Mathematical Problems of Computer Science 40, 85--95, 2013.

Comparative Analysis of Attack Graphs

Levon H. Aslanyan, Daryoush Alipour and Minoosh Heidari

Institute for Informatics and Automation Problems of NAS RA
e-mail: lasl@sci.am

Abstract

It is well-known that nowadays computers and networks that are unique in their computational and service provision power have also major weaknesses and vulnerabilities that can be exploited by outsiders in compromising the valuable data and knowledge. Network administrators and network security analysts must be aware of different properties of current software solutions and diversity of problems regarding the possible protection of network assets. This means that they must know and use the latest and newest types of vulnerabilities, techniques and tools. "Attack Graphs" present formalized network maps and help with analysis of possible vulnerabilities that may exist in the network. Hence, in this paper we will describe some basic concepts that can be used to understand and generate the attack graphs.

Keywords: Network security, Network vulnerability, Attack graph.

1. Introduction

Defending large scale networks is very difficult. The outside interest to information, conflicting relations and business objectives draw to special type of activities compiled round the term ***hacker***. Many of the applied systems are created just to provide the necessary work with information. Protecting the system becomes an additional burden. A defender in such a situation must be able to locate all paths into the network and prevent attackers from using them at the moment when an attacker needs to find only one unprotected path. A network defender has the advantage of intimate knowledge of the network such as: the ways traffic may move through it, the services running on it, and the vulnerabilities in those services. A defender can use that knowledge to improve situational awareness.

Attack graphs are one way to leverage those data. There are many different papers on attack graphs and many representations, but the core idea remains the same: an attack graph shows the ways an attacker can compromise a network or host. Defenders can then use the attack graph to identify critical bottlenecks and work to secure those bottleneck hosts and services first.

Attack graphs are a valuable tool to network defenders, illustrating paths an attacker can use to gain access to a targeted network. Defenders can focus their efforts on patching the vulnerabilities and configuration errors that allow the attackers to have the greatest amount of access [1].

The remainder of this paper is organized as follows: Section 2 provides an overview of vulnerability, Section 3 describes the attack graph, and Section 4 draws analysis and conclusions.

2. Vulnerability

In computer security, vulnerability is a flaw or weakness in a network that can be exploited by one or more threats to violate the system's security policy. Network vulnerabilities have the potential of being exploited in a way that may lead to the use of the computer to achieve the intruder's desired goal. Hence, exploits give an attacker to take advantage of a flaw, action or vulnerability in a network.

2.1 Terms and Standards for Information Security Vulnerability Domain

There are some best practices and standards to classify vulnerabilities that have been discussed in this section.

2.1.1 CVE Names

Common Vulnerabilities and Exposures (CVE®) is a dictionary of common names (i.e. CVE Identifiers) for publicly known information security vulnerabilities. CVE is an international information security community effort and it is now the industry standard for vulnerability and exposure names [2].

2.1.2 OSVDB References

OSVDB is an independent and open sourced web-based vulnerability database created for the security community. Common Vulnerabilities and Exposures (CVE) simply provides a standardized name for vulnerabilities, much like a dictionary. OSVDB is a database that provides a wealth of information about each vulnerability. Where appropriate, entries in the OSVDB refer to their respective CVE names. In addition, over the past 8 years, OSVDB has imported over 23,000 vulnerabilities that cannot be found in CVE [3].

2.1.3 CVSS Scoring

CVSS stands for The Common Vulnerability Scoring System and is a vendor agnostic, industry open standard designed to convey vulnerability severity and help determine urgency and priority of response. It solves the problem of multiple, incompatible scoring systems and is usable and understandable by anyone. CVSS is a vulnerability scoring system designed to provide an open and standardized method for rating IT vulnerabilities. CVSS helps organizations prioritize and coordinate a joint response to security vulnerabilities by communicating the base, temporal and environmental properties of a vulnerability. FIRST (the Forum of Incident Response and Security Teams) hosts a special interest group to update and promote CVSS and provides a central repository for CVSS documentation [4].

2.1.4 NVD

NVD is the U.S. government repository of standards based on vulnerability management data represented with the use of the Security Content Automation Protocol (SCAP). These data enable automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics [5].

