Modeling of Post-Incident Root Cause Analysis

Peter Stephenson, CISSP, CISM, FICAF
Center for Regional and National Security
Eastern Michigan University

Abstract: *In today's business environment the objective of an investigation of a digital incident often is not to trace the incident to its external source and prosecute the perpetrator. Rather, the objective may be determining the underlying root cause that permitted the incident to be successful and applying appropriate countermeasures to prevent its recurrence.*

Because digital incidents are not always from an external source, the focus often is upon the internal network and the people who use it. Frauds, abuses and other insider threats are, by most accounts, more common than externally caused events. In any event, such root cause investigations usually center upon the internal network, the entry point (especially in large-scale virus and worm infections) and the damage done during the incident.

A characterization of virtually all such investigations would necessarily include the potential for cover-ups by individuals who feel threatened by the outcome (whether because they caused the event or because they failed to prevent it), the complexity of the spread of the consequences of the attack in a complex enterprise and the limitations on the training of internal investigators. Clearly, therefore, an important objective of the investigation is a high level of confidence that the outcome of the investigation accurately describes the actual events being investigated.

This paper discusses an approach to post-incident root cause analysis of digital incidents (also sometimes called digital post mortems) that has structure and rigor and the results of which can be modeled formally using Colored Petri Nets. The ability to model the investigation and its outcome lends materially to the confidence that the investigation truly represents the actual events. The approach is best suited to large, complex investigations, and we use a case study of a SQLSlammer worm infection on a reasonably large multinational enterprise as a working example.

BACKGROUND

Late in January of 2003 a worm variously known as SQLSlammer and Sapphire was released into the wild. It flourished on the Internet faster than any other worm or virus in the Internet's history. Internet storm centers around the world watched in amazement as the worm literally took over the Internet in a matter of hours.

SQLSlammer had a very small packet footprint (376 bytes) and no payload. It selected addresses to infect pseudo randomly and it attacked Microsoft SQL Servers on UDP port 1434. A patch for the enabling vulnerability had been issued in July of the previous year.

Shortly after the release of the worm a large financial services company was infected. The worm spread throughout the organization's 27,000 user network in the United States in under ten minutes, completely flooding the network. It was noticed, although it did no damage, in the company's international subsidiaries.

An investigation did not reveal the exact root cause of the internal infection due to lack of detailed evidence. However, the writer modeled the investigation and some important conclusions may be drawn from that model.

Obviously, for privacy reasons, the victim information in this paper has been disguised. However, the architecture of the victim enterprise enters into the speed with which the worm was able to spread.

For simplicity, we focus upon the overall facts of the investigation and the spread of the worm within the victim enterprise. We omit many of the investigation details that could compromise the victim's privacy.

Victim's Enterprise Architecture

The architecture of the victim network used a combination of switches and core routers. The network core was a very high speed routed ring with limited safeguards in place. The network was designed to be high availability and, from a security and compartmentalization perspective, could be considered flat. It comprised a single security policy domain and virtually no internal firewalls.

In addition to the internal network, the victim enterprise had a perimeter network containing a large web farm, several extranets connecting to business partners and an internal wireless network. The extranets and the wireless network were not explicitly firewalled into the internal LAN. The perimeter network was solidly firewalled, using a bastion architecture with multiple firewalls both within the perimeter network and between the perimeter and the internal LAN.

The enterprise had SQL Server installed pervasively throughout the enterprise to provide the database support for the organization and its web applications. SQL developers had access to the enterprise through dial-up which, likewise, had no firewall isolation to the internal LAN.

Clearly, there were numerous ways for the worm to enter the system. However, the challenge for the investigation was to determine the exact point of entry and apply appropriate countermeasures.

Internal Political Considerations

The victim organization, when analyzed from a business (as opposed to technical) perspective revealed several internal political impediments to a successful defense against the worm.

Primary among these political issues was the reluctance on the part of SQL developers to take their servers down and patch them. They viewed the threat as trivial when the vulnerability that enabled the worm's success was announced nearly six months earlier.

A secondary consideration was the lack of communication between the information security department and the IT department. Although the security group reported the vulnerability when it was announced initially, the IT department naively believed that simply configuring external firewalls to deny UDP port 1434 would be adequate protection for the enterprise.

This paper focuses upon technical analysis and countermeasures, but the reader should bear in mind that much of the technical weakness derived from operational considerations.

THE INVESTIGATIVE PROCESS

In this paper we focus upon the investigative approach and the modeling of the outcome. For reasons of space and victim privacy we omit many investigative details.

As starting points we have developed an approach to digital investigation called End-to-End Digital Investigation (EEDI), an investigative process language called the Digital Investigation Process Language (DIPL) and an implementation of the EEDI process created explicitly for digital post mortems.

