THE JOURNAL OF
DIGITAL FORENSICS,
SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 9 | Number 2 Article 15

2014

Relating Admissibility Standards for Digital Evidence to Attack Scenario Reconstruction

Changwei Liu
George Mason University

Anoop Singhal
National Institute of Standards and Technology

Duminda Wijesekera
George Mason University

Recommended Citation

Liu, Changwei; Singhal, Anoop; and Wijesekera, Duminda (2014) "Relating Admissibility Standards for Digital Evidence to Attack Scenario Reconstruction," *Journal of Digital Forensics, Security and Law*: Vol. 9 : No. 2 , Article 15.
DOI: https://doi.org/10.15394/jdfsl.2014.1180
Available at: https://commons.erau.edu/jdfsl/vol9/iss2/15

RELATING ADMISSIBILITY STANDARDS FOR DIGITAL EVIDENCE TO ATTACK SCENARIO RECONSTRUCTION

Changwei Liu
Department of Computer Science, George Mason University
Fairfax VA 22030 USA
cliu6@gmu.edu

Anoop Singhal
National Institute of Standards and Technology
Gaithersburg MD 20899 USA
anoop.singhal@nist.gov

Duminda Wijesekera
Department of Computer Science, George Mason University
Fairfax VA 22030 USA
&
National Institute of Standards and Technology
Gaithersburg MD 20899 USA
dwijesek@gmu.edu

ABSTRACT

Attackers tend to use complex techniques such as combining multi-step, multi-stage attack with anti-forensic tools to make it difficult to find incriminating evidence and reconstruct attack scenarios that can stand up to the expected level of evidence admissibility in a court of law. As a solution, we propose to integrate the legal aspects of evidence correlation into a Prolog based reasoner to address the admissibility requirements by creating most probable attack scenarios that satisfy admissibility standards for substantiating evidence. Using a prototype implementation, we show how evidence extracted by using forensic tools can be integrated with legal reasoning to reconstruct network attack scenarios. Our experiment shows this implemented reasoner can provide pre-estimate of admissibility on a digital crime towards an attacked network.

Keywords: forensics, electronic crime, digital evidence, admissibility, network attack scenario, evidence graph

1. INTRODUCTION

During trials, judges are often asked to rule on the admissibility of electronic evidence. In preparation for arguments relating to admissibility, digital crime investigators look for evidence and arguments about their attributes (described shortly) in order to convince a judge or a jury at the trial stage that the potential attack scenario they constructed from the evidence is more convincing than the one presented by the defense. According to federal rules on digital forensics, five issues must be considered when assessing whether given digital evidence will be admitted. They are (1) relevance, (2) authenticity, (3) not hearsay or admissible hearsay, (4) best evidence, and (5) not unduly prejudicial. Although some of these issues may not directly apply to all instances, a prosecution considers all of them and their applicability. Otherwise, he/she runs the risk of evidence being ruled as insufficient in the court, and consequently, the charge would be dropped or the accused would be ruled not guilty as charged.

Independent of these legal standards, to avoid being traced or leaving incriminating evidence, attackers tend to use complex techniques such as multi-host, multi-step attacks covered up by using anti-forensic techniques or tools, which makes finding incriminatory evidence and reconstructing an attack problematic. In attacks committed against digital assets, attack scenarios do constitute a substantial part of a prosecution's narrative that needs to be accepted by a jury or a judge to get a conviction.

Although using IDS alerts as forensic evidence has been contested, they provide the first level of information to forensics analysts on creating potential attack scenarios (Sommer, 2003). In order to use IDS alerts as evidence, many have proposed aggregating redundant alerts and correlating them to determine multi-step, multi-stage attacks (Dain and Cunningham, 2001; Debar and Wespi, 2001; Wang and Thomas, 2008). While such aggregation can help in reconstructing a multi-stage, multi-step attack scenario, to the best of our knowledge, none of this work has been integrated with legal acceptability standards of evidence. In some cases, forensic investigators can quantify assertions by quantifying the relevance of evidence and their credibility with uncertainty (Eoghan, 2002). However, in some cases, the evidence could be missing or intentionally destroyed, which prevents quantifying their relevance and credibility. In order to address the problem of missing or destroyed evidence and assign weights to an attack scenario constructed from some evidence over the alternatives in supporting or refuting such an construction (that is what constitutes a prosecution's story vs. a defense's alternative explanation that exonerates the accused), we use an anti-forensics database and corresponding hypothesis to help with expanding investigations (Liu, Singhal, and Wijesekera, 2012 and 2014). Also, substantial amount of researchers have proposed using Bayesian networks to quantify the relevance and credibility of criminal or digital evidence for quantifying admissibility judgments. In addition, Dempster-Shaffer theory has also been used in calibrating digital evidence. We found little if any publications that formalized legal acceptability and the associated quantification method to directly applicable software in order to construct multi-step, multi-stage attack scenarios for the prosecutorial use in a court of law. As a preliminary step towards achieving this goal, we proposed to use federal rules as guidelines to pre-check the admissibility of evidence (Liu, Singhal, and Wijesekera, 2014). However, in that work, we did not describe a specific method to integrate federal rules to a Prolog reasoning based system, nor did we discuss how to assign quantitative values to evidence credibility with aspects of legal standards and integrate them to an attack scenario. This paper provides that missing link by formalizing legal requirements of evidence admissibility to our attack scenario reconstruction framework, and shows how to quantify the forensic investigators' assertion on the attack represented by given evidence.

The rest of the paper is written as follows. Section 2 describes background of the addressed problem and related work. Section 3 describes an experimental network and the constructed attack scenario by using Prolog reasoning rules. In Section 4, we discuss how to formalize the specific predicates and rules in a Prolog based system to reflect the admissibility of federal rules. In Section 5, we describe how to integrate the quantified evidence to our constructed evidence graph for admissibility evaluations. Section 6 shows the applicability of our method by using an experimental network. Lastly, Section 7 has our conclusions.