2.2 Vulnerability and Exposure

A vulnerability is a state in a computing system (or set of systems) that allows an attacker to execute commands as another user, to conduct a denial of service, and so on.

An information security "vulnerability" is a mistake in software that can be directly used by a hacker to gain access to a system or network. An information security "exposure" is a system configuration issue or a mistake in software that allows access to information or capabilities that can be used by a hacker as a stepping-stone into a system or network.

An "exposure" describes a state in a computing system (or set of systems) that is not a vulnerability, but allows an attacker to conduct information gathering activities, to hide activities, and so on.

2.3 Vulnerability and Exposure Examples

Examples of vulnerabilities include:

- •phf (remote command execution as user "nobody")
- •rpc.ttdbserverd (remote command execution as root)
- •world-writeable password file (modification of system-critical data)
- •default password (remote command execution or other access)
- •denial of service problems that allows an attacker to cause a Blue Screen of Death
- •smurf (denial of service by flooding a network)

Examples of exposures include:

- •running services such as finger (useful for information gathering, though it works as advertised)
- •inappropriate settings for Windows NT auditing policies (where "inappropriate" is enterprise-specific)
- •running services that are common attack points (e.g., HTTP, FTP, or SMTP)
- •use of applications or services that can be successfully attacked by brute force methods (e.g. use of trivially broken encryption, or a small key space).

2.4 Vulnerability Scanners

A vulnerability scanner is a software designed to assess computers and networks for weaknesses and vulnerabilities.

Vulnerability scanners are divided into two groups: network-based scanners and host-based scanners.

A network-based scanner is installed on a computer that scans a number of other hosts on the network, such as: Port Scanners(Nmap, Nessus), Web application security scanner, Network vulnerability scanner (BoomScan).

A host-based scanner is installed in the host, such as: Database Security Scanner.

3. Attack Graph

Attack graph ([6--8], [11]) is an integral part of modeling the overview of network security. System administrators use attack graphs to determine how vulnerable their systems are and to determine what security measures to deploy to defend their systems [9].

Each attack graph shows a set of scenarios of penetrating a computer network. A penetration scenario actually defines the order of steps that an intruder should take to achieve his goal, and each step is characterized to show which host must be abused [10].

Major discussions in this area are generating an attack graph, and analyzing it for intrusion detection and hardening the system-critical.

3.1 A Simple Example

Consider the example network shown in Figure 1. There are two target hosts, Machine 1 and Machine 2, and a firewall separating them from the rest of the Internet. As shown, each host is

running two of three possible services (ftp, sshd, a database). There are four possible atomic attacks, identified numerically as follows: (0) sshd buffer overflow, (1) ftp .rhosts, (2) remote login, and (3) local buffer overflow. The ftp .rhosts attack needs to find the target host with two vulnerabilities: a writable home directory and an executable command shell are assigned to the ftp user name. The local buffer overflow exploits a vulnerable version of the xterm executable.

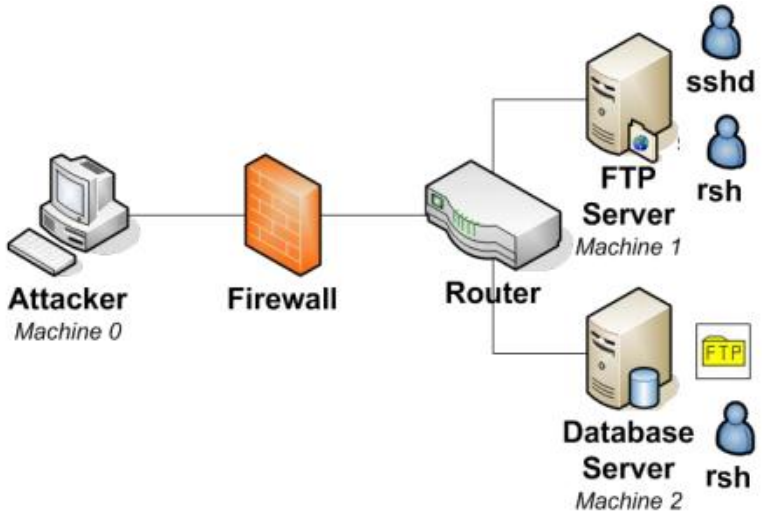

Fig. 1. Example Network.