The underlying basis for the EEDI process is the Digital Forensic Research Workshop (DFRWS) investigative framework (the "Framework"). We describe those investigative components in the following sections.

Problem Statement

There has, to date, been demonstrated no structured methodology for conducting digital investigations and no formal approach to modeling such an investigation. Due to the nature of digital incidents, such structure and formality would be useful. Forensic digital analysis is unique among the forensic sciences in that it is inherently mathematical and generally comprises far more data from an investigation than is present in other types of forensics. In this paper we describe one approach to solving these problems and give an example of its use in an actual digital post mortem investigation.

The DFRWS Framework

The DFRWS Framework is a consensus developed between 2001 and 2003 by an ad hoc group of researchers and practitioners in digital forensic science. The framework is shown in Figure 1 below.

IDENTIFICATION	PRESERVATION	COLLECTION	EXAMINATION	ANALYSIS	PRESENTATION
Event/Crime Detection	**Case Management**	**Preservation**	**Preservation**	**Preservation**	Documentation
Resolve Signature	Imaging Technologies	**Approved Methods**	**Traceability**	**Traceability**	Expert Testimony
Profile Detection	**Chain of Custody**	Approved Software	Validation Techniques	Statistical	Clarification
Anomalous Detection	**Time Synch.**	Approved Hardware	Filtering Techniques	Protocols	Mission Impact Statement
Complaints		**Legal Authority**	Pattern Matching	Data Mining	Recommended Countermeasure
System Monitoring		Lossless Compression	Hidden Data Discovery	Timeline	Statistical Interpretation
Audit Analysis		Sampling	Hidden Data Extraction	**Link**	
		Data Reduction		Spatial	
		Recovery Techniques			

Figure 1 - DFRWS Digital Investigative Framework

Each of the columns represents a Class of actions to be taken in a digital investigation, while the rows represent Elements of the Classes. The elements in **bold** type represent required elements. This model is discussed in detail in the report of the DFRWS for 2001[1].

Briefly, the descriptions of the Classes are:

Identification Class

The identification class describes the method by which the investigator is notified of a possible incident. Since about 50% of all reported incidents have benign explanations, processing evidence in this class is critical to the rest of the investigation. Likewise, as it is the first step in the EEDI process, it is the only primary evidence

[1] Digital Forensics Research Workshop. "A Road Map for Digital Forensics Research 2001." Digital Forensics Research Workshop 6 November (2001): http://www.dfrws.org

not corroborated directly by other primary evidence. Therefore, a more significant amount of secondary evidence is needed to validate the existence of an actual event[2].

The DFRWS definition from the 2003 workshop is:

> *"Determining items, components and data possibly associated with the allegation or incident. Perhaps employing triage techniques."*[3]

Preservation Class

The Preservation Class deals with those elements that relate to the management of items of evidence. The DFRWS describes this class as "…a guarded principle across 'forensic' categories." The requirement for proper evidence handing is basic to the digital investigative process as it relates to legal actions.

The DFRWS definition from the 2003 workshop is:

> *"Ensuring evidence integrity or state"*

The Collection Class

The Collection Class is concerned with the specific methods and products used by the investigator and forensic examiner to acquire evidence in a digital environment. As has been noted, the Preservation Class continues as an element of this Class. With the exception of the Legal Authority element, the elements of this class are largely technical.

The DFRWS definition from the 2003 workshop is:

> *"Extracting or harvesting individual items or groupings."*

The Examination Class

The Examination Class deals with the tools and techniques used to examine evidence. It is concerned with evidence discovery and extraction rather than the conclusions to be drawn from the evidence (Analysis Class). While the Collection Class deals with gross procedures to collect data that may contain evidence (such as imaging of computer media), the Examination Class is concerned with the examination of that data and the identification and extraction of possible evidence from it. Note that the Preservation Class continues to be pervasive in this class.

The DFRWS definition from the 2003 workshop is:

> *"Closer scrutiny of items and their attributes (characteristics)"*

The Analysis Class

The Analysis Class refers to those elements that are involved in the analysis of evidence collected, identified and extracted from a gross data collection. The validity of techniques used in analysis of potential evidence impact directly the validity of the conclusions drawn from the evidence and the credibility of the evidence chain constructed therefrom. The Analysis Class contains, and is dependent upon, the Preservation Class and the Traceability element of the Examination Class.

---

[2] Stephenson, Peter R. Structured Investigation of Digital Incidents in Complex Computing Environments. PhD Thesis, 2003: unpublished

[3] Digital Forensics Research Workshop. "Day 1-DF-Science, Group A Session 1 (D1-A1), Digital Forensic Framework." Work notes for the 2003 Digital Forensics Research Workshop – unpublished 6 August 2003

The various elements of the Analysis Class refer to the means by which a forensic examiner or investigator might develop a set of conclusions regarding evidence presented from the other five classes. As with all elements of the Framework a clear understanding of the applicable process is required. Wherever possible, adherence to standard tools, technologies and techniques is critical. Finally, when mapping this class to the DIPL or when performing model checking, we are concerned solely with the process, not the results of the analysis or the detailing of the contents of evidentiary items.

