Using Attack Graphs in Forensic Examinations

[*]Changwei Liu,[§]Anoop Singhal,[*]Duminda Wijesekera
cliu6@gmu.edu, anoop.singhal@nist.gov, dwijesek@gmu.edu
[*]Department of Computer Science, George Mason University, Fairfax VA 22030.
[§]National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg MD 20899.

Abstract-Attack graphs are used to compute potential attack paths from a system configuration and known vulnerabilities of a system. Attack graphs can be used to determine known vulnerability sequences that were exploited to launch the attacks and help forensic examiners in identifying many potential attack paths. After an attack happens, forensic analysis, including linking evidence with attacks, helps further understand and refine the attack scenario that was launched. Given that there are anti-forensic tools that can obfuscate, minimize or eliminate attack footprints, forensic analysis becomes harder. As a solution, we propose to apply attack graph to forensic analysis. We do so by including anti-forensic capabilities into attack graphs, so that the missing evidence can be explained by using longer attack paths that erase potential evidence. We show this capability in an explicit case study involving a Database attack.

Keywords-Attack Graph, Forensic analysis, Anti-forensics, Anti-forensics vulnerability database

I. INTRODUCTION

Digital forensics uses scientifically validated methods to collect, validate and preserve digital evidence derived from digital sources for the purpose of reconstruction of events found to be criminal or helping to anticipate unauthorized actions shown to be disruptive to planned operations [1]. Forensic examinations use the process of isolating attacked system and data, recovery, information collection, forensics analysis, and presenting evidence [3]. If an forensic examiner somehow knows the series of actions taken by an attacker, (s)he can look for digital evidence left behind by the attacker. Given a system configuration and known vulnerabilities existing in repositories such as NVD [17], attack graphs can provide all series of potential actions taken by any attacker (known as attack paths) to facilitate an investigation job.

Knowing that forensic examinations may be used to identify them, attackers take precautionary measures to minimize traceable information that may be used by forensics analysts. Such anti-forensics techniques and tools are now emerging in the arsenal of many attackers [14,15]. Their usages would leave gaps in evidence, which would make it difficult to link a series of exploits used by an attacker during a forensics examination.

Independently, attack graphs specify preconditions and post conditions of each act that can be used to create an attack. Combining them in a directed graph where the pre-conditions of a step are enabled by the post-conditions of prior executed steps, it would create an attack [4, 5, 6]. Therefore, given a set of vulnerabilities in a system, an attack graph analysis provides investigators with potential attack scenarios. Finding evidence that matches one or many such paths would then facilitate re-creating the attack. However, it would make this linkage problematic, given the fact that anti-forensics tools would erase some of these evidences. As a solution, we propose to enhance attack graphs with anti-forensic activity nodes that can be used to explain the missing evidence. To the best of our knowledge, combining anti-forensics techniques with attack graphs for forensic examinations is novel.

The rest of this paper is organized as follows. Section II describes related work, including a summary of attack graphs. Section III presents attack graph application for forensic analysis. Section IV describes proposed extensions to attack graphs, and our experiment demonstrating its utility in performing forensics examinations. Section V concludes the paper.

II. RELATED WORK

Attack graphs are directed graphs, where the nodes specify exploits with their pre-conditions and post-conditions. A directed edge from a source node to a destination node exists if the post condition of the source node satisfies a part of the pre-conditions of the destination node, so that the conjunction of all post conditions of source nodes imply the pre-condition of the destination node.

Attack graphs have many uses. They can be used to patch vulnerabilities existing in a system, or help find relevant attack information after an attack and etc.

Currently, many tools generate attack graphs and use them to secure systems and networks. For example, the TVA tool [7] exploits dependency graphs to represent the pre- and post-conditions [7]. The tool described by Ingols et al. [8] creates a network model using firewall rules and network vulnerability scans, and shows the effect of countermeasures on the system [8].

We use MulVAL toolkit, a rule-based toolkit that generates attack graphs [2, 5, 16] from network configurations, machine configurations and vulnerabilities from bug-reports.

Digital forensics analysis focuses on computer and network data. When performing forensics analysis on the date in a computer, typically, forensics investigators use imaging tools to extract a computer's physical memory or sectors of a disk to a file, and feed that file into data analysis tools [9], where either dead or live analysis will be performed. While live analysis risk getting changing data, a dead analysis is better but requires terminating all system processes [12].

