Towards a Theory of Cyber Attack Mechanics

Peter R. Stephenson*
Center for Regional and National Security,
Eastern Michigan University

Paul S. Prueitt**
George Washington University
Ontologystream, Inc.

*peter.stephenson@emich.edu
**paul@ontologystream.com

Abstract. The concise description of the mechanisms of cyber attacks is a key element in planning and executing defenses against such attacks. To date, analysts have focused upon attacks against information systems based upon data gathered from the attack itself as well as the vulnerability being exploited. This has resulted in an immense number of potential attacks being described based upon an equally immense number of potential vulnerabilities. Additionally, the nature of post-attack analysis is such that the attack must have occurred before it can be analyzed, placing analysts in the position of being reactive rather than proactive.

Typically, defenses against cyber attacks are constructed based upon a combination of the known vulnerabilities within the enterprise and known exploits against those vulnerabilities. It is not uncommon for an organization to find its enterprise vulnerable to certain types of attacks simply because there has been no such attack previously, leading management to dismiss the importance of taking protective action, even though the vulnerability was known to exist.

The notion of defense against cyber attacks need not be based upon historical anecdotal accounts or analyses of pre-existing attacks. Rather, we may view such a defense as having no temporal antecedent. Attacks either are or they are not. If they can exist at all, they will behave in a predictable manner on a very large network such as the Internet. It is this behavior, regardless of the specific attack, and whether the attack is old and well-known or new and unknown, that allows preparation for and minimization of the impact of a cyber attack.

There are significant forensic implications of this notion because an understanding of attack behavior in general may lead to a forensically sound analysis of the origin of the attack. Forensics involves the measurement of what has happened. In this sense measurement of abstract aspects of attack mechanism leads to practical consequences arising from current digital forensic practice.

In this paper we describe the first stage of research into a novel approach to the understanding and application of cyber attack mechanics. First, we define exactly what we mean by a cyber attack. Then we view the impact of the attack from the perspective of the victim, not the attacker. Finally, we abstract attack types into a set of taxonomies and ontologies that allows a precise definition of all aspects of an attack type rather than the individual attack itself.

This offers the advantage of being able to describe, based upon attack mechanics, an approach to protecting the enterprise against cyber attacks, internal or external, whether the actual attack is known or not. We conjecture that it is possible, theoretically, at least, to have total protection against a threat of any kind, known or unknown, as well as against vulnerabilities of any kind, also known or unknown.

Topics: cyber attacks, information systems threats, information systems vulnerabilities, network forensics, cyber attack taxonomy, network trace-back

1 Introduction

Prueitt [1] and others state that any organism is subject to both an exophysics and an endophysics. The exophysics describes the organism's interaction with external elements while the endophysics describes the organism's internal elements. When the organism is a large computing system or network, the exophysics may be thought of as the human interactions with the system while the endophysics is considered to be the internal actions of the computing systems themselves. We can apply this theory to the relationship of cyber attacks and their target computing systems.

Further, we can apply the notion of taxonomies to this theory in that a computing system, such as an enterprise network, can have an endo-taxonomy and an exo-taxonomy that describe and catalog the internal and external relationships relative to the system in question. Starting from this proposition, this research aims to describe a Theory of Cyber Attack Mechanics with the following characteristics:

- An attack is the imposition of a threat upon a vulnerability by a threat agent.
- A cyber incident occurs when an attack is successful
- Cyber Attack space may be described using endo-taxonomies and exo-taxonomies
- Exo-taxonomies are, by their nature, ambiguous while endo-taxonomies tend to be ordered and may be described using simple, structured ontologies

1.1 Definitions

As a foundation, we provide the following definitions [2 updated]:

Definition 1 – Security Policy Domain.

A security policy domain, E_p, consists of all of the elements, $\boldsymbol{e}$, of an enterprise that are subject to the same security policy, $\boldsymbol{p}$.

$$E_p = \{\boldsymbol{e} \in E \mid \boldsymbol{e} \text{ conforms to a policy } \boldsymbol{p}\}$$

Definition 2 – External Stimulus.

An external stimulus applies to a set of states and yields a set of states.

$$\beta \cdot \{ \boldsymbol{s}_i \mid \boldsymbol{i} \in \{1,2,3, \ldots , m\} \} \Rightarrow \{ \boldsymbol{s}_k \mid k \in \{1,2,3, \ldots , n\} \}$$

By convention we will say that where β is applied to a set of states where the size of the set is 1, we say $\beta \cdot \boldsymbol{s}$ rather than $\beta \cdot \{ \boldsymbol{s} \}$.