The Link element is the key element used to form a chain of evidence. It is related to traceability and, as such, is a required element.

The DFRWS definition from the 2003 workshop is:

"Fusion, correlation and assimilation of material to for reasoned conclusions."

The Presentation Class

This class refers to the tools and techniques used to present the conclusions of the investigator and the digital forensic examiner to a court of enquiry or other finder of fact. Each of these techniques has its own elements and a discussion of expert witnessing is beyond the scope of this thesis. However, for our purposes we will stipulate that the EEDI process emphasizes the use of timelines as an embodiment of the Clarification element of this class.

The DFRWS definition from the 2003 workshop is:

"Reporting facts in an organized, clear, concise and objective manner."

These six classes define, in a broad sense the digital investigative process. From this Framework we develop additional definition and apply additional structure to the process.

The End-to-End Digital investigation Process

The End-to-End Digital Investigation process is a collection of generalized steps to be taken in conjunction with the DFRWS Framework. While the Framework gives a roadmap for addressing those issues comprising a structured investigation, the EEDI process provides a set of steps the investigator must perform in order to preserve, collect, examine and analyze digital evidence.

The End-to-End process details consist of:

- Collecting evidence
- Analysis of individual events
- Preliminary correlation
- Event normalizing
- Event deconfliction
- Second level correlation (consider both normalized and non-normalized events)
- Timeline analysis
- Chain of evidence construction
- Corroboration (consider only non-normalized events)

In order to understand our explicit meanings we offer some definitions of terms used in the above process.

Digital Forensic Correlation

Correlation, in the context of a digital forensic investigation refers to the comparison of evidentiary information from a variety of sources with the objective of discovering information that stands alone, in concert with other information, or corroborates or is corroborated by other evidentiary information.

Correlation is sometimes referred to as fusion, the process by which all of the available evidentiary data is analyzed and correlated into a single consistent representational model such as a timeline. Correlation uses the processes of forensic normalization and deconfliction.

Normalization we define as the combining of evidentiary data of the same type from different sources with different vocabularies into a single, integrated terminology that can be used effectively in the correlation process.

Deconfliction is the combining of multiple reportings of the same evidentiary event by the same or different reporting sources, into a single, reported, normalized evidentiary event.

Procedurally, we begin by charting out the steps in the investigation using a narrative. The narrative may be created before or after the investigation depending upon the purpose of the analysis.

If the purpose is the design of an investigative process, we develop the model prior to conducting the investigation. This approach also is appropriate for creating a model framework for future use and for developing simulations of investigative direction. When analyzing an existing investigation, we apply the narrative of the actual steps taken to the framework in preparation for modeling the actual investigation.

Once the narrative is complete, we translate it into the Digital Investigation Process Language (DIPL). The DIPL is a semi-formal process language derived from the Common Intrusion Specification Language (CISL) which, in turn, was evolved from Lisp.

The final step is the formal modeling of the investigation using the DIPL as input for a formal modeling program. For our purposes we have selected Colored Petri Nets (CPN).

The process offers the ability to identify investigative process flaws that could compromise the investigation procedurally, or could lead to developing flawed evidence or missing important evidence. The chain of evidence developed in the EEDI process depends upon the full, complete and correct use of the process from beginning to end.

There remains only to develop a structured method of recording the investigative process in detail, either pre- or post-incident. This structured method should allow for a formal analysis and modeling of the investigation, its outcome and potential future directions. We accomplish this structure with the Digital Investigation Process Language[4].

The Digital Investigation Process Language

The Digital Investigation Process Language's purpose is to allow a structured description of the investigative process, whether planned or actually executed. The language provides for a semi-formal description of the investigative process using a vocabulary of Semantic Identifiers (SIDs) arranged in s-expressions[5] to describe tasks, functions and actions. A simple example of DIPL code is shown below.

```
(ManageCase
    (Initiator
        (RealName 'Joe Investigator')
    )
    (Link '123.221.3.2', '222.122.56.4')
    (Data
        (ChainOfCustody 'Joe
         Investigator')
```

[4] Stephenson, Peter R. "DIPL Semantic Identifier Vocabulary" language description document. Unpublished 2003