To perform forensics analysis on data in a network, investigators extract valuable information from the network capture files that contain network's voice and data traffic since these files are from the users' environment with real time network activity or traffic. Some of these network forensic tools, such as SNORT [19], are well known in networking community and used as IDS as well.

Currently, most of the digital anti-forensic techniques fall in two categories of (1) attacking data, (2) attacking tools [15].

Techniques used to attack data include overwriting data or metadata in a file system, deleting files or media, hiding information using obfuscation, steganography and encryption, hiding data in unallocated spaces or slack space etc. [15]. Techniques used to attack forensics tools interfere with or mislead forensic analysis by crafting images or data that is not usable by tools [13].

An evidence graph takes a time-sequenced collection of intrusion evidence as nodes and correlation relationship between them as edges [10]. Evidence graphs facilitate presenting and reasoning about evidence collected after attacks, which is different from attack graphs as we discussed above. Work reported in [10] generates evidence graphs and attempts to automate the forensic analysis by using reasoning mechanism on the evidence using pristine and collected evidence. Liao et al. use expert systems with fuzzy rules to relax the assumptions on evidence quality to improve this problem [11]. However, both papers [10] and [11] focus on how to analyze evidence better in order to detect attacks. Contrastingly, our objective is to show investigators where evidence could be found or should be found but missing or hidden due to the usage of anti-forensic tools.

III. APPLYING ATTACK GRAPHS TO FORENSIC ANALYSIS

In this section we show how an attack graph can be used to search for forensic evidence after an attack. Our method expects to have two kinds of data, (1) known attack steps with their pre-conditions and post-conditions, (2) network and system configuration information, in addition to having an attack graph generation tool. After an attack occurs, our method consists of using attack graphs to generate potential attack paths, the attack steps with dependency relationships, which lead to the discovered attack and finding evidence to match the attack. We show how our method works by using the following example.

A. Experiment network and attack graph

The network we simulated is shown in Figure 1. In our experiment network, the external firewall controls network access from the Internet to the enterprise network, and the internal firewall controls the access to the database server that can be accessed by the webserver and workstations. The webserver hosts a webpage at port 8080 using the Tomcat 7 server that provides access to Internet users. The eventual attack we wish to execute is to gain access to database tables as an Internet user. We do so by launching a SQL injection attack that exploits the following java servlet code that does not sanitize input values:

(theResult = theStatement.executeQuery(

*"select * from profiles where name='Alice' AND password='"+passWord+"');).*

Suppose an internal user Alice uses a workstation that runs Windows XP SP3 operating system with IE6, which has vulnerability (CVE-2009-1918) that enables executing any code on this machine. As an external attacker we use social engineering to trick Alice to visit a malicious web page to control Alice's workstation.

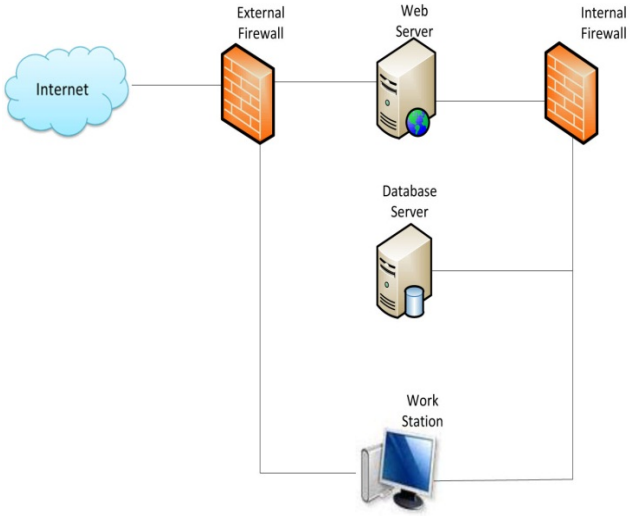


Figure 1: Experiment network

By using MulVAL with the above configuration and vulnerability information as input, we generated the attack graph shown in Figure 2, where the bold red line illustrates one of our attack paths. The square vertices in the graph represent system configurations including vulnerabilities, and diamonds represent privileges that an attacker gains by exploiting existing vulnerabilities. The ovals link preconditions to post-conditions of each attack step. The notation n:0 inside diamonds and ovals in Figure 2 are probabilities used for quantitative analysis that we do not use.