Service	Description	Common Exploits
sshd	sshd (Secure Shell Daemon) is the daemon program for ssh. **sshd** listens for connections from clients. 'ssh' client and 'sshd' server, provide secure encrypted communications between two untrusted hosts over an insecure network.	**sshd buffer overflow (sshd_bof)** It gives a remote user a root shell on the target machine.
FTP Server	File Transfer Protocol is a standard network protocol used to transfer files from one host to another one. An FTP server is a software application on networks that provides lots of software to download.	**Ftp remote host (ftp-rhosts)** Using an ftp vulnerability, the intruder creates an .rhosts file in the ftp home directory and takes a remote login trust between his machine and the target one.
rsh	The remote shell (rsh) is a command line computer program. 'rsh' like 'ssh' can execute commands on remote systems.	**The remote shell (rsh)** The intruders log in from one machine to another, using an existing remote login trust between two hosts, and gets a user shell without password.
Database Server	A database server is a software that provides database services on a computer or a network.	**Buffer overflow (local_bof)**

3.1.1 Different Paths for the Above Example Attack

- **sshd_bof(0,1) → ftp_rhosts(1,2) → rsh(1,2) → local_bof(2)**

The first assumed attack path starts with *sshdbof(0,1)*. This indicates a buffer over exploit executed from Machine 0 (the workstation) against Machine 1 (the file server).*sshd_bof(0,1)*exploit is that the attacker can execute an arbitrary code on the file server. The *ftp_rhosts(1,2)* exploit is now possible, meaning that the attacker exploits a particular ftp vulnerability to anonymously upload a list of trusted hosts from *Machine 1* (the file server) to *Machine 2* (the database server). r*sh(1,2)*means **t**he attacker can leverage this new trust to remotely execute shell commands on the database server, without providing a password. A local buffer over exploit is then possible on the database server, which runs in the context of a privileged process. The result is that the attacker can execute a code on the database server with full privileges.

Other possible attack paths can be viewed as either of the following:

- **ftp_rhosts(0,1) → rsh(0,1) → ftp_rhosts(1,2) → rsh(1,2) → local_bof(2)**

- **ftp_rhosts(0,2) → rsh(0,2) → local_bof(2)**

3.1.2. Attack Graph from machine 0 to DB Server

In this section, we construct an attack graph of the example network so that each state transition corresponds to a single atomic attack by the intruder. A state in the model represents the state of the system between atomic attacks. The intruder launches his attack starting from a single computer, Machine 1. His eventual goal is to disrupt the functioning of the database. For which, the intruder needs

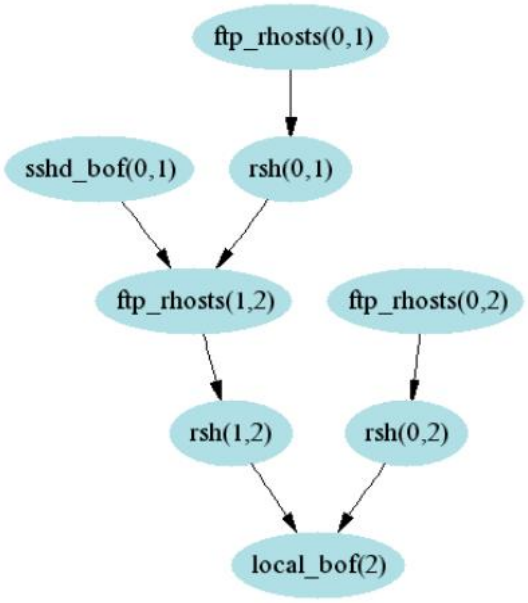

Fig. 2. Attack Graph of Above Example.

root access on the database Machine 2.

3.2 Types and Views

In this part, we describe various forms of attack graphs that are listed in some past papers.

3.2.1 Exploit-dependency Attack Graph

In the exploit dependency graph, each exploit or dependency appears only once, and no edges appear between independent exploits. For example, in the exploit dependency graph, each of the three exploits ftp_rhosts(0,1), sshd_bof(0,1), and ftp_rhosts(0,2) appears only once, and since these exploits are independent, there are no edges between them.