[5] SEXP---(S-expressions). Ronald Rivest. 3 May 1997. MIT. 16 April 2003 <http://theory.lcs.mit.edu/~rivest/sexp.html>.

```
        (CaseName A237-4)
        (EvidenceID A237-4-1)
    )
    (BeginTime [16:54:23 GMT 03052002])
    (EndTime [21:13:45 GMT 03052002])
    (Comment 'This is the source and
    destination of the attack from
    firewall logs on FW-3. The logs are
    entered as evidence.')
    (When
        (Time [09:30:00 GMT 05052002])
    )
)
```

Figure 2 - Example of DIPL Code Showing Evidence Management

In this fragment, an example of case management, Joe Investigator has taken evidence, in the form of a log from a firewall called FW-3, which shows a link between two IP addresses. He has placed the evidence into chain of custody as the custodian, named the case and numbered the evidence. The log covers the period from 16:54:23 GMT 3 May 2002 to 21:13:45 GMT of the same date. The evidence is time/date stamped 09:30:00 GMT 5 May 2002.

Colored Petri Net Modeling

The use of Colored Petri Nets leads to a formal model, and in some cases, a simulation of the investigation and its possible outcomes. For our purposes, we have used a CPN modeler called Design/CPN.[6]

We can use the modeler in two ways. First, we can create a model of the actual DIPL investigation. Often, this does little more than create the formal model of the actual process. It can, however, be used to simulate the probable outcome of a set of investigative actions. This capability promises to be of significant value in the investigative process.

The second approach is to create a CPN template of an acceptable investigation and load the actual DIPL process into it. Where the actual process has failed, the modeler will fail as well showing clearly where the process flaw exists.

Because the focus of this paper is the modeling of the outcome, we will not dwell upon the EEDI steps that led up to the point where we could model the outcome. We will, however, summarize the findings of the structured EEDI process. The generalized overall process flow to a formal model is shown below.

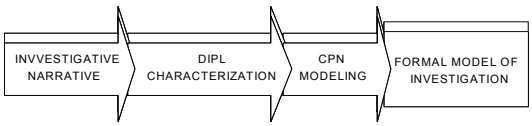


Figure 3 - Generalized EEDI Process Flow to a Formal Model

THE INVESTIGATION

The post event root cause investigation of the SQLSlammer worm was conducted on an enterprise network supporting some 27,000 users in two countries. The network was designed to be a high performance, high availability network in the style of sophisticated service provider infrastructures.

The purpose of the investigation was to determine the root cause of the infection of the enterprise and predict some appropriate countermeasures to prevent a reoccurrence.

[6] http://www.daimi.au.dk/designCPN/

The investigative team comprised technical, investigative and business process specialists. The team collected logs, conducted interviews and compiled basic information in several general categories:

- Logs from monitoring devices
- Logs from hosts and servers
- Interviews with involved personnel
- Interviews with business and technical managers
- Device configuration files
- Network maps
- Event observation timelines
- Notes of relevant meetings
- Response team notes and observations

Once the available information was collected, the team performed detailed analysis in three main areas:

- Technical data analysis
- Business process analysis
- Regulatory compliance analysis

The general procedure followed the EEDI process. The investigative specialist (this writer), after characterizing its findings using the DIPL, found that there were several impediments to completing the investigation successfully:

- Many logs were missing for the period of the incident
- Interviews uncovered conflicting statements from interviewees
- Critical devices, such as edge routers, had been purged following the incident
- Intrusion detection logs could not be extracted from their digital formats or post-event analysis
- A project manager refused to allow the investigation to proceed when he found that the investigating team was nearing an explanation
- Fundamental business processes necessary to allowing a successful investigation were not in place

The investigative specialist was able to generate a DIPL characterization of the incident at a sufficiently high level to point to several potential key root causes:

- Lack of firewalls between the corporate VPN, the corporate wireless network, the corporate extranet, and the internal network
- SQL servers not patched
- Lack of policy enforcement
- Naïve view of layered security requirements
- Flat network security architecture

From those preliminary findings, and the information that generated them, the specialist was able to model, mathematically and using Colored Petri Nets, the root causes and the countermeasures that would have mitigated them.

Preliminary Modeling

The first step in creating a CPN model of a completed investigation is understanding, formally, the issues that will be included in the model. This understanding starts with the output of the EEDI process and is translated into the appropriate formalisms. The DIPL is useful for structuring the understanding obtained using the EEDI process into a structured characterization. The issues that must be addressed include:

- The victim system environment
- The nature of the attack or incident
- The impact of the incident upon the pre-attack environment

- The changes in state of the victim environment caused by the incident
- The countermeasures that would have prevented the state change

We address each of these in the sections that follow.

THE VICTIM SYSTEM ENVIRONMENT

We only need to characterize the victim system at a high level for our purposes. We are concerned with understanding the security policy domains that comprise the security architecture, not the details of the network design.

The notion of policy domains is fundamental to security architecture. Once the policy domains have been defined, physical and logical controls can be put in place to enforce them. Policy domains are defined, simply, as logical and/or physical divisions of the enterprise that are governed by a single security policy.