The 25 steps in Table 1 describe the 25 nodes in the attack graph of Figure 2, and is an explanation of the attack scenario. (Appendix 1 provides the full attack graph generated by MulVAL). In the attack graph, nodes 10 and 11 show that, initially, the attacker from the Internet can only access a webserver through port 8080. Node 20 and 21 show that the attacker can find out a workstation that connects to Internet, say Alice's machine. Exploring from the initial state with these pre-conditions, the attackers see two attack paths. Along the first path, the attacker can take advantage of the webpage input vulnerability in node 13 to remotely exploit the webserver by TCP protocol and port 8080 (node 6 and 7). Node 5 shows that the webserver connects to MySQL server by TCP protocol through port 3306. On the second path, the attacker can compromise a workstation by tricking the user to visit a malicious website (node 19, 18, 22, 23,

17). The workstation also provides access to the MySQL server, as shown in node 15 and 16. Either by using the webserver (node 4) or the compromised workstation (node 14), the attacker can launch the SQL injection attack on the MySQL server (node 3, 24, 25, 2 and 1).

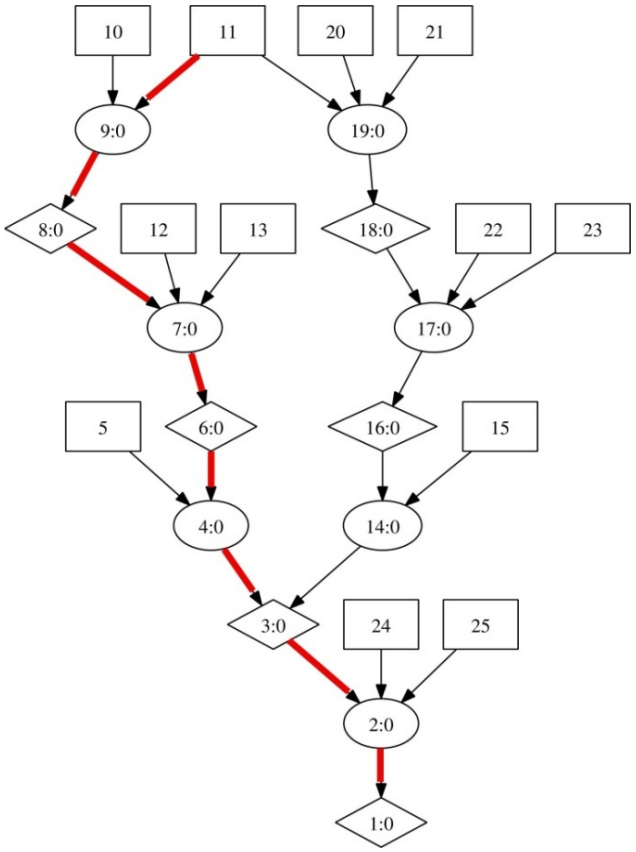


Figure 2: MulVAL Generated Attack Graph

B. Apply attack graph to forensics analysis

In order to show how attack graphs can be helpful for forensic analysis, we suppose the attack stated in Figure 2 was successful. Suppose the database general query logging was also working at that time. Then a forensic investigator could notice a log entry like "*120315 11:44:46 51 Query select * from profiles where name='Alice' and (password='alice' or '1' = '1')*", which is a typical SQL injection query without neutralizing the input data to remove "*or '1' ='1' "*. Our attack graph narrows the investigation to the webserver and Alice's workstation. Assume that there was no evidence found on the workstation. Instead, some IP addresses were found in the webserver log with a visit time close to the above malicious entry time in MySQL log file. For example, the following IP address, an IP address that is not from the enterprise network, with a timestamp *"15/Mar/2012:11:44:46"* was logged in our Apache Tomcat webserver.

"129.174.92.32 -- [15/Mar/2012:11:44:46-0400] POST /lab/Test HTTP/1.1" 200 670."

Considering that the webserver has only port 8080 open to the Internet, investigators could conclude that this attack took advantage of the web application string input vulnerability. In response, the network administrator can fix the vulnerabilities shown along the attack path of the graph.