2. BACKGROUND AND RELATED WORK

Our framework codifies federal rules on digital evidence in a Prolog based tool, which constructs an evidence graph visualizing the corresponding attack scenarios that happened in an attacked network. In order to explain our method, this section describes the admissibility requirements of digital evidence and the definition of an (probabilistic) evidence graph.

2.1 Admissibility/Credibility of Digital Evidence

According to federal rules on digital forensics, whenever the admissibility of digital evidence is called into question, following rules are applied. They are (1) Authenticity (Rules 901 and 902), (2) Hearsay or not (Rule 801-807), (3) Relevance (Rule 401), (4) Prejudice (Rule 403), (5) Original writing (Rule 1001-1008). Among them, the most important rule is relevance criteria. Federal rules of evidence do not include a formal mathematics or statistics standard that evaluates levels of certainty associated with digital evidence, because there isn't a universally accepted way to assess the credibility or accuracy of digital evidence due to many reasons. First, the complexity and multiplicity of computer systems and attack techniques vary. Second, the level of attack certainty (the credibility/weight) that digital investigators assign to their findings is influenced by their expertise and experience (Casey, E.). However, federal rule 104(e) is related to credibility or weight, which does not limit the right of a party to introduce evidence to a jury prior to determining its relevance and credibility. Here, the so-called weight is described as the measure of credible proof on one side of a dispute as compared with the credible proof on the other (Balls, Amcoff, Bremer, Casati, Coecke, and Clothier, 2005).

In order to facilitate the expression of opinions regarding the certainty of an assertion on the evidence credibility and weight, Bayesian network modeling is used to calculate the analysts' judgment on the certainty of evidence (Liu, Singhal, and Wijesekera, 2012; Keppens, Shen, and Schafer, 2005; Kwan, Chow, Law, and Lai, 2008). Besides, Weiss 2003 suggests using a 10-point scale of certainty based on legally defined standards of proof and compares "legal", "scientific" and "Bayesian" scales, which provides a more precise way to characterize a forensics analyst's judgment. The hierarchy of legal standards of proof and corresponding Bayesian probabilities consists of (from highest to lowest) (1) beyond any doubt (100%), (2) beyond a reasonable doubt (>99%), (3) clear and convincing evidence (90-99%), (4) clear showing (80-90%), (5) substantial and credible evidence (67-80%), (6) preponderance of the evidence (50-67%), (7) clear indication (33-50%), (8) probable cause: reasonable grounds for belief (10-33%), (9) reasonable, articulable grounds for suspicious (1-10%), (10) no reasonable grounds for suspicion (<1%), and totally impossible (0%). Civil and criminal litigation uses different levels of acceptability of evidence.

2.2 (Probabilistic) Evidence Graph

An evidence graph is a graphical model, which presents evidence of intrusions and their dependencies that can be used to reconstruct multi-stage and multi-step attacks that may have been launched against an enterprise network. The formalized definition can be found in (Wang, and Thomas, 2008; Liu, Singhal, and Wijesekera, 2013). The following is the one defined in Liu, Singhal and Wijesekera.

Definition 1 (Evidence Graph): An evidence graph is a sextuple G=(N_h, N_e, E, L, N_h-Attr, N_e-Attr), where N_h and N_e are two sets of disjoint nodes representing a host computer involved in an attack and its related evidence; E $\subseteq(N_h \times N_e) \cup (N_e \times N_h)$; L is mapping from a node to its label; N_h-Attr and N_e-Attr are attributes of a host node and an evidence node respectively.

- Attributes of a host node

a. Host ID: Identity of a host node.
b. States: Host node category consisting of one or many of the "source", "target", "stepping-stone" and "affiliated". Affiliated hosts have suspicious interactions with an attacker, one of victims or stepping-stone computers.
c. Timestamps: Time stamps that record the attack states of a machine. System time should be synchronized in a distributed system.

- Attributes of an evidence node

a. General attributes: Includes event initiator, event target, event description and event time stamp(s).
b. Relevancy(r): Measurement of evidence impact on attack success, which includes the irrelevant true positive = 0, unable to verify = 0.5 and relevant true positive =1.
c. Weight (w): A value from [0, 1] that is used to represent the impact of evidence on the corresponding attack. For example, port-scan evidence gets less weight than buffer overflow evidence.
d. Host importance (h): A value from [0,1] indicating the bigger importance of two hosts connected by an evidence edge.

Liu, Singhal, and Wijesekera (2013) defined a probabilistic evidence graph to quantify the overall weight of intrusion evidence that represents the attack certainty in an attack scenario. Similar work that uses Bayesian network modeling can be found in Kwan, Chow, Law, and Lai (2008), but the definition in Liu, Singhal, and Wijesekera (2013) uses the evidence attributes from Definition 1, and it calculates p(e) and p(h) as follows.

Definition 2(Probabilistic Evidence Graph): is a graph G=(N_h, N_e, E, L, N_h-Attr, N_e-Attr, p) where the probability assignment functions p[0,1] for an evidence edge "e" and a victim host "h" are defined as follows.

a. p(e)=c(e) × w(e) × r(e) × h(e), where "w", "r" and "h" are weight, relevancy and the importance of an evidence edge "e" (Wang, W., and Thomas, E.D., 2008). "c" is the category of evidence, including primary evidence, secondary evidence and hypothesis testing based on expert knowledge. While primary evidence is explicit and direct, second evidence is implicit and circumstantial. Hypothesis is the expert opinion when a particular attack is not substantiated by physical evidence. 1, 0.8 and 0.5 are suggested to assign to the three different evidence categories respectively. However, the assignment could be different by different expert knowledge or the value can be changed if the evidence category has been changed. For example, the hypothesis value 0.5 will be replaced by 1 if solid primary evidence has been found.

b. $p(h)= p[(\cup e_{out})\cup(\cup e_{in})]$, where e_{out} are all edges whose source computer is host "h" with a given attack-related state, and e_{in} are all edges whose target computer is "h" with the same state.