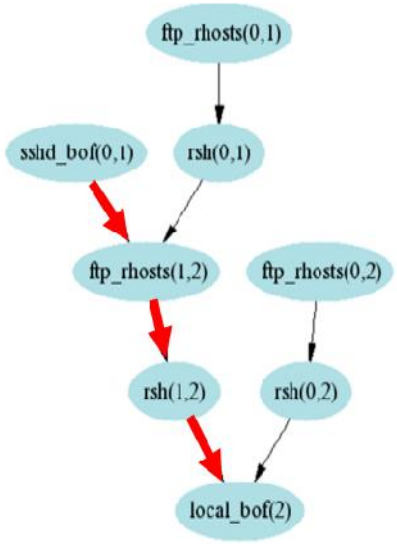

Fig. 3. Exploit-Dependency Attack Graph.

3.2.2 State-enumeration Attack Graph

Attack graphs represented transitions of a state machine, where states are network security attributes and state transitions are attacker exploits, resulting in graphs that enumerate transition paths through the state space. These state-based graphs have a property that one can simply follow a path through the graph to generate an attack path (sequence of exploits leading from the initial state to the goal one). But such graphs have serious scalability problems, as they can grow exponentially with the number of state variables.

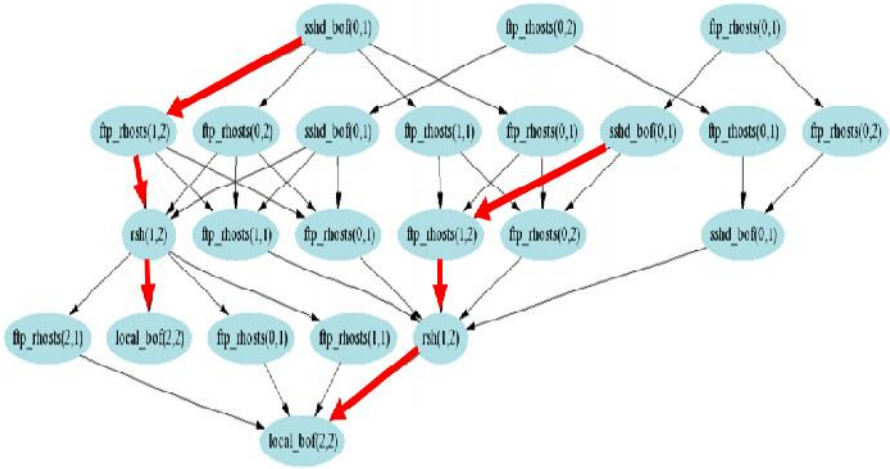

Fig. 4. State-enumeration Attack Graph.

3.2.3 Condition-oriented Attack Graph

In a condition-oriented attack graph, a *node* represents a *subset of the network state*, and an *edge* represents an *exploit* (or group of exploits) that moves the network from one state to another one.

Graph vertices represent conditions, which are connected by edges that represent exploits.

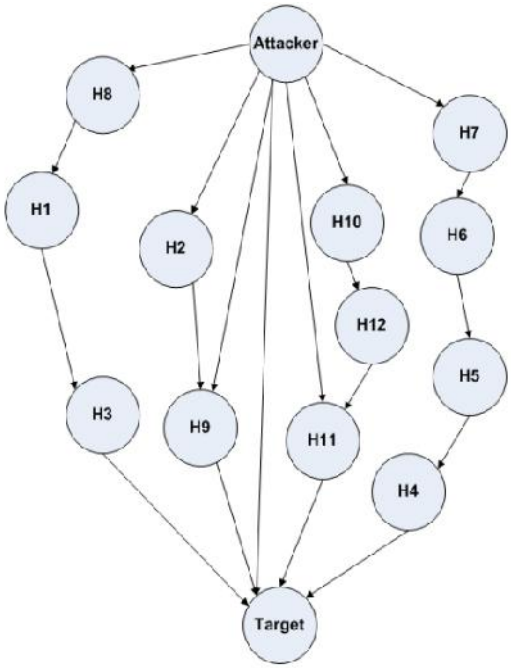

Fig. 5. A condition-oriented attack graph.

A state is a network attribute or a set of network attributes. Network attributes include hosts, host connectivity, and available software at hosts, access rights at hosts, and any other network characteristic deemed relevant to the modeler.

There are various types of “Condition-oriented Attack Graph” that have small differences between them: Finite State Machine (FSM) Attack Graph, Coordinated Attack Graph, Full Attack Graph, Host-compromised Attack Graph, Predictive Attack Graph, Node Predictive Attack Graph.