TABLE1: ATTACK STEPS IN OUR ATTACK GRAPH

1	execCode(dbServer,user)
2	RULE 2 (remote exploit of a server program)
3	netAccess(dbServer,tcp,3306)
4	RULE 5 (multi-hop access)
5	hacl(webServer,dbServer,tcp,3306)
6	execCode(webServer,apache)
7	RULE 2 (remote exploit of a server program)
8	netAccess(webServer,tcp,8080)
9	RULE 6 (direct network access)
10	hacl(internet,webServer,tcp,8080)
11	attackerLocated(internet)
12	networkServiceInfo(webServer,httpd,tcp,8080, apache)
13	vulExists(webServer,'CWE89',httpd, remoteExploit,privEscalation)
14	RULE 5 (multi-hop access)
15	hacl(workStation,dbServer,tcp,3306)
16	execCode(workStation,user)
17	RULE 3 (remote exploit for a client program)
18	accessMaliciousInput(workStation,secretary,'IE')
19	RULE 22 (Browsing a malicious website)
20	hacl(workStation,internet,httpProtocol,httpPort)
21	inCompetent(secretary)
22	hasAccount(secretary,workStation,user)
23	vulExists(workStation,'CVE-2009-1918','IE',remoteClient,privEscalation)
24	networkServiceInfo(dbServer,mySQL,tcp,3306, user)
25	vulExists(dbServer,'SQLinjection',mySQL, remoteExploit,privEscalation)

Table 2 shows the evidence that might be left on log files, browsers' history and Temporary Internet files when an SQL Injection attack is launched. If investigators cannot find sufficient evidence, they would not be able to conclude the concurrence with the attack suggested by the attack graph. Moreover, attackers actively hide or overwrite the evidence. One explanation would be that the attacker has used anti-forensic techniques described in Section IV.

Conversely, forensics examiners can identify attacks not revealed in attack graphs. Attack graphs may have missed some vulnerabilities or configurations, and a careful forensic analysis could find unknown vulnerabilities, for example, by using Wang's [10]. Here, the idea is to use a reasoning framework to find the correlation between different evidence

TABLE2: EXAMPLE SQL INJECTION ATTACK VULNERABILITY FORENSIC ANALYSIS TABLE

ID	Attack	Software	Forensic Tool	Data	Key Data
SQL1	SQL Injection	MySQL 5.1 above		General Query log	'something' or '1'='1'
SQL2	SQL injection	Microsoft SQL server		Query Log	'something' or '1'='1'
SQL3	SQL injection	Microsoft SQL server	WFTSQL/Hypnosis	Cache Memory	'something' or '1'='1'
CVE-2009-1918-1	CVE-2009-1918	Windows/IE 6-8 CSS	Wireshark	pcap file	suspicious link based on port like 8080
CVE-2009-1918-2	CVE-2009-1918	Windows/IE 6-8 CSS		IE History	suspicious link based on port like 8080
CVE-2009-1918-3	CVE-2009-1918	Windows/IE 6-8 CSS		Local Setting Temp folder	VNC executable file
CWE-89-1	CWE-89	Software with MySQL		Log file on Software server	IP address combine SQL log

organized in a time sequence. In order to use Wang's hierarchical reasoning framework, evidence should be normalized and aggregated using a pre-processor. In real scenario, in some cases, it might be hard for investigators find valuable un-tainted evidence. Under this situation, hypotheses testing based on investigators' expert knowledge should be used to implement the "missing" evidence in order to build up the attack path to restore the attack scenario.

IV. ANTI-FORENSIC CAPABILITIES

In order to model anti-forensic activity in attack graphs, we propose to extend the attack graphs by adding a new type of nodes called anti-forensics activity and model the dependency between such a node and its ability to prevent a forensic tool from being used. This addition shows potential anti-forensics techniques that can be used to clean up the evidence left behind by an attack, which is used to identify an attack in Section III. Our anti-forensic activity nodes represent a new type of vulnerabilities that can be used against forensics analysis. When the evidence collected from the step stone computers or targeted computers cannot prove an unauthorized activity, our extensions could provide a potential two-level analysis. The first level explains how the attacker possibly launched the attack and the second level describes potential cover-ups. Below is the description of the details.

a) Adding Anti-forensics Activity Nodes

Based on the original attack graph, we add an extra node called the anti-forensic activity node along the attack path with corresponding privilege that might be used by the attacker to minimize the evidence generation or usage. Adding anti-forensics nodes to the attack graph has two advantages. First, our addition does not increase the complexity of the attack graph. This is relevant, because attack graph complexity in enterprise networks with lots of computers and complicated configurations has been a hindrance to using them commercially. Second, once attackers launch an attack, it is possible for them to escalate their privileges to perform anti-forensics activities. If the current attack path does not sufficiently escalate the privileges for the attacker to use anti-forensic tools, then (s)he may use other attack paths that can get to the same victim computer to gain them. For example, in Figure 2, an attacker can get to node "3" to execute a SQL query by the left path through un-sanitized string input vulnerability at the webpage. If privilege obtained along this path does not allow executing anti-forensic tools, then the attacker may use another path, say by hacking Alice's computer, to exploit the victim computer's IE vulnerability following a social-engineering attack.