The controls imposed upon the network architecture must be adequate to enforce the security policy upon that architecture. These controls can be of multiple types including system or device access controls, subject authorizations relating to specific objects (i.e., access control lists and object access privileges), etc., and may be enforced in a wide variety of manners.

We have stated that the investigation determined that the victim enterprise was a single (flat) policy domain. However, that was, strictly speaking, not the case if one considers the enterprise and its interconnections such as extranets, perimeter networks and wireless LANs.

These other policy domains, however, were implicit rather than explicit in the sense that there were not explicit policies written for these domains beyond those for the internal enterprise. With that in mind, we define two policy domains for the overall enterprise: public and private.

The private domain is the single, flat, domain that describes the internal enterprise, while the public domain is everything else. We, therefore, define, formally, the enterprise environment in terms of these two policy domains.

Finally, we concern ourselves only with those aspects of the policy domains that impact, directly or indirectly, upon the incident under investigation. This limits the universe of issues with which we must contend in our analysis and gives us a bounded environment to model.

The Policy Domains and the Nature of the Incident

We begin by characterizing the relevant portions of the private security policy domain. In the context of an external virus or worm attack, we view the internal, or private, policy as, first, preventing access by the virus or worm from the public domain. In order to do that, we must describe the threat against which we must protect the internal network.

That requirement leads to the second issue listed above: *the nature of the attack or incident*. In this case the attack is the SQLSlammer worm. We know that it is a worm that penetrates UDP port 1434 and infects both Microsoft SQL servers and Microsoft MSDE ("Microsoft Database Embedded") devices. MSDE is a light weight database embedded within many Microsoft and third party systems including some Cisco device management systems. It is likely that more damage was done due to MSDE infections than MSSQL infections.[7]

[7] http://www.robertgraham.com/journal/030126-sqlslammer.html

With this in mind the hardening of the entry points in the demark between the private and public policy domains is critical. While patching the SQL servers could also have been important, once the worm entered the enterprise it would have spread rapidly whether or not the SQL servers were patched. Additionally, when the worm attempts to infect an address that does not have UDP port 1434 open, or an address that does not exist within the enterprise, packets of different types are generated. For example, attacking a device without UDP port 1434 open can result in an ICMP return. Thus, the number of packets being generated on the enterprise is only partially a result of the number of infected devices. These facts lead to the conclusion that if the worm is able to enter the enterprise at all, it very likely will bring the network down. We can characterize this by:

Let A_{dos} = the set of all denial of service attacks
Let x = a class of denial of service attack that can cause a network flood
Let s = the SQLSlammer attack
Let p = an open port
Let y = a successful denial of service attack
Let d = a target device
Let v = a vulnerable device

$$\exists x : A_{dos} \mid \{f(x) = s, f(p)= \text{UDP1434}\} \rightarrow y \equiv f(d) = v$$

We define the function **f( )** as being some function applied to, for example, a class of denial of service attacks (**f(x)**), that causes a change in the attacks. Thus, f(x) = s is some function that when applied to the class of denial of service attacks **x** yields a SQLSlammer attack. f(p)= UDP1434, therefore, is some function that causes the general class of open ports, **p** to select UDP port 1434 specifically. It is not important to understand, at this stage, exactly what that function is. We only need know that it exists.

If there is a path to a vulnerable device it will lead to a successful denial of service attack by the SQLSlammer worm. Note that this says nothing about patched devices. It simply requires that the device be vulnerable. Since it is possible to have devices (MSDE) that are not patchable, the nature of the attack depends upon only two things: the presence of vulnerable devices and a path to them.

This is a very important statement because it establishes, mathematically, the ground rules for a successful attack. A system whose environment fits the description of one against which the attack will succeed must, therefore, be vulnerable, and will, inevitably, succumb.

We proceed to define the victim's system environment in terms of, (A), its security policy domains, and (B), those domains in terms of the relevant issues described in the above requirements for a successful attack. Note that all issues outside of this system environment (i.e., political and business issues), while important to the investigation, have no direct bearing upon the modeling of the success or failure of the attack.

We derive the mathematical statement that describes the private domain and its relationships with the public domain based upon the results of the EEDI investigation. While the EEDI investigation determined that there were several different environments that qualify as part of the public policy domain (VPN, wireless, etc.) we treat them all as a single domain for the purposes of a mathematical description.

Let S_{priv} = the set of security policies describing the victim's private security domain
Let s = an individual security domain policy
Let p = an open port
Let d = a target device
Let v = a device vulnerable to the SQLSlammer worm

$$\exists s : S_{priv} \mid f(s) = \{f(p) = \text{UDP1434}, f(d) = v\}$$

There is a security domain policy that allows an open UDP port 1434 and devices which exhibit the vulnerability to the SQLSlammer worm. We know that this is true because the EEDI investigation revealed that there was no firewall between systems in the public security domain (i.e., VPNs, wireless networks and extranets) and the private security domain.