2.3 Related Work

In the area of non-digital forensics, Keppens and Zeleznikow (2003) describes how to use inductive and abductive reasoning to model potential crime scenarios and correlate evidence. Keppens, Shen, and Schafer (2005) uses Bayesian inference to evaluate how well the given criminal evidence is a better fit for substantiating one scenario over possible alternatives. Both papers are based on traditional forensics, and do not apply to digital evidence per-se.

In the area of digital forensics, Wang and Thomas describe one schema that use reasoning to correlate attack steps substantiated by digital evidence, which uses aggregated security event alerts by checking if they have the same source-destination pair, belong to the same attack class and timestamps falling within a self-extending time window. A self-extending time window is a time window that is updated by elongating the window to include the one with a new alert but the time difference is within a predefined limit T.

Definition 1 also decorates an item of evidence between two hosts with relevance, weight and host importance to assess the overall weight of the aggregated evidence. Based on this extra information, we discussed how to compute the attack credibility of the attacked host computer substantiated by evidence, but did not relate it to the quantitative aspects of legal standards. Also, in order to quantify the strengths of hypotheses that are based on the legal admissibility judgment of evidence, Kwan at el. proposed using a Bayesian belief network to analyze the evidence with respect to hypothetical scenarios, and thus, enhancing the credibility and traceability of the results produced by digital forensic investigations (Kwan, Chow, Law and Lai, 2008). The Bayes' theorem used for evidence analysis in Kwan is P(E|H)=P(E)P(H|E)/P(H), where P(E|H) is the conditional probability of evidence E based on hypothesis H; P(E) is the prior probability of evidence E; P(H|E) is posterior probability representing the probability that hypothesis H has actually occurred when E is detected; and P(H) is the prior probability of hypothesis H judged by investigators' background knowledge. In their paper, Hwan at el. also demonstrated how to use this Bayesian probability assignment model by applying the model to a digital forensics legal case from Hong Kong Police Department, in which the defendant used his computer to distribute a pirated movie on the Internet by using BitTorrent (Magistrates' Court at Tuen Mun).

Wang and Thomas (2008) use fuzzy-logic based correlation to find the causality relationship between different evidence, which constructs the attack scenario in a graph form. Fuzzy logic does not learn membership rules during or after problem solving. Also, it uses imprecise values, which makes it difficult to use any mathematical model to compute precise numbers. In order to solve these limitations and integrate an anti-forensics database to hypothesize about missing or destroyed evidence, we proposed to modify a Prolog based attack graph generation tool, MulVAL, to reconstruct attack scenarios (Liu, Singhal, and Wijesekera, 2014). MulVAL adopts Datalog as the modeling language to analyze the software vulnerability, configuration description, rules and other security related facts for constructing and reasoning about attack paths. This methodology makes it easy to leverage existing vulnerability database and scanning tools by expressing their output in Datalog and feeding it to its reasoning engine. In our previous paper, we proposed to integrate federal rules of evidence into MulVAL, but did not provide a way to formalize the admissibility standards or compute the credibility of forensics investigators' assertions substantiated by evidence.

3. AN EXAMPLE NETWORK AND RECONSTRUCTED ATTACK SCENARIO

3.1 Our Experimental Network

Figure 1 shows the topology of our experimental network that is similar to Liu, Singhal, and Wijesekera (2014). Table 1 shows the machine IP address and vulnerability information. By exploiting vulnerabilities listed in Table 1, we were able to successfully launch two kinds of attacks on the database server. One was launched by using a workstation to get access to the database server (CVE-2009-1918), and the other was by using the webserver to attack the database server (SWE89). Our installed intrusion detection system, configured webserver and database server were able to detect the attacks and log malicious accesses. The corresponding aggregated alerts and log information used as evidence are shown in

Table 2, which were used in the extended MulVAL for attack scenario reconstruction. We chose the last alert or log item's time stamp in Table 2 as the time stamp of the aggregated alert or log from the hyper alerts or logs that have the same source-destination pair, belong to the same attack class with time stamp falling in our pre-defined time window.

3.2 Reconstructing Attack Scenarios

By using the modified MulVAL reasoning model

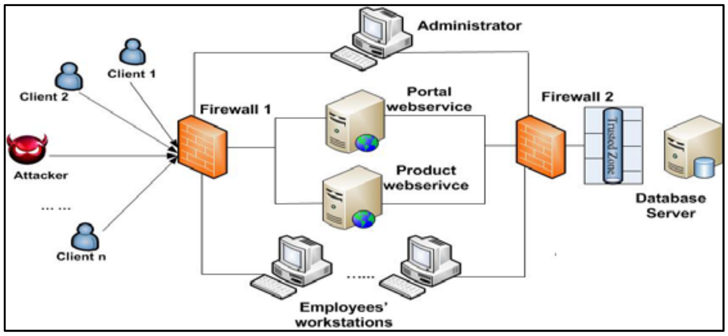

Figure 1 an Experimental Attacked Network

Table 1 Machine IP Address and Vulnerability

Machine	IP Address/Port	Vulnerability
Attacker	129.174.124.122	
Workstations	129.174.124.184/185/186	HTML Objects Memory Corruption Vulnerability (CVE-2009-1918)
Webserver1--Product Web Service	129.174.124.53:8080	SQL Injection (CWE89)
Webserver2--Portal Web Service	129.174.124.53:80	SQL Injection (CWE89)
Administrator	129.174.124.137	Cross Site Scripting Flaw (XSS)
Database server	129.174.124.35	

Table 2 Formalized Evidence of the Alerts and Log from the Attacked Experimental Network

Timestamp	Source IP	Destination IP	Content	Vulnerability
08/13-12:26:10	129.174.124.122	129.174.124.184	SHELLCODE x86 inc ebx NOOP	CVE-2009-1918
08/13-12:27:37	129.174.124.122	129.174.124.185	SHELLCODE x86 inc ebx NOOP	CVE-2009-1918
08/13-14:37:27	129.174.124.122	129.174.124.53	SQL Injection Attempt	CWE89
08/13-16:19:56	129.174.124.122	129.174.124.137	Cross Site Scripting	XSS
08/13-14:37:29	129.174.124.53	129.174.124.35	name='Alice' AND password='alice' or '1'='1'	CWE89
... ...				

