requires sharing of sensitive data between agents running in a distributed environment which may be controlled by different organization units. In this context sensitive data are any data that fall in the area of regulated by privacy laws, organizational confidentiality, and secrecy requirements. This can be addressed by two approaches. First, by using distributed algorithms that do not require sensitive data, e.g. during the risk analysis only values of assets need to be communicated but no information that could be used to identify assets. Alternatively, data collection agents preprocess data before sharing. This preprocessing makes the data anonymous or pseudonymous. Anonymization guarantees a higher degree of privacy and secrecy, but pseudomization is more useful in case of incident handling, since authorized stuff can recover the original values.

[7] LÜKEX 2011 http://www.denis.bund.de/luekex/

[8] http://nmap.org/

9 Conclusion

In this paper we presented an open source software framework for automated identification of possible attack scenarios in IT infrastructures by using public vulnerability and exploit databases. We are planning to exploit our current results in the multi-agent system (JIAC), for security simulation (NeSSi[2]), risk management, attack analysis, and intrusion response. Based on our literature review, further activities are required in research points: (i) automatic and semi-automatic information extraction methods, (ii) distributed approaches for attack analysis for complex and large scale IT infrastructures, (iii) quantitative risk analysis, (iv) recommendations based on assets and processes in IT infrastructure, (v) security simulation for analysis of different mitigation scenarios, and (vi) usability, scalability and extensibility of the framework.

References

[1] D.P. Fidler, Was stuxnet an act of war? Decoding a cyberattack, IEEE Security & Privacy 9 (4): 56–59, 2011.

[2] R. Langner, Stuxnet: dissecting a cyberwarfare weapon, IEEE Security & Privacy 9 (3): 49–51, 2011.

[3] R. Deibert and R. Rohozinski, Shadows in the cloud - investigating cyber espionage 2.0, Information Warfare Monitor Technical Report, vol. 3, pp. 1–58, 2010.

[4] M. Lesk, The new front line: estonia under cyberassault, IEEE Security & Privacy 5 (4): 76–79, 2007.

[5] S. Fricke, K. Bsufka, J. Keiser, T. Schmidt, R. Sesseler, and S. Albayrak, Agent-based telematic services and telecom applications: A toolkit for realizing development, deployment, and management of agent-based systems and services, Communications of the ACM 44 (4): 43 – 48, 2001.

[6] A. Herzog, N. Shahmehri, and C. Duma, An ontology of information security, Techniques and Applications for Advanced Information Privacy and Security: Emerging Organizational, Ethical, and Human Issues, pp. 278–301, 2009.

[7] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, and E.H. Spafford, ADEPTS: Adaptive intrusion response using attack graphs in an e-commerce environment, IEEE International Conference on Dependable Systems and Networks, pp. 508–517, 2005.

[8] N. Cuppens-Boulahia, F. Cuppens, F. Autrel, and H. Debar, An ontology-based approach to react to network

attacks, International Journal of Information and Computer Security 3 (3): 280–305, 2009.

[9] S. Schmidt, R. Bye, J. Chinnow, K. Bsufka, S.A. Camtepe and S. Albayrak, Application-level simulation for network security, SIMULATION 86 (5–6): 311–330, 2010.

[10] P. Maggi, D. Pozza, R. Sisto, Vulnerability modeling for the analysis of network attacks, Dependability of Computer Systems, pp. 15–22, 2008.

[11] B. Schneier, Modeling Security Threats, in Dr. Dobbs Journal of Software Tools 24 (12): 21, 1999.

[12] S.A. Camtepe and B. Yener, Modeling and detection of complex attacks, SecureComm, pp. 234–243, 2007.

[13] R.C. Linger and A.P. Moore, Foundations for survivable system development: service traces, intrusion traces, and evaluation models, Technical Report CMU/SEI-2001-TR-029, 2001.

[14] G.C. Dalton, R.F. Mills, J.M. Colombi, and R.A. Raines, Analyzing attack trees using generalized stochastic petri nets, IEEE Information Assurance Workshop, pp. 116–123, 2006.

[15] J.P. McDermott, Attack net penetration testing, ACM Workshop on New Security Paradigms, pp.15–21, 2000.

[16] T. Shimura, J. Lobo, and T. Murata, A Petri net semantics for logic programs with negation, Int. Conf. on Software Engineering and Knowledge Engineering, pp. 292–299, 1992.

[17] M. Meier, S. Schmerl, and H. König, Improving the efficiency of misuse detection, DIMVA, pp. 188–20, 2005.

[18] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing, Automated generation and analysis of attack graphs, Security and Privacy, pp. 273–284, 2002.

[19] R. Hewett, and P. Kijsanayothin, Host-centric model checking for network vulnerability analysis, Computer Security Applications Conference, pp. 225–234, 2008.

[20] K. Ingols, R. Lippmann, and K. Piwowarski, Practical attack graph generation for network defense, Computer Security Applications Conference, pp. 121–130, 2006.

[21] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, Modeling modern network attacks and countermeasures using attack graphs, Computer Security Applications Conference, pp. 117–126, 2009.

[22] S. Schmidt, and S. Albayrak, A quantitative framework for dependency-aware organizational IT risk management, International Conference on Intelligent System Design and Applications, pp. 1207 - 1212, 2010.

[23] S.Wei, J. Mirkovic, and M. Swany, Distributed worm simulation with a realistic internet model, Workshop on Principles of Advanced and Distributed Simulation, pp. 71–79, 2005.

[24] M. Liljenstam, J. Liu, D.M. Nicol, Y. Yuan, G. Yan, and C. Grier, RINSE: The real-time immersive network simulation environment for network security exercises SIMULATION 82 (1): 43–59, 2006.

[25] X. Ou, W.F. Boyer, and M.A. McQueen, A scalable approach to attack graph generation, ACM Conference on Computer and Communications Security, pp. 336–345, 2006.

[26] V. Mehta, C. Bartzis, H. Zhu, E.M. Clarke, and J.M. Wing, Ranking attack graphs, RAID, pp. 127–144, 2006.

[27] R.E. Sawilla, and X. Ou, Identifying critical attack assets in dependency attack graphs, ESORICS, pp.18–34, 2008.

[28] M. Banko and O. Etzioni, The tradeoffs between open and traditional relation extraction, Annual Meeting of the Association for Computational Linguistics, pp. 28–36, 2008.

[29] C.-H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan, A survey of web information extraction systems, IEEE Transactions on Knowledge and Data Engineering 18(10): 1411–1428, 2006.

[30] P. McNamee, H.T. Dang, H. Simpson, P. Schone, and S.M. Strassel. An evaluation of technologies for knowledge base population, International Confenference on Language Resources and Evaluation, pp. 369–372, 2010.

[31] F. Peng, and A. McCallum, Information extraction from research papers using conditional random fields, Information Processing & Management 42 (4): 963–979, 2006.

[32] R. Navigli, Word sense disambiguation: A survey, ACM Computing Surveys 41 (2): 1–69, 2009.

[33] M. Huebscher and J. McCann, A survey of autonomic computing degrees, models, and applications, ACM Computing Surveys 40 (3): 7:1–7:28, 2008.