One of the main advantages of our addition is that it does not destroy the causality relationship between attack states, and therefore one can use the original network configuration, vulnerability information, and the new escalated privilege to generate an attack graph with anti-forensic nodes. We propose that, along every exploit diamond node of the attack graph, an anti-forensics vulnerability node is added to model the anti-forensic capabilities.

b) Dependency between anti-forensics nodes and preventing forensic discovery

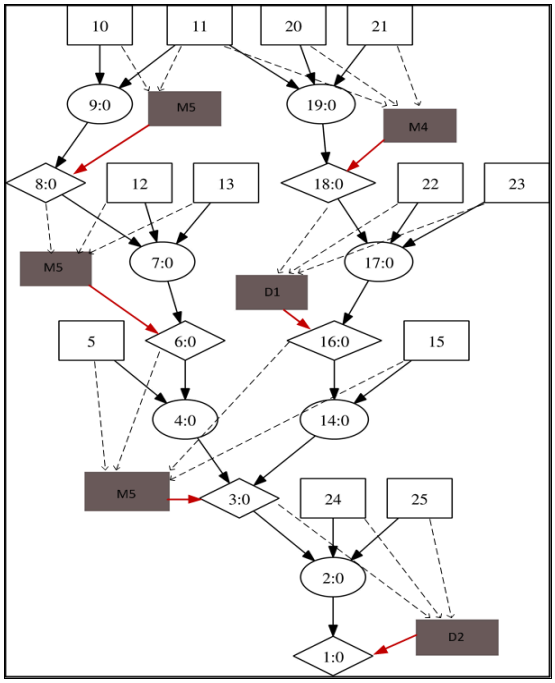
Dependency between anti-forensic nodes and their effect on forensic activity on a specific configuration can be specified as stated in the anti-forensic table given in Table 3. We can use this table and the missing evidence to reason which technique or tool has been used to remove forensic evidence.

A. Extending the Example Scenario

Figure 3 shows an attack graph enhanced with anti-forensics nodes shown as dark rectangles. We use the previous example and the attack graph shown in Figure 2. In this example, the specific anti-forensic activity explanation for each node can be found in Table 3. Take the right path (11->18->16->3->1) with its anti-forensics nodes as example. It shows that an attacker uses the CVE-2009-1918 vulnerability on the windows workstation to fully control Alice's machine, in which the attacker is able to use the escalated root privilege to remove the malicious link sent to the workstation either by the shell or VNC (we use TIGHTVNC in our experiment) [18] if no one sits in front of the computer. VNC is remote control software that can used to see and interact with desktop application across any network. Shell commands are stealthier because they have no

TABLE3: THE ANTI-FORENSIC TECHNIQUE/TOOL VULNERABILITIES DATABASE

ID	Category	Tool	Technique	Windows	Linux	Privilege	Access	Vulnerability	Effect
A1	Attack Tool		Obfuscate signature	All	All	User	Internet	SNORT Rule	Bypass being detected by rules
D1	Destroy Data	BCWipe	Delete file content	98 Above	All	User	Computer		Delete data permanently
D2	Destroy Data		Remove log file	All	All	User	Internet	MySQL5.1 above set log off command	Set general log off
H1	Hide Data	Steghide	Steganography	All	All	User	Computer		Hide data to image or audio file
H2	Hide Data	Slacker.exe		2000 Above	No	User	Computer		Hide data in slack space
H3	Hide Data	TrueCrypt	Encryption	XP Above	All	User	Computer		Encrypt disk or partition and hide
H4	Hide Data	Rootkit	Hide data in memory	All		User	Internet		Bypass live incident response
H5	Hide Data	Evil Maid	Encryption passphrase	All	All	User	Physical Computer	TrueCrypt bootloader	Capture the key
M1	Minimize footprint	ShellCode	Memory injection	All	All	User	Internet	Buffer Overflow	No code in hard disk
M2	Minimize footprint	Reverse Shellcode	Syscall Proxying	All	All	User	Internet	Buffer Overflow	Attack without code
M3	Minimize footprint	Metasploit meterpreter	Memory injection	All	All	User	Internet	Metasploit exploit	Hack the victim machine
M4	Minimize footprint	random js toolkit	Change URL	All	All	User	Internet	Browser	Infect webpages
M5	Minimize footprint	A4Proxy	Hide IP	All	All	User	Computer		Hide ID when surfing on internet
T1	Trail Obfuscation		Obfuscate payload	All	All	User	Internet	SNORT Rule	Obfuscate payload to bypass SNORT