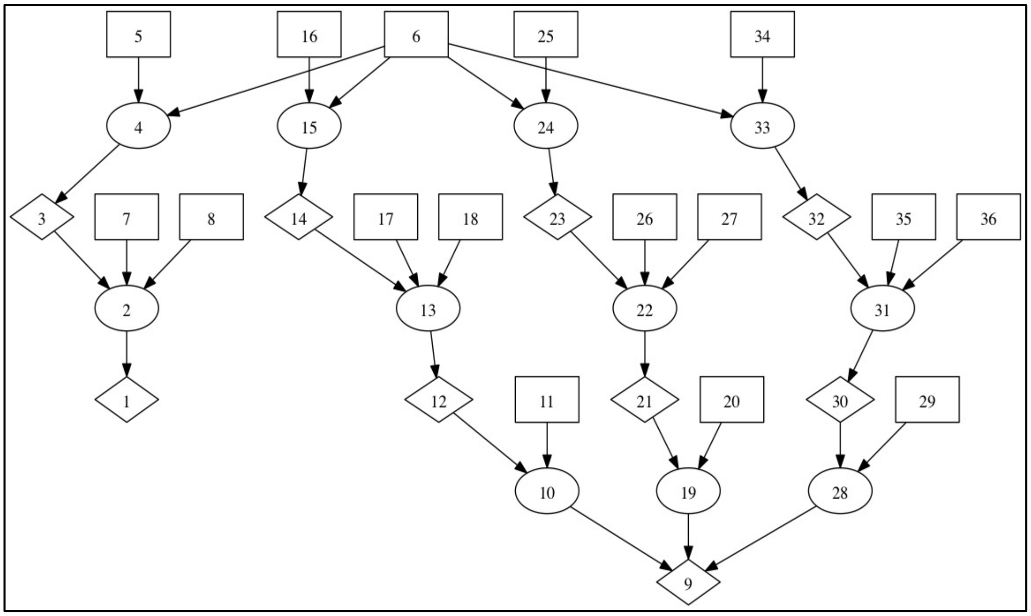

Figure 2 the Experimental Network Attack Scenario Reconstructed from the Alert/Log

in Liu, Singhal, and Wijesekera (2014), we constructed an evidence graph that reflects the attacks as shown in Figure 2, where Appendix 3 describes the notations of all nodes. The graph representation in Figure 2 differs from Definition 1, where a primitive fact node, shown as a box, represents specific network configuration or vulnerability information of a host that corresponds to a unit of evidence. A derivation node, shown as an ellipse, representing a successful application of an interaction rule on (input) facts including primitive facts and prior derived facts substantiated by the evidence obtained after an attack step. A derived fact node is shown as a diamond. There are three constructed attack paths in Figure 2: (1) the attacker used cross-site scripting attack to get administrator's session ID and therefore to get admin privilege (6→4→3→2→1); (2) the attacker used a web application that does not sanitize users' input to launch a SQL injection attack on the database (6→15→14→13→12→10→9); (3) by using buffer overflow vulnerability on the IE browser, the attacker compromised workstations and used them as stepping stones to access the database (6→24→23→22→21→19→9 and 6→33→32→31→30→28→9).

3.3 Rules and Facts Used for Reasoning

Attack scenario generation uses rules that are constructed from generic attack scenarios to analyze the collected evidence and check if the evidence can be correlated together to construct an attack tree in the form of a graph. Appendix 1 shows a snippet of generic attack generating rules, and Appendix 2 has the evidence abstracted in the form of predicates as shown in Appendix 1. The evidence shown in Appendix 2 are primary facts because they correspond to the combination of a specific software vulnerability, computer configuration, network topology, and security policy, which have been successfully exploited in the experimental network to launch the attacks. The meaning of the predicates representing the primary facts is as follows: (1) predicate "attackedHost" represents the destination victim computer; (2) predicate "hacl", which means the host access control list, is used to describe network topology; (3) predicate "advances" represents the access rights within the firewall, which are used by the attacker to reach the next computer after the attacker has comprised a computer; (4) predicate "timeOrder" is used to assure that an attack step's start time and end time falls within a reasonable interval, which would be discussed shortly, and (5) all other evidence collected on a particular host computer or between the source-destination computer pairs are formalized as follows:

"vulExists(workStation1, 'CVE-2009-1918', httpd).
vulProperty('CVE-2009-1918', remoteExploit, privEscalation).
networkServiceInfo(workStation1 , httpd, tcp , 80 , apache). "

4. PREDICATES AND RULES USED FOR EVIDENCE ADMISSIBILITY

Admissibility criteria of evidence place additional constraints on the data and their handling processes, including the chain of custody issues. To ensure that collected evidence satisfies these constraints, we address two issues. First, in a dynamic environment, the evidence changes at a high rate, but forensic tools keep up at a regular rate. In particular, attackers can use anti-forensics tools to obfuscate or destroy the evidence. Second, an individual tool or evidence collection resource may not be able to meet all the needs of a particular investigation. In either case, the admissible evidence should be validated and substantiated using additional constraints including timestamp, relevance, and possibly testing for validatable hypothesis. Admissibility criteria may render the graph in Figure 2 inapplicable, as there isn't proper analysis on each piece of evidence that is used to create the graph. In order to analyze evidence against admissibility constraints, we enhance MulVAL by adding rules that help ascertaining the legal admissibility standards of evidence, which in turn requires adding new predicates.