Definition 3 – Computer Security Incident.

A computer security incident, $\boldsymbol{i}$, results when a change of state of an element, $\boldsymbol{e}$, conforming to a policy $\boldsymbol{p}$ causes that element no longer to conform to that policy, and where the state change is caused by the application of a stimulus, β, external to the system. Note that E_p may contain more than one element.

Consider the set of element states, $\{s_e \mid \boldsymbol{e} \in E_p\}$. If there exists a member $\boldsymbol{s}$ of this set, $\boldsymbol{e} \in E_p$, and an external stimulus β such that $\beta \cdot \boldsymbol{s}$ leads to $\boldsymbol{e} \notin E_p$ then we say that $\beta \cdot \boldsymbol{s} \Rightarrow \boldsymbol{i}$

Definition 4 – Impact

An impact μ results when an external stimulus β is applied to a state $\boldsymbol{s}$.

$$\beta \cdot \boldsymbol{s} \Rightarrow \mu$$

Definition 5 – Vulnerability

A vulnerability, $\boldsymbol{v}$, is a weakness or flaw, in an element of a system, that has the potential to be exploited with a damaging outcome, μ. Let the set, $\{ \boldsymbol{s}_{flawed} \}$, be an enumerated set of states for an element, $\boldsymbol{e}$, of a system.

$\beta \cdot \boldsymbol{e} \Rightarrow \mu$ iff $\{ \boldsymbol{s}_{\text{flawed}} \} \neq \{ \}$

If $\{ \boldsymbol{s}_{\text{flawed}} \} \neq \{ \}$ then $\{ \boldsymbol{s}_{\text{flawed}} \} = \boldsymbol{v}$

An external stimulus, β, can be, but need not be, a natural disaster, an unintentional act by an individual that causes harm or a malicious act by an individual or group of individuals

Definition 6– Threat

A threat, τ, is an external stimulus β that may lead to an incident when the external stimulus β is applied to an element, $\boldsymbol{e}$.

τ is defined when $\beta \cdot \boldsymbol{e} \Rightarrow \boldsymbol{i}$

in which case $\tau \cdot \boldsymbol{e} \Rightarrow \boldsymbol{i}$

Definition 7 - Information Systems Risk,

Information Systems Risk is the probability, **P**, that a threat agent will successfully exploit a vulnerability to create an impact.

$\rho = \mathbf{P}(\tau \cdot \boldsymbol{v} \Rightarrow \mu)$

Definition 8 - Cyber Attack

A cyber attack is an ordered threat-vulnerability pair:

$\boldsymbol{a} = (\tau, \boldsymbol{v})$

2 Problem Statement and Objectives

Currently, digital attacks, or, in the vernacular, "cyber attacks" are categorized in a somewhat haphazard manner. Howard, in his PhD dissertation, approaches the problem by developing a taxonomy of attacks:

> "…from the criticisms of the current taxonomies, and from a *process* or *operational* viewpoint. From this viewpoint, an *attacker* on computers or networks attempts to link to ultimate *objectives* or motivations. This link is established through an operational sequence of *tools*, *access*, and *results* that connects these attackers to their objectives…" [3].

Howard analyzes several different attack taxonomies in his dissertation. The Howard Taxonomy creates five top level families: Attackers, Tools, Access, Results and Objectives. This approach is typical of the development of attack taxonomies and is, at best, subjective based upon the developer's focus. Moreover, the viewpoints all have, up to now, been within the exophysics of the computer-human system. Virtually all well-known attack taxonomies address attacks from the viewpoint of the exophysics. Thus, an attack is viewed as the product of some exploit or set of exploits against a vulnerability or set of vulnerabilities.

An additional problem pointed out by Howard is that many attack taxonomies are, actually, more focused upon individual vulnerabilities. That limitation persists today and, in addition, there are evolving taxonomies of threats [4]. These existing taxonomies, focused upon vulnerabilities and threats as separate topics, are useful but, alone, they do not offer an acceptable attack taxonomy.

Finally, existing taxonomies tend to focus upon the known. While they often contain families broadly enough defined to accommodate future specific attacks, they have not derived those families from a generalized set of formal statements about the nature of attacks.

2.1 Objectives

The objectives of this stage of the research are:

- To identify a generalized hypothesis of the mechanics of cyber attacks. This is distinctly different from understanding cyber attacks individually.
- To describe descriptive cyber attack taxonomy that assists in the understanding of the generalized properties of the mechanics of cyber attacks.
- To prepare for the two additional stages of the research: (1) application of cyber attack taxonomy and formal proof of the generalized hypothesis, and, (2) development of a theoretical artificial immune system that can defend against cyber attacks.