3.2.4 Exploit-oriented Attack Graph

An exploit-oriented attack graph is the reverse of a condition-oriented graph with respect to nodes and edges. State is represented in the edges of the graph and the exploits are represented in nodes of the graph. Exploit-oriented attack graphs may be referred to as exploit dependency graphs. A common representation of exploit-oriented attack graphs is to have unlabeled edges. The exploit-oriented attack graph's initial state(s) and the goal state(s) of the network are special nodes. Initial states are exploit nodes with null preconditions and true post-conditions Goal states are exploit nodes with true preconditions and null post-conditions

There are various types of “Exploit-oriented Attack Graph” that have small differences between them: Condition-exploit-oriented Attack Graph, Multiple Prerequisites Attack Graph, Logical Attack Graph, Hybrid-oriented Attack Graph.

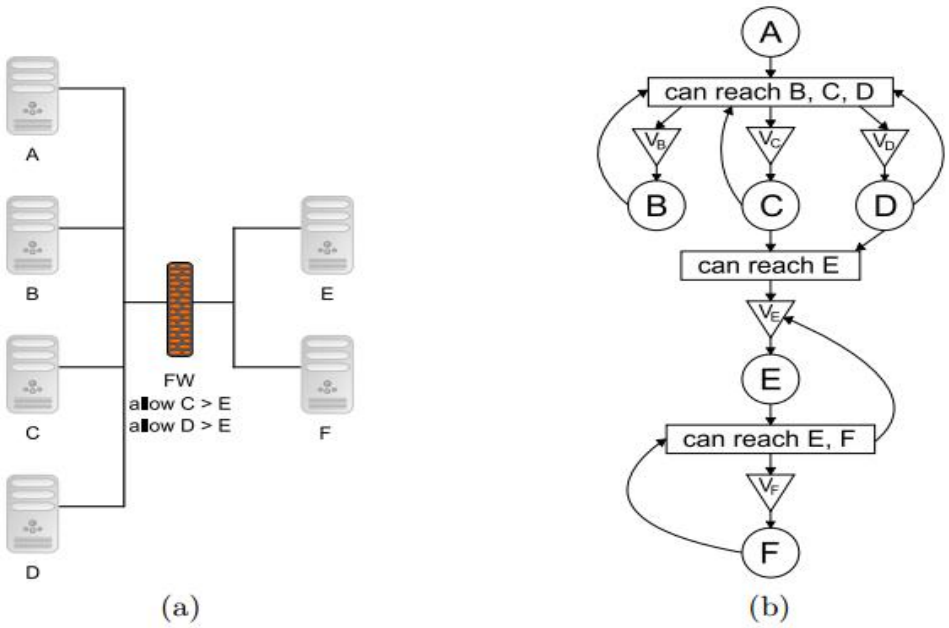

Fig. 6. A simple example network (a) , and its Multiple Prerequisites attack graph (b).

3.2.5 Attack Graph with Probabilities

Numbers are estimated probabilities of occurrence for individual exploits, based on their relative difficulty.

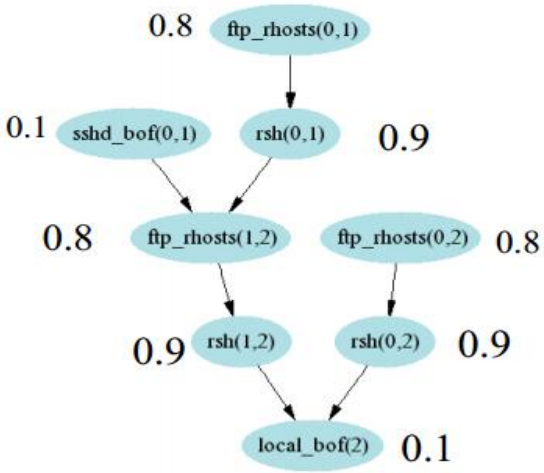

Fig. 7. Attack Graph with Probabilities.

Probabilities Propagated Through Attack Graph

When one exploit should follow another in a path, this means both are needed to eventually reach the goal, so their probabilities are multiplied: p(A and B) = p(A)p(B). When a choice of paths is possible, either is sufficient for reaching the goal: p(A or B) = p(A) + p(B) – p(A)p(B).