4.1 Timestamp

One of the attributes useful in modeling legal admissibility check is to have a timestamp on each piece of evidence in order to substantiate the chronological order of the attack steps. We do so by checking if the start time and end time of an attack step represented by the corresponding evidence fall in a *reasonable* timeframe. We say reasonable because exact timestamp values may not be accurate due to system delays in reading clocks and logging mechanisms. Consequently, a distributed timestamp is needed in a distributed enterprise network. Also, attackers may try different techniques until the attack is successful, possibly having multiple attempts to attack the same victim bearing different time stamps from different source computers.

```
1. interaction_rule(
2.          (evidence(H,Perm) :-
3.               execCode(H,Perm),
4.               timeOrder(Source,H,T1,T2),
5.               hold(T1,T2)),
6.            rule_desc('evidence with timestamp',
1.0)).
7.  hold(T1,T2) :-
8.      compare(<,T1,T2),
9.      (T2-T1) @< 0.50.
/*This timeOrder in input fact will instantiate the
timeOrder in line 4 during runtime */
10.
timeOrder(webServer,dbServer,14.3727,14.3729).
11.
timeOrder(workStation1,dbServer,12.2610,14.3730)
```

Figure 3 Predicates and Rules Related to Timestamps

Based on the above reasons, as shown between Line 1 to Line 6 in Figure 3, we implement a new rule "*evidence(H,Perm)*" that uses two new predicates *"timeOrder(Source,H,T1,T2)"*, *"hold(T1,T2)"* and a derived fact predicate "*execCode(H,Perm)*" that represents the attack status after an attack step has been executed to assure that the attack launched from a source computer to a destination computer did occur in a reasonable timeframe. In this rule, the "*interaction_rule*" in Line 1 is a string that uniquely identifies a rule in MulVAL framework, with a description on Line 6. In Line 3, predicate "*execCode(H,Perm)*" says the attacker has successfully attacked the destination computer "H" with the obtained privilege "Perm". In

Line 4 and Line 5, the two new predicates "*timeOrder(Source,H,T1,T2)*" and "*hold(T1, T2)*" specify the attack timeframe. The four variable terms "Source", "H", "T1" and "T2" in "timeOrder" will be instantiated by constants in Line 10 and 11 that are extracted from primary facts input file in Appendix 2 (bold font), where the four terms with constants represent source computer, destination computer, timestamp 1, the start time of the attack from the "Source", and timestamp 2, the end time of the attack to "H". "*hold(T1,T2)*" in Line 5 calls the new added rule "*hold(T1,T2)*" between Line 7 to Line 9, where Line 8 of the rule body, "*compare(<,T1,T2)*", is a Prolog built-in function that checks the validity of T1<T2, and Line 9 uses Prolog built-in functions, "*@<*"(less than), to ascertain the time interval of the attack is less than 50 minutes.

4.2 Computing the Relevance of Evidence

Relevance is important for evidence admissibility. We constructed and used an expert knowledge advisory database to check the evidence relevance of the corresponding attack to minimize false positives. To construct the expert knowledge database, we downloaded vulnerable machine states in the SQL table records as Table 3 from MITRE OVAL (http://oval.mitre.org/) and NIST NVD (http://nvd.nist.gov/). Columns in table 3 include a unified vulnerability name, the operating system, software and its versions that support a successful attack.

Based on our expert knowledge database, we added a predicate "*vulRelevance(Software,Privilege)*" to rules in the rule file "interaction_rules" (Appendix 1) to determine the evidence's relevance. In this new predicate, variable terms "Software" and "Privilege" are instantiated by the data in the "Software" column and "Privilege" column respectively. We use this predicate model if the software configuration on the attacked computer matches "Software" and "Privilege" that are reported to support a successful attack.

```
1. interaction_rule(
2.  (execCode(Host, root) :-
3.      execCode(Host, _Perm2),
4.  vulExists(Host,  _,  Software,  localExploit,
privEscalation),
5.      vulRelevance(Software,Privilege)),
6.   rule_desc('local exploit',1.0)).
```

Figure 4 an Example Rule with Predicate "vulRelevance"

An example rule with the added predicate "*vulRelevance(Software,Privilege)*" is shown in Figure 4. Here, the term "Software" in predicate "*vulExists*"(Line 4) represents the name of the software used in the attacked computer where the evidence has been collected and the term "Software" in predicate "*vulRelevance*"(Line 5) is obtained from the bug report community such as OVAL or NVD. When the specific "*Software*" in predicate "*vulExists*" matches the one in predicate "*vulRelevance*", the relevance between the evidence and corresponding attack can be established and the value of 1 will be assigned to it.

Table 3 Expert Knowledge/Vulnerability Database Obtained from OVAL

Entry	OS	Software	Privilege	Version	Attack Action
CVE-2009-1918	Windows	IE	User	IE 5.01 SP4; IE 6 SP1;IE 6 Win XP SP2	Allows remote attackers to execute arbitrary code via a crafted HTML document

There are two scenarios where the instantiated “Software” in the two predicates “*vulExists*” and “*vulRelevance*” do not match.

(1) The instantiated “Software” in predicate “*vulExists*” is known not to have the vulnerability.

(2) The instantiated “Software” in predicate “*vulExists*” is different from the one in “*vulRelevance*”, but has not been reported to vulnerability advisory database, and hence *assumed* not exploitable to launch a successful attack.

Under condition (1), because relevance is 0, our reasoning won’t generate any attack traces. For condition (2), according to Definition 1, we assign 0.5 out of 1 to relevance, indicating that we need further investigation or may need to simulate the attack to make the relevance judgment better.