We point out that, at this stage of the research, the taxonomy and ontologies developed are proof-of-concept. In the next stage of the research, application, we expect to refine the threat and vulnerability classes that will result in the working taxonomy for the remaining stages.

2.2 Solution Approach

As can be seen from Definition 8, attacks are the results of threats applied against vulnerabilities. Thus, we may view the attack process holistically, that is, from both the perspective of the exophysics (i.e., what are the interactions – threats – external to the network organism?) and that of the endophysics (i.e., what are the internal mechanisms – vulnerabilities – of the network organism subject to impact caused by a threat?).

By viewing the attack problem in this manner we can develop a more representative taxonomy that is universal in nature and exhibits the characteristics of an acceptable taxonomy [5]:

1) mutually exclusive - classifying in one category excludes all others because categories do not overlap,
2) exhaustive - taken together, the categories include all possibilities,
3) unambiguous - clear and precise so that classification is not uncertain, regardless of who is classifying,
4) repeatable - repeated applications result in the same classification, regardless of who is classifying,
5) accepted - logical and intuitive so that they could become generally approved,
6) useful - can be used to gain insight into the field of inquiry.

2.2.1 Underlying Analysis Techniques

Three primary techniques have been used to develop the attack taxonomy approach in this research. These techniques include:

- Categorical abstraction
- Structural ontology
- Stratified complexity
- Shallow link analysis, Iterated Scatter-Gather, and Parcelation (SLIP)

We address each of these briefly.

2.2.1.1 Categorical Abstraction and Stratified Complexity

Broadly, we may say that each of the elements of machine taxonomy is a categorical abstraction (cA). The set of cA can be represented as symbols or as descriptive phrases that remind about the element. The taxonomy serves as an aid in community communications as well as, if formalized to a sufficient degree, elements that can have role in algorithmic processes. Taxonomies exist within a larger, and often not explicit, model of physical or informational processes. These physical processes enfold information into patterns where the pattern disguises the full nature of informational bits. The cA forms from the induction, or abstraction, of a reduced set of descriptors so that differences between elements in the same category are diminished.

In the case of cyber attacks, we use methods for producing cA about the universe of threats applied against vulnerabilities. We are interested in the functional properties of combinations of elementary patterns, cAs, and the

formation of a higher level of abstraction where function may be realized from any of a large set of combinations of lower level cAs. We call the lower level the level for atomic cAs, and the higher level the molecular cAs.

Using formal methods we break down the results of all possible combinations in the universe of possible attacks to a manageable set of representations. Several inductions are involved in the production of abstractions that leave out details that are not salient in context. The set of molecular cAs contains compounds whose event Chemistry (eC) can become known by applying a special set of formal, abductive, mechanisms similar to Q-SAR (qualitative structure activity relationship) analysis. A stratification of abstractions about process activity allows us to describe a very complicated universe of events, both known and unknown, in a manner that permits us to, we hope, anticipate the unknown from the known.

For example, we know that it is possible to create attacks against buffer overflow vulnerabilities. We do not, nor can we ever, know all the possible permutations of such vulnerabilities or, given an equally large number of possible specific threats, all of the possible attack permutations. However, if we view buffer overflow attacks (including the universes of possible vulnerabilities and threats) at a level of abstraction that addresses the characteristics of an overflow condition, we find that we can talk comfortably about the notion of such attacks. Two benefits accrue, the first is an aid to communication, and the second is as a computational device that allows triggered alerts, general computational reasoning, and case based reasoning.

Extending this approach, a set of early warnings or countermeasures may be developed to address buffer overflow attacks. If our work is done properly, the manageable level of abstraction will address buffer overflow attacks at all levels of abstraction. We would then have induced a solution set for the problem of buffer overflow attacks without needing to know every possible vulnerability, threat and subsequent attack in the buffer overflow universe. We may come to see the buffer-overflow attack as a single “basin of attraction” within an implicit dynamic model of the functions of computers.

Here we see an example of the usefulness of notational stratification wherein the threats and vulnerabilities form from those cA elements that are easily measured as a substructure. The set of sub-structural atoms have natural ways in which the atoms can be aggregated into molecules that fit basins of attraction in the implicit dynamic model. These molecules serve abstract functions needed by, or afforded by, the sets of threats and vulnerabilities that are jointly shaping the model.