M4: Change URL
M5: Hide IP
D1: Delete file content
D2: Remove log file
Note: The specific information is in table 3.

Figure 3: Anti-forensics Nodes

any visual effect of the desktop. For the next step of attacking the MySQL Server, the attacker can either use command at runtime to turn off MySQL server logging (for our experiment version MySQL 5.5, "*SET GLOBAL general_log = 'OFF';*" can be sent from the webpage in Appendix 2 to turn off the query log file) or attack the database server by exploring more vulnerabilities (such as CVE-2009-2446 for a MySQL version from 4.0.0 to 5.0.83) to physically delete the log file.

In an actual scenario, an attacker may not perform anti-forensic attacks on every attack step. Accordingly, the investigators do not need to look up the Anti-forensics database at every step as long as they can find sufficient evidence left behind by the attacker.

B. The Anti-Forensic Capabilities Database

We now describe the details contained in our anti-forensics database. As shown, Table 3 has attributes "ID", "Category", "Tool", "Technique", "Windows", "Linux", "Privilege", "Access", "Vulnerability" and "Effect". "ID" is used as a primary key for each record. We use attribute "category" to categorize the anti-forensics technique and tools. For example, "destroy data" means that the evidence data will be destroyed, but the evidence data would probably be hidden in either memory or hard disk by using different tools or techniques. The category "Hide data" shows this latter capability with a list of tools. The attributes "tool" and "technique" list what tools and technique an attacker might use for anti-forensics. "Windows" and "Linux" are OS

platforms where the tool and technique would be used. The attribute "privilege" states escalated privilege an attacker may need in order to perform anti-forensics on the specific platform using a specific tool or attack. There are three kinds of access in attribute "Access", from which the attacker would launch the specific anti-forensics. Notice that "physical computer" is different from "computer" here. While we define "physical computer" as physical access to a computer, "computer" means that the anti-forensics can also be done by remote access to the computer through Internet by using shell or a remote desktop control tool such as TightVNC [18] to install anti-forensic tool on the victim computer. Lastly, the attribute "vulnerability" is used to show the vulnerability exploited by the attacker, and the attribute "effect" specifies effect of the exploit on the data record.

By querying different attributes, when we trace the attack on a machine with a specific configuration and privileges, we can find anti-forensic capabilities available to a potential attacker at each step in minimizing the evidence potentially left behind the attack.

Combining the machine configuration in the attack graph and the collected evidence for a specific attack investigation, the forensic investigator can find out what evidence has been removed by comparing the collected evidence to forensic analysis database shown in Table 2 that illustrates what evidence should be left over. Once the investigator has determined that anti-forensic has been used, with the collected existing evidence and information obtained from reading the attack graph, s(he) can query the anti-forensics table in order to find out what techniques or tools might have been used by the attacker to clean up potential evidence. Therefore the investigator can reconstruct potential attack scenarios even with partial evidence.

Constructing the anti-forensics database requires that repositories like National Vulnerability Database (NVD) [17] be enhanced to collect the capabilities of anti-forensic technique or tools and provide them as a part of the NVD repository.

V. CONCLUSIONS

We showed how attack graphs could be used to help forensics investigators narrow down potential attack scenarios, along with evidence left by attackers. By using anti-forensic tools and techniques, attackers can prevent such activities by ensuring that evidence left behind is minimized, obfuscated or removed completely. In order to recreate attack scenario in the presence of anti-forensics tools and techniques, we propose enhancing attack graphs with anti-forensics nodes and an anti-forensics database. We showed how these two additions could be used to recreate attack scenarios from partial evidence. Given that NVD helps in preventing attacks and detecting attack paths when we find ourselves attacked, we advocate that NVD be enhanced with anti-forensics databases. We would also like to enhance NVD with the capabilities for anti-forensic techniques or tools.