4.3 Substantiated Validation vs. Hearsay

As mentioned in federal rules of evidence, except for “admissible hearsay” standard for business records that can be admissible in the court, declared “hearsay”, such as a verbal attack report, cannot be used as evidence. However, so-called “hearsay” evidence could help an investigator to discover an attack not discovered by IDS or logging systems. For example, in our experimental network, through the cross-site script attack on the administrator’s computer, the attacker was able to send out phishing update information to other clients of the portal web service, asking them to update their confidential personal information. Because our intrusion detection system did not detect the phishing attack, a further investigation on the phishing attack should be performed to obtain evidence that satisfies the required level of acceptability standards of admissibility. Here, captured and validated network packets could substantiate the attack by showing that the clients sent confidential information to a malicious computer instead of the portal web service. Also, the investigator could detect that the administrator’s cookie session ID has been stolen as further substantiation of this phishing attack.

```
1. interaction rule(
2. (execCode(Host, root) :-
3.         execCode(Host, _Perm2),
4.     vulExists(Host, _, Software, localExploit,
privEscalation),
5.          vulRelevance(Software,Privilege),
6.               notHearsay(_)),         //will be
instantiated by the input fact
7.   rule_desc('local exploit',1.0)).

8. notHearsay(X):-
9.          \+ hearsay(X).
```

Figure 5 Predicate and Rule Used for Judging Whether Given Evidence is Hearsay

The predicate and rule we implemented for judging declared hearsay or validated evidence are illustrated in Line 6, Line 8 and 9 in Figure 5, where the predicate “*notHearsay(_)*” in Line 6 calls the rule between Line 8 and 9. The rule “*notHearsay(X)*” has a single predicate body, which is “*\+hearsay(X)*” representing the evidence is not declared hearsay. Here, “X” is the source evidence and “\+” is default negation. As mentioned, because the source of evidence has two categories (declared hearsay and validated evidence resource including admissible hearsay, this rule “*notHearsay(X)*” will disqualify the declared hearsay because “\+*hearsay(X)*” returns “false” on declared hearsay. If the evidence “X” is substantiated by validated evidence, “*notHearsay*(*X*)” returns “true”, which can be used to combine other admissibility criteria, such as relevance in Figure 4, to judge the admissibility of validated evidence. Figure 5 shows an example of the combined “*notHearsay*” and relevance admissibility criteria judgment, in which the judgment on relevance (Line 3, 4, 5) has been introduced in 4.3.

4.4 Using Dynamic Clauses to Assist Choosing Admissible Evidence

We use the “*assert(clause)*” and “*retract(clause)*” predicates in Prolog to insert and retract hypothesis as dynamic clauses in our desire to have different explanations of the same evidence or missing evidence.

```
1.   hypo_condition(Hyp):-
2.                  assert(Hyp),
3.                  postCondition.

4.    postCondition :-
5.            execCode(H, Perm).
.... // Actual rule is skipped because of limited space

6.   | ?-
hypo_condition(hacl(internet,webserver,_,_)).
//assert this access control policy
7.   yes
8.     | ?- postCondition.        //postCondition holds
9.   yes
10.   |   ?-   retract(hacl(internet,webserver,_,_)).
//retract this control policy
11. yes
12. | ?- postCondition.     //postCondition does not hold any more
13. no
```

Figure 6 Use "assert/retract" to Assert Hypothetical Conditions

Figure 6 is an example that asserts a network access list to check if the hypothetical precondition results in the post conditions that present the computer status after a particular attack. In this example, in the rule of "*hypo_condition(Hyp)*" (Line 1), the hypothetical condition is asserted (Line 2) to check if the "*postCondition*" (Line 3) of the hypothetical condition will match current computer status represented by "*execCode(H,Perm)*" in Line 5. "*execCode(H,Perm)*" is the body of the rule "*postCondition*" in Line 4, called by the predicate "*postCondition*" in Line 3. With asserted dynamic conditions, the forensic investigators could test different assumptions, and do "what if" analysis in the corresponding reconstructed attack scenario to find the best explanation for evidence admissibility. Our example shows that with the asserted hypothesis "*hacl(internet, webserver, _, _)*" that represents that he attacker can access webserver from the Internet (Line 6-7), the "*postCondition*" matches the current computer status (Line 8-9). Without this access condition (Line 10-11), the current computer status won't be matched (Line 12-13), because the returned result is "false", implying that the attacker must have used the Internet access to attack the webserver.

5. QUANTIFYING EVIDENCE FOR ADMISSIBILITY CHECK

There are situations where there are different explanations on the same evidence or different evidence between the same source-destination computers, because of the following reasons: (1) actions thought benign by IDS could have played a role in a coordinated attack; (2) attackers have launched different attacks, but only one of them succeeded; (3) different experts might have different explanations on the same evidence. If there is no specific test that can be used to determine whether digital evidence possesses the requisite scientific validity, besides using the asserted dynamic clauses to observe the difference as we discussed in 4.4, we use Definition 2 to calculate the evidence's credibility probabilities under different attack scenario explanations so that we can choose one with higher value over the alternatives. The Court in Daubert suggested, in a case like this, the court can inquire the admissibility of the evidence judged by the expert's "principles and methodology" that has been tested, published or widely accepted (Ryan and Shpantzer, 2003).

MulVAL has assigned a probability number to each rule (for example, line 7 in Figure 5), indicating the attack success probability. However, this number is meaningless to the attack scenario constructed by evidence, because the collected evidence could be quite different in even similar attack cases. In order to solve the problem, we use a Java program to read "r","w","h" for each piece of evidence and assign the calculated "p(e)= c(e) × w(e) × r(e) × h(e)" to the corresponding derivation node(a derivation node is an oval node in Figure 2). Here, "w" and "h" are assigned by forensics experts, and "r" is obtained using the method described in Section 4.2. Afterwards, following the constructed evidence graph structure, our Java program calculates "p(h)=p[($\cup e_{out}$)$\cup$ ($\cup e_{in}$)]" for each derived node(the diamond nodes in Figure 2). Our paper has detailed discussion on how to calculate the overall weight probability, so we will skip the detail here.

Because the nature of litigation (civil vs. criminal) places different standards of evidence admissibility, different overall weight standards could be used to evaluate the admissibility/weight of the evidence for different cases (Weiss, 2003). Since this has specific comparison between the legal standards of proof and its corresponding 10-scale Bayesian probability, we used the work in Weiss, C., 2003 and chose the evidence that had a probability larger than "(5) substantial and credible evidence (67-80%)" as the admissibility evidence. As described in our own work, this probability is not fixed. More evidence on the same attack step could increase the weight of the evidence. This has been specifically discussed in Liu, Singhal, and Wijesekera (2013).