The interpretation of "security domain policy" as used here is the actual policies (configurations) of devices separating the domains (or lack thereof). We also know that, based upon the combination of these two characterizations, a SQLSlammer attack upon the private security domain from the public security domain would have to succeed.

THE IMPACT OF THE INCIDENT UPON THE VICTIM

Referring to the literature on the SQLSlammer worm (see footnote 7 as an example) we can see that a vulnerable, infected enterprise will succumb in a very short (minutes) period of time. Therefore, we can expect a state change in the victim enterprise from a pre-event state of proper operation to a post event state of the network being completely flooded with packets resulting from the infection if (A) the victim enterprise can be made to change state, and (B) the enterprise is vulnerable to the SQLSlammer attack.

That there is something that can make the system change state is intuitively obvious from:

$$s_1 \in S_s \mid f(s_1) \rightarrow s_2$$

There is a correct operating state of the system, s_1, a member of the set S_s of all operating states of the system, such that some action $f(s_1)$ can cause it to change states to a new state, s_2. What that action is must be defined separately.

However, we must take this statement as true in the general case since no system enjoys perfect protection against state changes of some sort. This does not, necessarily, mean that the state of the target system will change to a failed state. Whether that occurs or not depends upon the nature of the cause of the state change.

It also does not necessarily mean that the victim enterprise will change state as a result of the SQLSlammer attack. That must be determined by the statements above describing the victim's vulnerabilities and the nature of the attack. From a practical perspective, however, we can see that a catastrophic effect upon the victim in this investigation is very likely.

COUNTERMEASURES

The final issue we must address is that of predicting the countermeasures that would have protected the victim enterprise. From the above it is clear that causing any of the preconditions for a successful attack to fail would describe a successful countermeasure.

Since we cannot cause the SQLSlammer worm to "un-exist", we must focus upon correcting those elements of the private security domain policy that permitted the success of the attack. That allows us to focus upon two specific areas: the path and the individual device vulnerabilities.

From the investigation we know that an attempt was made to mitigate the path vulnerability from one portion of the public security domain: the connection between the public Internet and the protected enterprise. However, we also know that there were at least four ways that the public security domain could have had unfettered access to the private security domain and the vulnerable devices contained on it.

Mitigation of the vulnerability on the vulnerable devices would not, based upon the literature, have been of much value since of nearly 2,000 devices identified during the investigation as vulnerable, only about 500 were patchable SQL Servers. In order for the patch to be successful it would need to have been placed on all 2,000 vulnerable devices, a clear impossibility. The mathematical statement in the above section tells us that as long as the vulnerability exists on even a single device, the enterprise contains vulnerable devices, and if a path to the vulnerable devices exists, the attack will succeed. Since we cannot mitigate the vulnerability with anything close to 100% confidence, we must concentrate upon the path.

Therefore, countermeasures should focus upon closing all paths between the public security policy domain and the private security policy domain that the attack can exploit. The nature of those paths is clear from the mathematical statements describing the nature of the attack. There is no particular benefit, in this case, in patching the SQL servers.

PETRI NET MODEL

Once the characterization of the security policy domains, the nature of the incident and the appropriate countermeasures is complete, a Petri Net model of the actual incident can be created. This does not attempt to characterize the technical details of the incident, but, rather, the state changes that occurred during the course of the event.

The objective is to show clearly what state changes occurred and what countermeasures would block those state changes. Used in reverse, this approach can assist the investigator during the investigation in simulating probable outcomes as the investigation proceeds.

Fundamentally, the Petri Net model for this investigation consists of three parts:

- The existence of the attack
- The attempted delivery of the attack against the victim
- The effect of countermeasures

We address each of these in turn and conclude with an overall Petri Net of the results showing appropriate state changes. Note that we have simplified this representation somewhat due to space considerations. The investigator can take this model to whatever level of detail he or she wishes.

Describing the Attack

We begin with the global declarations that we will use throughout this model.

Declarations:

```
color Attacks = with sqlslammer_attack | other_attack;
color slammer_selected = Attacks;
color Successful = Attacks;
color Inhibitors = with configured | not_configured;
color Inhibited = Inhibitors;
var attack : Attacks;
var countermeasure : Inhibitors;
var perimeter : Inhibitors;
var vpn : Inhibitors;
var wireless : Inhibitors;
var laptop : Inhibitors;
```

The Petri Net of the attack is fairly simple:

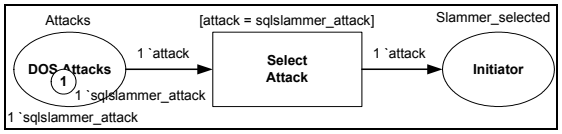


Figure 4 – Attack Petri Net

In this Net, we show that The *DOS Attacks* input place is colored *Attacks* and has the choice of a SQLSlammer attack or some other type of denial of service attack. The initial marking is *1 sqlslammer_attack* and there is a single *sqlslammer_attack* token ready to release.