ACKKNOWLEDGEMENT

We would like to thank the anonymous reviewers for their valuable feedback and suggestions that helped improve the last version of the paper.

REFERENCES

[1] A. Jaquith, "Security Metrics: Replacing Fear, Uncertainty, and Doubt", Addison Wesley, Mar 26, 2007.

[2] A. Singhal, X. Ou, "Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs", NIST InterAgency Report, September 2011.

[3] Adrian T.N. Palmer, "Computer Forensics: The Six Steps", US-CERT, 2008.

[4] P. Ammann, D. Wijesekera, S. Kaushik. "Scalable, graph-based network vulnerability analysis", In Proceedings of 9th ACM Conference on Computer and Communications Security, Washington, DC, November 2002

[5] X Ou, W. F. Boyer, M. A. McQueen, "A scalable approach to attack graph generation," In: 13th ACM Conference on Computer and Communications Security (CCS), pp. 336–345 (2006).

[6] R. Lippmann, K. Ingols. "An annotated review of past papers on attack graphs", Technical report, MIT Lincoln Laboratory, March 2005.

[7] S. Jajodia, S. Noel, B.O.'Berry. "Topological Analysis of Network Attack Vulnerability", In Managing Cyber Threats: Issues, Approaches and Challenges, V. Kumar, J. Srivastava, A. Lazarevic (eds.), Springer, 2005.

[8] K. Ingols, M. Chu, R. Lippmann, S. Webster and S. Boyer. "Modeling Modern Network Attacks and Countermeasures Using Attack Graphs", Proceedings of ACSAC Conference 2009.

[9] SANS Institute InfoSec Reading Room, "an Overview of Disk Imaging Tool in Computer Forensics", 2001.

[10] W. Wang, E.D. Thomas, "A graph based approach toward network forensics analysis", ACM Transactions on Information and Systems Security 12 (1) 2008.

[11] N. Liao, S. Tian, T. Wang, "Network forensics based on fuzzy logic and expert system", Computer Communication, vol. 32, no. 17, pp. 1881–1892, 2009.

[12] B. Carrier, "File System Forensic Analysis", Addison-Wesley Professional, March 2005.

[13] B. Sullivan, "application-Level Denial of Service Attacks and Defenses", Presented in conjunction with the BlackHat DC 2011 talk, January 2011.

[14] S. Garfinkel. "Anti-Forensics: Techniques, Detection, and Countermeasures", In Proceedings of the 2nd International Con- ference on i-Warfare and Security (ICIW), Naval Postgraduate School, Monterey, CA, March 2007.

[15] M. Whitteker, "Anti-forensics: Breaking the forensic process", Information Systems Security Association Journal, pp. 10-16, November 2008.

[16] MulVAL V1.1, Jan 30, 2012, http://people.cis.ksu.edu/~xou/mulval/.

[17] National Vulnerability Database,

http://nvd.nist.gov/.

[18] TightVNC Software,

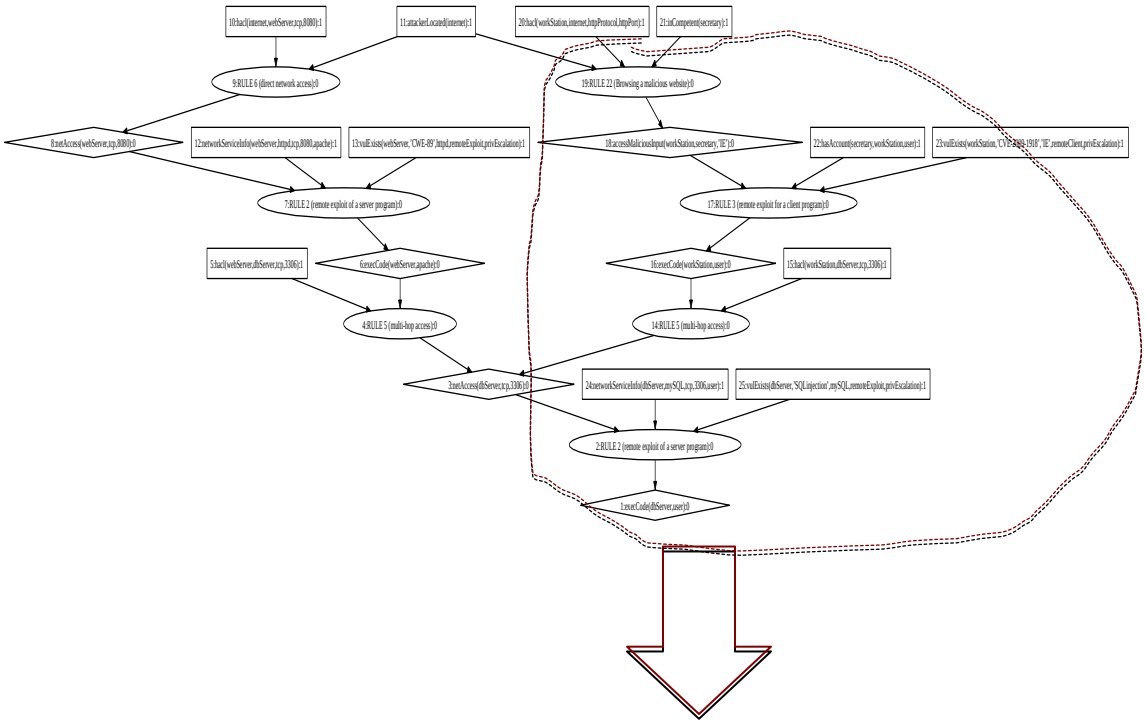
http://www.tightvnc.com/.