6. EXPERIMENTAL RESULTS

By adding predicates and rules to the Prolog based reasoning model and calculating the overall credibility/weight of the evidence and corresponding attacked hosts, we would be able to use the most admissible evidence obtained from a particular attacked network to construct a substantiated attack scenario.

In our experimental network, by using the new added predicates, rules and performing further investigation on declared hearsay in our experimental network, we obtained the new evidence graph in Figure 7, where the attack path (Node 6->Node 33->Node 32 -> Node 31 -> Node 30 -> Node 28 -> Node 9) on "Workstation 2" from Figure 2 has been removed since the evidence is not admissible as discussed before (Firefox in Windows 7 won't support a successful attack by using "CVE-2009-1918" vulnerability that is based on IE browser. Besides, a further investigation did not find any successful attack trace). In addition, a new attack path that is based on the declared "hearsay" of phishing attack has been added to the constructed evidence graph, because a further investigation proved that malicious script had been injected to the forum webpage and "phishing URL" was also found in the clients' computers. This new added attack path is "node 1 -> node 37 -> node 39", which represents that the attacker launched a phishing attack towards the clients by using the compromised administrator's session ID.

The numbers following the last ":" in derivation nodes (oval nodes) and derived nodes (diamond nodes) are "p(e)" and "p(h)" calculated by Definition 2. For node 22, because the attacker could fully compromise workstation by using "CVE-2009-1918", which enabled her to delete all browsing history, a data recovery tool had to be used to recover the deleted evidence. The overall weight for this evidence has been assigned as 0.8, (where "c" was "0.8") because recovered data was not complete and considered as secondary evidence. "w", "r","h" were assigned as 1 respectively.

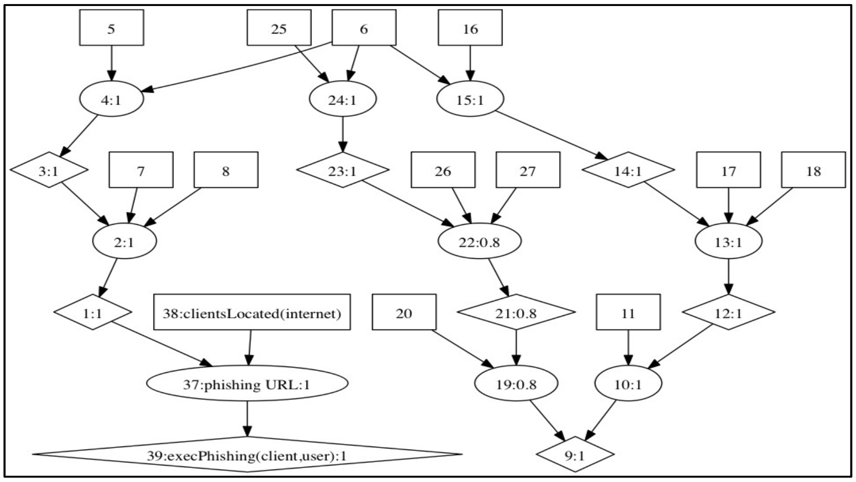

Figure 7 the Constructed Evidence Graph after Admissibility Check

7. CONCLUSIONS AND FUTURE WORK

We extended a prolog based reasoning framework to evaluate evidence's admissibility so that we can construct an attack scenario substantiated by evidence that can rise to a legal standard of acceptability of evidence. Our preliminary experiment shows that we can successfully formalize the legal requirements to prolog rules that can support a given level of acceptability standard of evidence for attack scenario reconstruction. A final test of this claim requires more work in the following aspects. First, we need to formalize the operational security policy into this framework to ascertain if the accesses are legal. Second, we need to find a suitable way to use facts that are already known to be false, so that we can show the inadmissibility of evidence. Third, we plan to test the extended reasoning framework in a real attack scenario and work with cybercrime attorneys to ensure our reasoning can be folded into a formal charge.

REFERENCES

Alferes, J., & Pereira, L.M. (1996). *Reasoning with Logic Programming*. Springer, Berlin.

Balls, M., Amcoff, P., Bremer, S., Casati, S., Coecke, S., & Clothier, R. (2005). The principles of weight of evidence validation of test methods and test strategies. *Altern Lab Anim*, *34*:603-20.

Casey, E. Digital evidence and computer crime. *Forensic Science, Computers, and the Internet*, 3rd ed. Academic Press, 840. ISBN 978-0123742681.

Dain, O., & Cunningham, R. (2001). Building scenarios from a heterogeneous alert stream, Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, 231-235.

Daubert v. Merrell Dow Pharmaceuticals. (1993). Inc., 509 U.S. 579.

Debar H., & Wespi, A. (2001). Aggregation and correlation of intrusion-detection alerts, in *Recent Advances in Intrusion Detection*, LNCS 2212, 85-103.

Eoghan, C. (2002). Error, uncertainty, and loss in digital evidence. *International Journal of Digital Evidence*, *1*(2), Summer 2002.

Federal Rules of Evidence. (2010). Retrieved from http://www.uscourts.gov/uscourts/rulesandpolicies/rules/2010%20rules/evidence.pdf

Keppens J., & Zeleznikow, J. (2003). A model based reasoning approach for generating plausible crime scenarios from evidence. Proceedings of the 9th International

Conference on Artificial Intelligence and Law.

Keppens, J., Shen, Q., & Schafer, B. (2005). Probabilistic abductive computation of evidence collection strategies in crime investigation. Proceedings of the 10th International Conference on Artificial Intelligence and Law.

Kwan, M., Chow, K. P., Law F., & Lai, P. (2008). Reasoning about evidence using Bayesian network, Advances. *Digital Forensics IV, International Federation for Information Processing* (*IFIP*), Tokyo, 141-155.

Liu, C., Singhal, A., & Wijesekera, D. (2012). Mapping evidence graphs to attack graphs. IEEE International Workshop on Information Forensics and Security, December 2012.