One can abstract appropriately to see the beginnings of the expression of attack types. The taxonomy is then anticipatory in nature. This ontology allows us to create a structured taxonomy of cyber attacks derived from the substructure of threats and vulnerabilities.

2.2.1.2 Structural Ontology

The notion of structural ontology often assumes some machine-assisted ontological process. While not addressing that aspect directly, we grant that such a machine-based measurement process is feasible in this case. Our concern in this research has been to develop a set of defining specifications that will result in measurement that can be compared with elements of taxonomy of cyber attacks. The benefit of such an approach is that it allows a precise set of definitions of attack types from which we can abstract, later in this research, a clear notion of a network organism’s self and not-self required in the construction of an artificial immune system.

The fundamental difficulty in developing such a taxonomy is that we must proceed from the known to the unknown. We face, then, a paradox: we wish to proceed from first principles that do not depend upon knowledge of explicit threats and vulnerabilities, but we must depend in part upon known threats and vulnerabilities as starting points. We address this paradox by abstracting away from the specific threats and vulnerabilities and building a structural ontology based upon that level of abstraction. From the ontology we are then able to derive a taxonomy that is abstracted from explicit threats and vulnerabilities and no longer depends upon them as part of a process of attack definitions.

This is an important distinction: the notion of attack types (and the substructural threats and vulnerabilities) must no longer depend upon known attacks. This distinction is important because not only can we not know all possible threat-vulnerability combinations, we cannot even know with certainty what a computing environment, with a completely new set of attack possibilities, might look like in the future. Thus, there is a clear need for some level of

generalization. However, generalization in itself is weak and something deeper is required. The appropriate depth may be achieved using the techniques described here.

2.2.1.3 Shallow link analysis, Iterated Scatter-Gather, and Parcelation (SLIP)

SLIP is a technique that we use to derive a set of elemental representations of attacks based upon threats and vulnerabilities as described in definitions 3, 4 and 7. From Lemma 1 we can see the threat-vulnerability-attack relationship. It is this relationship that we seek to identify using SLIP. SLIP and its use are described in detail by Prueitt [6].

Simply, the SLIP browser tool allows us to explore a universe of attacks expressed as in Definition 8:

$$\left\{\left\langle \tau_i, v, \tau_j \right\rangle\right\}$$

where τ_i and τ_j represent threats against vulnerability v.

This is the specific application of an ontology referential base (Orb) of the form:

$$\left\{\left\langle a_i, r, a_j \right\rangle\right\}$$

where a_i and a_j are substructural elements and *r* is some relation between them (Prueitt, unpublished). Graphically, we can represent this as two nodes connected through a relational operator of some sort [7]. Using the SLIP browser tool, we can apply a collection of threats and a collection of vulnerabilities in the relation above and observe an ordering of those substructural elements into a set of possible attack types organized into an Orb set.

We set the relation as threat type pairs against vulnerability types. This describes the notion of threats against vulnerabilities as being the core concept of an attack. However, referring to Definition 8, we see that there is no limitation forcing a finite number of elements in the Orb set of all possible attacks. In fact, we can conjecture, based upon anecdotal evidence derived from an understanding of current cyber attacks and attack types, that an attack can be far more complicated than might be represented by a single Orb. These complicated attacks may be represented as Orbs arranged as an attack tree.

Examination of combinations of threats and vulnerabilities using SLIP clearly shows that there are a large number of potential combinations of threats that can be applied to a given vulnerability. Thus, we extend the fundamental Orb into an n-ary where we have the possibility of multiple threats against a particular vulnerability, and this n-ary can define a specific attack or attack type depending upon level of abstraction. These n-aries take the general form:

$\left\langle r, a_1, a_2 \ldots a_{(n)} \right\rangle$ or, in the case of an attack, $\left\langle v, \tau_1, \tau_2 \ldots \tau_{(n)} \right\rangle$.

We show examples in Figures 1 and 2. Figure 1 shows a simple Orb where two specific threats applied to a vulnerability result in an attack space. An attack space is the set, as defined in Definition 8, of all attacks such that the elements of the attack space result from a categorical Abstraction of the substructural elements of the Cyber Attack Taxonomy described in Section 3. The Client Side vulnerability family may be exploited by the Malicious Code threat family applying the threat of Flawed Developer Code family.