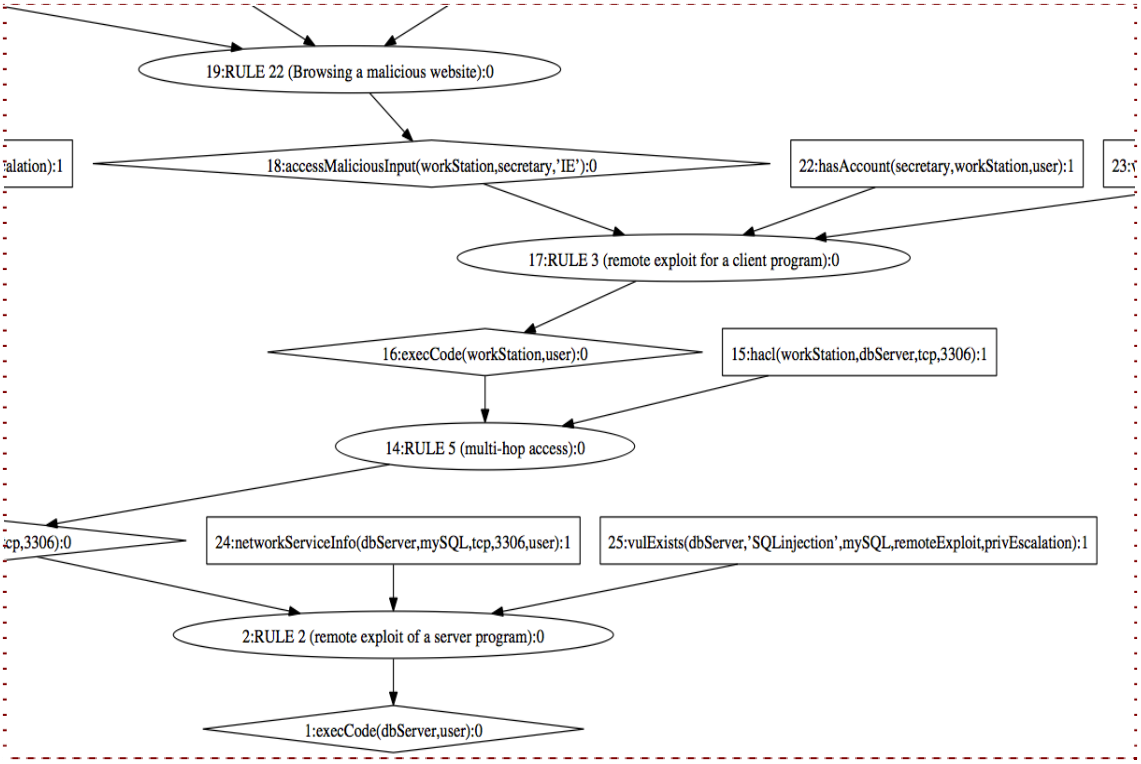
[19] SNORT,

http://www.snort.org/

APPENDIX

1. The attack graph for our experimental network *(we amplify a part of the graph for a clearer view)*





2. Experiment webpage that has authentication vulnerability