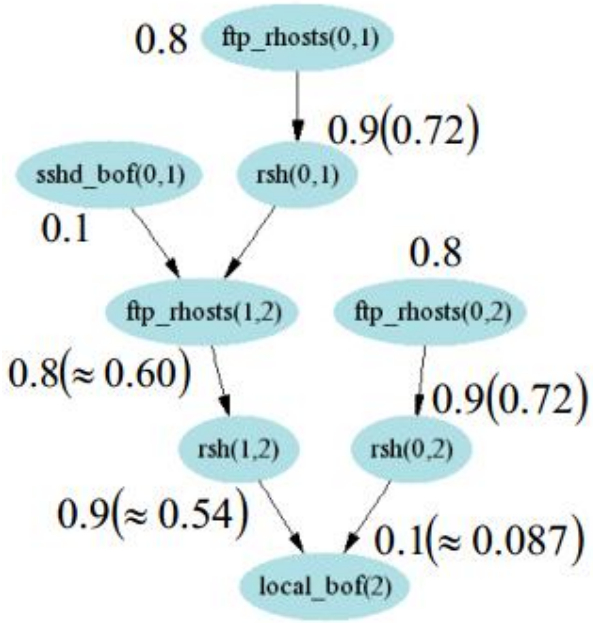

Fig. 8. Probabilities Propagated Through Attack Graph.

3.2.6 Attack Graph Aggregation

Machines and the exploits among them can be aggregated to a machine-exploit set if they form a connected sub graph, which allows machines to be aggregated across protection domains.

Complexity for "State-transition graph" is exponential and for "Exploit-dependency graph" is quadratic, but still too complex for easy understanding. 100 exploits could have up to 10000 edges.

Using hierarchical graph aggregation with abstraction is a solution for managing complexity.

4. Conclusions and Future Works

In this paper, we have proposed an overview and approach to definitions and survey of attack graphs. The main purpose of this approach is to emphasize the importance of attack graph as a high-performance network security solution.

Many related research should be done in the future: comparison of generating algorithms, review types of tool kits, and the methods to analyze attack graph will be further studied.