We see that the *Select Attack* transition will fire if the *attack* variable is a *sqlslammer_attack* token. Since it is, we will see that the *Initiator* output place will contain the *sqlslammer_attack* token from the input place and will be ready to attempt to deliver the token to the victim as we see next.

Describing the Attack Delivery

The Petri Net for the attack delivery begins with the *Initiator* place becoming the input place for the *Deliver Attack* transition. The single *sqlslammer_attack* token now resides in the *Initiator* place.

Again, the Net is straightforward:

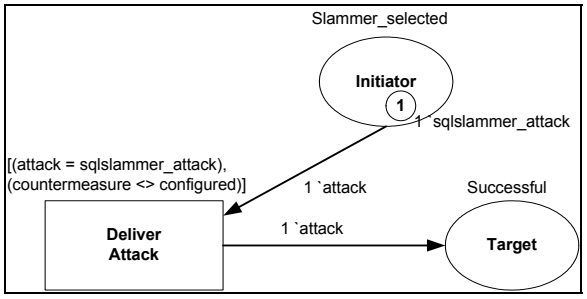


Figure 5 – Attack Delivery Petri Net

The *Initiator* input place is armed with the *sqlslammer_attack* passed by the *Select Attack* transition and is ready to deliver it to the *Deliver Attack* transition. The transition will fire if the *attack* variable is a *sqlslammer_attack* token and the *countermeasure* variable is anything except *configured* token.

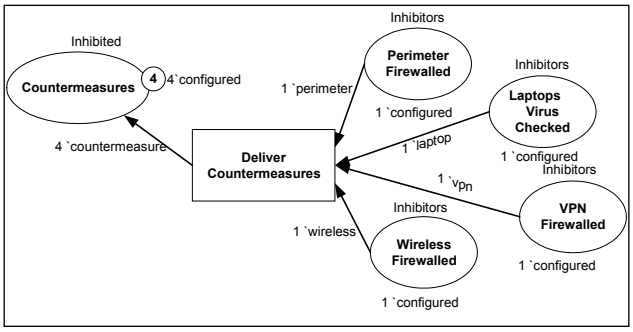
If the transition fires, the *sqlslammer_attack* token will be delivered to the *Target* and the attack will be successful. Although there could be several tokens satisfying the *countermeasure* variable, it takes only one that is not a *configured* token to allow the attack to succeed. It remains for the countermeasures on the victim to prevent the attack from being delivered.

The Effect of Countermeasures

The objective of the countermeasures in any enterprise is the interdiction of an attack attempt. In the Petri Net model we show that by setting a transition to deliver appropriate countermeasures to the *Deliver Attack* transition.

These tokens must represent all failure modes (in this case delivery paths that fit the attack description and apply to the victim's security domain policy configuration) and we must simulate the results of a failure of any of these countermeasures.

The Petri Net for this is:



In this Net there are four input places that characterize the paths within the victim enterprise between the public security policy domain and the private domain. These are taken from our investigation. The state of those input places will depend upon what the investigation finds them to be. Assuming that all of the paths have appropriate security policies configured, each will deliver a *configured* token to the *Deliver Countermeasures* transition.

Figure 6 – Countermeasures Petri Net

There is no guard on the *Deliver Countermeasures* transition, so it will fire on every token. In Figure 6 we show that the initial marking of the four input places is *configured*. Thus the transition will deliver only *configured* tokens to the *Countermeasures* place. The *Countermeasures* place acts as an input place for the *Deliver Attack* transition in Figure 5 and, since there are no tokens not of the type *configured* the attack will fail.

Should there be even a single *not_configured* token, however, the guard on the *Deliver Attack* transition will be satisfied and the transition will fire, delivering the successful attack to the *Target* output place. The overall Petri Net is shown in Figure 7. This Net shows the attack failing due to the correct configuration of the countermeasures.