Liu, C., Singhal, A., & Wijesekera, D. (2013). Creating integrated evidence graphs for network forensics. IFIP International Conference of Digital Forensics, 227-241.

Liu, C., Singhal, A., & Wijesekera, D. (2014). A model towards using evidence from security events for network attack analysis. 11th International Workshop on Security in Information Systems, April, 2014.

Magistrates' Court at Tuen Mun, Hong Kong Special Administrative Region v. Chan Nai Ming, TMCC 1268/2005, Hong Kong, China 2005. Retrieved from http://www.hklii.hk/eng/hk/cases/hksc/2005/

Ou, X., Boyer, W. & McQueen, M. (2006). A scalable approach to attack graph generation. Proceedings of the 13th ACM Conference on Computer and Communications Security, 336-345.

Ryan, D. J., & Shpantzer, G. (2003). Legal Aspects of Digital Forensics. May, 2014. Retrieved from http://euro.ecom.cmu.edu/program/law/08-732/Evidence/RyanShpantzer.pdf

Sommer, P. (2003). Intrusion Detection Systems as Evidence, Recent Advances in Intrusion detection 1998, RAID98, and electronic version retrieved 17th December 2003.

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

Weiss, C. (2003). *Expressing Scientific Uncertainty, Law, Probability and Risk*, (2), 25-46.

APPENDIX

1. Sample Reasoning Rules

```
primitive(vulExists(_host, _vulID, _program)).
primitive(vulProperty(_vulID, _range, _consequence)).
...
derived(execCode(_host, _user)).
derived(netAccess(_machine,_protocol,_port)).
...

/****          Interaction Rules                    *****/

interaction_rule(
   (execCode(Host, Perm) :-
            principalCompromised(Victim),
            hasAccount(Victim, Host, Perm),
            canAccessHost(Host)),
   rule_desc('When a principal is compromised any machine he has an account on will also be compromised',
   0.5)).

interaction_rule(
 (evidence(H,Perm) :-
         execCode(H,Perm),
          timeOrder(Zone,H,T1,T2),
          hold(T1,T2)),
         rule_desc('evidence with timestamp', 1.0)).

...
```

2. Input Facts in the Form of Predicates That Represent Evidence

```
/*final attack victim*/
attackedHost(execCode(admin,_)).
attackedHost(netAccess(dbServer,_,_)).

/* network topology and access control policy*/
attackerLocated(internet).
hacl(internet, webServer, tcp, 80).
hacl(webServer, dbServer,  tcp, 3660).
hacl(workStation1, dbServer, tcp, 3660).
hacl(workStation2,dbServer,tcp,3660).
hacl(internet, workStation1,_,_).
hacl(internet,workStation2,_,_).
hacl(internet,admin,_,_).
hacl(H,H,_,_).
advances(webServer,dbServer).
advances(workStation,dbServer)

/*time stamps used to find the evidence dependency*/
timeOrder(webServer,dbServer,14.3727,14.3729).
timeOrder(workStation1,dbServer,12.2610,14.3730).


/* configuration and attack information of webServer */
vulExists(webServer, 'CWE89', httpd).
vulProperty('CWE89', remoteExploit, privEscalation).
networkServiceInfo(webServer , httpd, tcp , 80 , apache).

/* configuration and attack information of workStation1 */
vulExists(workStation1, 'CVE-2009-1918', httpd).
vulProperty('CVE-2009-1918', remoteExploit, privEscalation).
networkServiceInfo(workStation1 , httpd, tcp , 80 , apache).

/* configuration and attack information of workStation2 */
vulExists(workStation2, 'CVE-2009-1918', httpd).
vulProperty('CVE-2009-1918', remoteExploit, privEscalation).
networkServiceInfo(workStation2 , httpd, tcp , 80 , apache).

/* configuration and attack information of admin*/
vulExists(admin, 'XSS', httpd).
vulProperty('XSS', remoteExploit, privEscalation).
networkServiceInfo(admin , httpd, tcp , 80 , apache).
```

3. The Notation of Each Node in Figure 2

1	execCode(admin,apache)
2	RULE 2 (remote exploit of a server program)
3	netAccess(admin,tcp,80)
4	RULE 6 (direct network access)
5	hacl(internet,admin,tcp,80)
6	attackerLocated(internet)
7	networkServiceInfo(admin,httpd,tcp,80,apache)
8	vulExists(admin,'XSS',httpd,remoteExploit,privEscalation)
9	netAccess(dbServer,tcp,3660)

10	RULE 5 (multi-hop access)
11	hacl(webServer,dbServer,tcp,3660)
12	execCode(webServer,apache)
13	RULE 2 (remote exploit of a server program)
14	netAccess(webServer,tcp,80)
15	RULE 6 (direct network access)
16	hacl(internet,webServer,tcp,80)
17	networkServiceInfo(webServer,httpd,tcp,80,apache)
18	vulExists(webServer,'CWE89',httpd,remoteExploit,privEscalation)
19	RULE 5 (multi-hop access)
20	hacl(workStation1,dbServer,tcp,3660)
21	execCode(workStation1,apache)
22	RULE 2 (remote exploit of a server program)
23	netAccess(workStation1,tcp,80)
24	RULE 6 (direct network access)
25	hacl(internet,workStation1,tcp,80)
26	networkServiceInfo(workStation1,httpd,tcp,80,apache)
27	vulExists(workStation1,'CVE-2009-1918',httpd,remoteExploit,privEscalation)
28	RULE 5 (multi-hop access)
29	hacl(workStation2,dbServer,tcp,3660)
30	execCode(workStation2,apache)
31	RULE 2 (remote exploit of a server program)
32	netAccess(workStation2,tcp,80)
33	RULE 6 (direct network access)
34	hacl(internet,workStation2,tcp,80)
35	networkServiceInfo(workStation2,httpd,tcp,80,apache)
36	vulExists(workStation2,'CVE-2009-1918',httpd,remoteExploit,privEscalation)