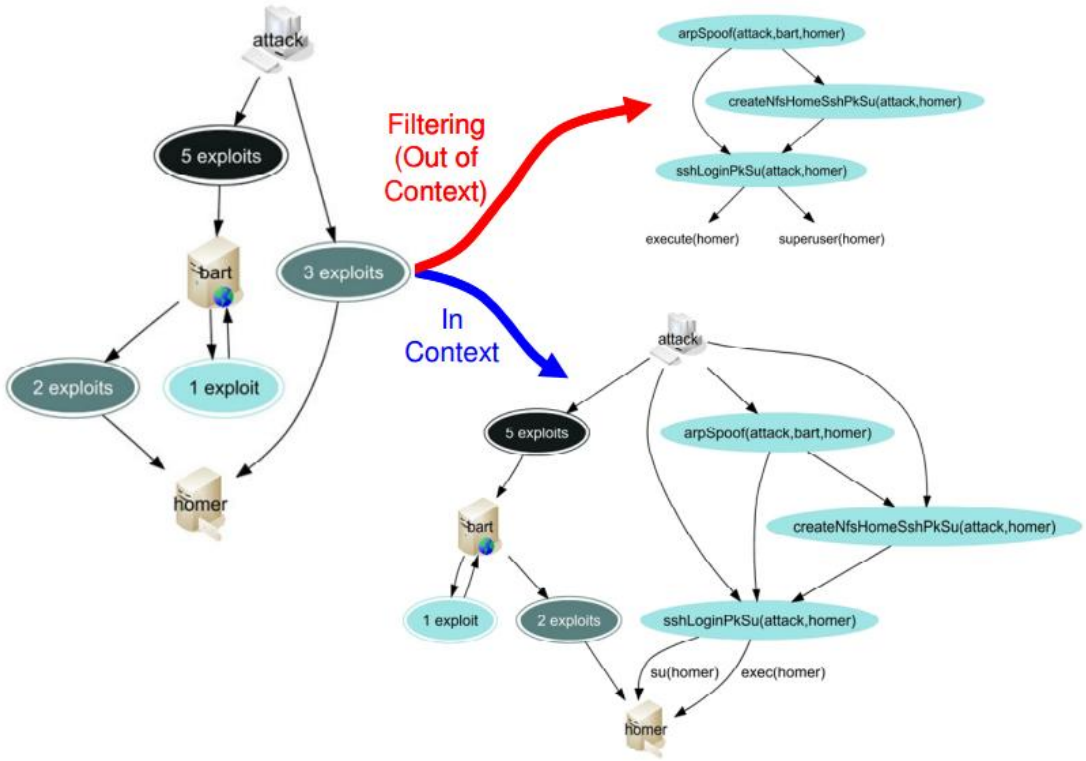

Fig. 9. Attack Graph Aggregation.

References

[1] K. Ingols, R. Lippmann and K. Piwowarski, *Practical Attack Graph Generation for Network Defense*, MIT Lincoln Laboratory, 2006.

[2] Common Vulnerabilities and Exposures (CVE®), The standard for Information security Vulnerability Names, [Online]. Available: http://cve.mitre.org

[3] Open Sourced Vulnerability Database, [Online]. Available: http://osvdb.org/

[4] Common Vulnerability Scoring System (CVSS-SIG), [Online]. Available: http://www.first.org/cvss

[5] National Vulnerability Database Version 2.2, NIST, USA, [Online]. Available: http://nvd.nist.gov/

[6] S. Jha, O. Sheyner and J.M. Wing, *Minimization and Reliability Analyses of Attack Graphs*. School of Computer Science Carnegie Mellon University, 2002.

[7] S. Noel, L. Wang, A. Singhal and S. Jajodia, "Measuring security risk of networks using attack graphs", *International Journal of Next-Generation Computing*, vol. 1, no. 1, pp. 135-147, July 2010.

[8] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation", *CCS Workshop on Visualization and Data Mining for Computer Security'04*, October 29, Fairfax, Virginia, USA, 10p., 2004.

[9] F. Chen, et al., "An atomic-domains-based approach for attack graph *generation", World Academy of Science, Engineering and Technology*, vol. 56, pp. 775-781, 2009.

[10] *M. Jamali and V. Ash*raf, "Attack graph analysis using parallel alg*orithm", 5th symposium on Advances in Science & Technology*, 7p., 2011.

[11] *N. C. Idika, Characterizing and Aggregating Attack Graph-based Security Metrics, Purdue University*, West Lafayette, Indiana, 2010.

Submitted 30.08.2013, accepted 11.10.2013.

Հարձակման գրաֆների համեմատական վերլուծություն

Լ. Ասլանյան, Դ. Ալիփոուռ և Մ. Հեյդարի

Ամփոփում

Հայտնի է, որ ժամանակակից համակարգիչներն ու ցանցերը, որոնք առանձնակի հզոր են իրենց հաշվողական և ծառայությունների մատուցման հնարավորություններով, ունեն նաև խնդիրներ՝ կապված դրանց խոցելիության և դրա հետ կապված տվյալների արտաքին բացահայտման հնարարավոր լինելու պարագայի հետ:

Ցանցային կառավարիչները և ցանցերի վերլուծման մասնագետները պետք է տեղյակ լինեն ընթացիկ համակարգերի խնդիրների հետ կապված համակարգերի և ցանցերի պաշտպանման մասին: “Հարձակման գրաֆները” ներկայացնում են ցանցերի հնարավոր հարձակումների ֆորմալացված պատկերը: Ներկա աշխատանքում դիտարկվում և վերլուծվում են անհրաժեշտ գաղափարները, որոնք առաջանում են հարձակումների գրաֆների ձևավորման և կիրառման ընթացքում:

Сравнительный анализ графов атак

Л. Асланян, Д. Алипоур и М. Гейдари

Аннотация

Современные компъютеры являются мощными вычислительными системами, а также системами предоставления вспомогательных информационных услуг. Вместе с тем, эти системы уязвимы с точки зрения внешних атак, что может являться причиной потери различных ценных информационных ресурсов. Системные администраторы обязаны знать всю информацию об уязвимости прикладных систем связанных с защитой сетей. Графы атак это удачный формализм, который предоставляет всю картину возможных атак для данной сети. Настоящая работа посвящена анализу и изучению средств проектирования и внедрения технологии графов атак.