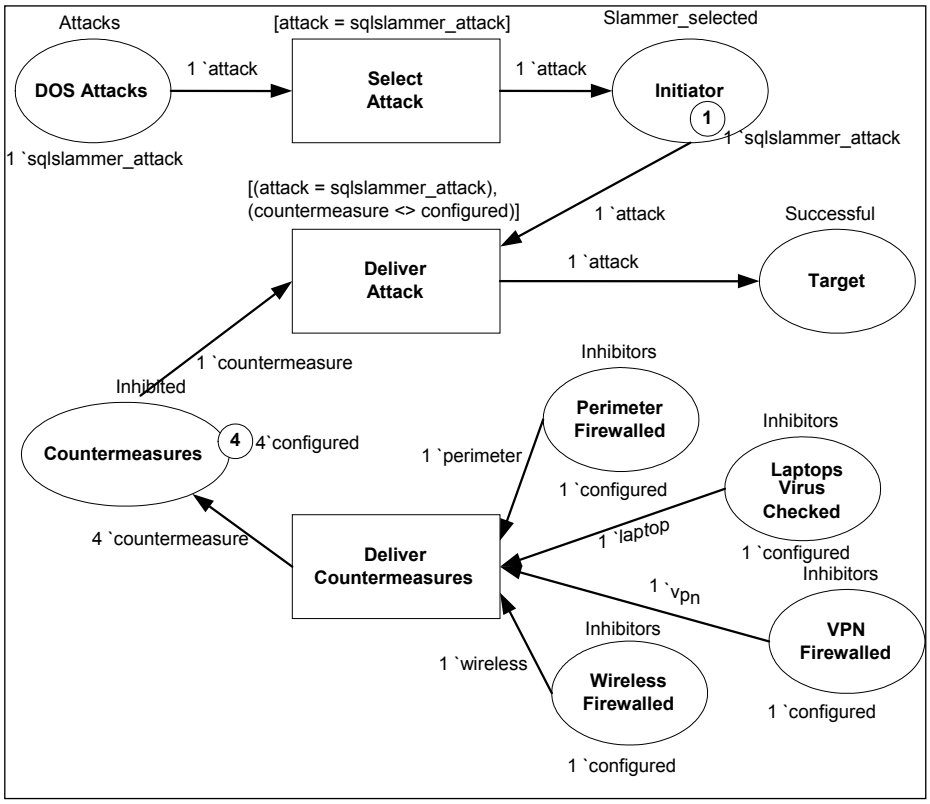


Figure 7 – Complete Post-Event Analysis Petri Net

CONCLUSIONS

From this case study we show that it is feasible to model the expected outcome of an incident given the results of a structured investigation. There are some interesting implications of this conclusion.

First, given the ability to model the outcome of an actual event, it should be equally feasible to model the outcome of a threat/vulnerability assessment before a catastrophic attack occurs. This use for the EEDI technique could allow the organization to take a proactive stance against attacks and to avoid expensive, but ineffective, countermeasures.

Second, since there is a point in every investigation when the victim system's architecture and the nature of the incident are well known, modeling the probable outcome of a line of inquiry should be feasible.

Finally, if, given the information gathered in the structured investigation, the modeling effort fails utterly, there may be the implication of false, misleading or missing evidence.

Future Work

There are many avenues for continued research in the areas described in this paper. For example, the proactive modeling of investigative outcome (intended to be suggestive of new chains of investigation) presents some interesting challenges.

A major area for future work is the formalization of the underlying mathematics describing the various processes. While the Colored Petri Net representations may be classed as formalisms due to the formal proofs that are the basis of CPNets, it is appropriate to extend the logic used in describing the investigation results leading up to the modeling stage such that it describes, accurately and formally, what is happening.

The examples used in this paper are preliminary and may not fully describe some of the actions occurring in complex investigations. It is the author's intention to revisit this underlying logic and create an extension to the standard logic and set theory as used here for illustrative purposes. Standard logic and set theory alone are insufficient to describe complex forensic investigations clearly and completely.

Additional refinement of the modeling process using approaches such as CSP instead of Colored Petri Nets may offer opportunity for a more detailed and predictive analysis. The author believes that such refinement could improve the mathematical representations of the investigative and forensic processes as well as their end results materially.

The conducting of the EEDI process across large public networks such as the Internet poses significant challenges, especially in the area of trace back.

ABOUT THE AUTHOR

Peter Stephenson is a writer, consultant, researcher and lecturer in information protection and forensics. He has spoken extensively on digital forensics and security, and has written or contributed to 14 books and several hundred articles in major national and international trade publications. He has lectured and delivered consulting engagements for the past 17 years in eleven countries plus the United States.

He is the developer of an operational framework for information protection, as well as structured methods for vulnerability assessment, and standards-based security architecture requirements engineering. He developed the end-to-end approach to digital incident investigation and the Digital Investigation Process Language (DIPL). Mr. Stephenson holds a BSEE and currently is completing his PhD at Oxford-Brookes University in Oxford, UK where his research involves structured investigation of information security incidents in complex computing environments. Mr. Stephenson is a research scientist at Eastern Michigan University's

Center for Regional and National Security, and is an adjunct professor in the Master of Science in Information Assurance program at Norwich University.