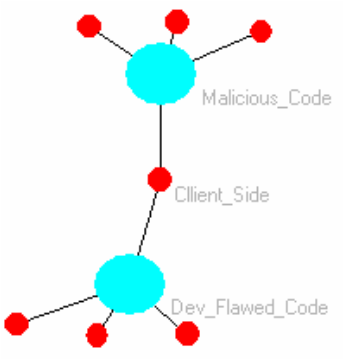

Figure 1 - Application of two threats against a vulnerability resulting in an attack space

We can see easily in this figure that the Malicious_Code and Dev_Flawed_Code threats may be applied against other vulnerabilities. In this example, the attack is characterized as the Orb:

$$\langle \text{Malicious\_Code, Client\_Side, Dev\_Flawed\_Code} \rangle$$

Figure 2 shows a much more complicated attack space.

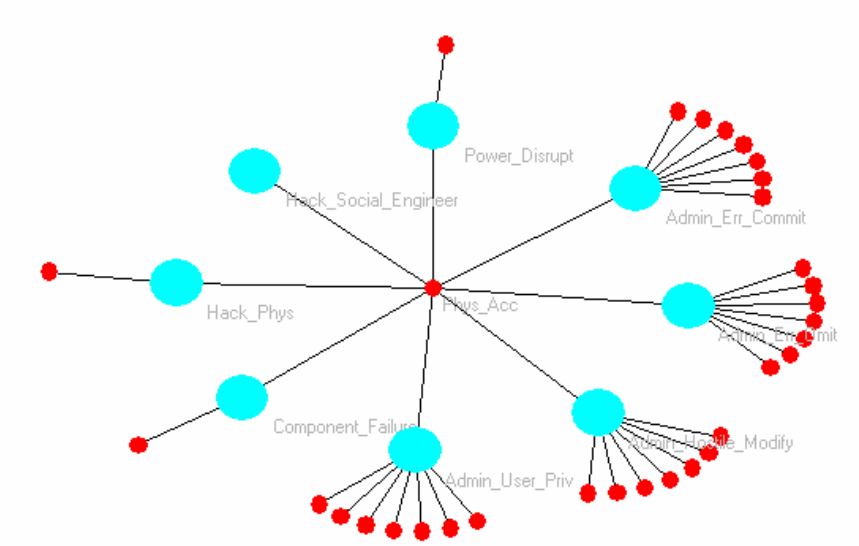

Figure 2 - Application of several threats against a vulnerability resulting in an attack space

In this figure we see that eight threat families can work together or in groups to form an attack space with the physical access (Phys_Acc) vulnerability family giving the n-ary:

<Phys_Acc, Power_Disrupt, Admin_Err_Commit, Admin_Err_Omit, Admin_Hostile_Modify, Admin_User_Priv, Component_Failure, Hack_Phys, Hack_social_Engineer>

Again, we can see that each of the threats indicated could be applied against other vulnerabilities resulting in other Orbs or n-aries. These maps are the result of applying SLIP to a conjecture consisting of cyber threat groups that could, potentially, be applied against information systems vulnerability groups. In examining the n-aries we may find that we can construct attack Orbs. Abstracting these Orbs down to specific attacks reveals the potential for an unlimited number of possible attacks given an unlimited combination of individual threats and vulnerabilities. Not

all Orb combinations taken from n-aries make sense given the currently available threats and vulnerabilities, of course. However, by thinking of them, at a higher level of abstraction, as "place holders" we account for future specific threat-vulnerability combinations and the stratification becomes anticipatory.

We also note that we are representing the threat and vulnerability family level of abstraction. In Figure 1 it makes no difference what the nature of the flawed developer code, the malicious code or the client side error may be. It is sufficient to say that malicious code in combination with a piece of flawed developer code can result in a client side error condition on the target application or operating environment. Function is fulfilled by any of a number of structures so that function-structure relationship is something to be discovered, and once discovered is useful.

It also says nothing about the intent of the person writing the malicious code or the person who made the error in the developer code. This is an example of an ambiguous exo-physics (the malicious code) impacting the unambiguous endophysics of the application or operating environment (the client side environment). This is an important concept because the attack is intended to result in an explicit change of state in the finite state machine of the target application or operating environment (Definition 2). The threat experiences no such state change and is, therefore, ambiguous or, perhaps, chaotic. The nature of the flawed developer code is, likewise, ambiguous since there is no finite state change that results in or from an error in coding of the software.

3 Cyber Attack Taxonomy

We begin the development of a taxonomy of cyber attacks by observing current threat and vulnerability patterns. However, if we are to account for future unknown threats, vulnerabilities and computing environments, we must be careful to select appropriate levels of abstraction. This requires that we develop an ontology that defines an environment that supports stratification of threats and vulnerabilities. Figure 3 shows the structure of the attack taxonomy.

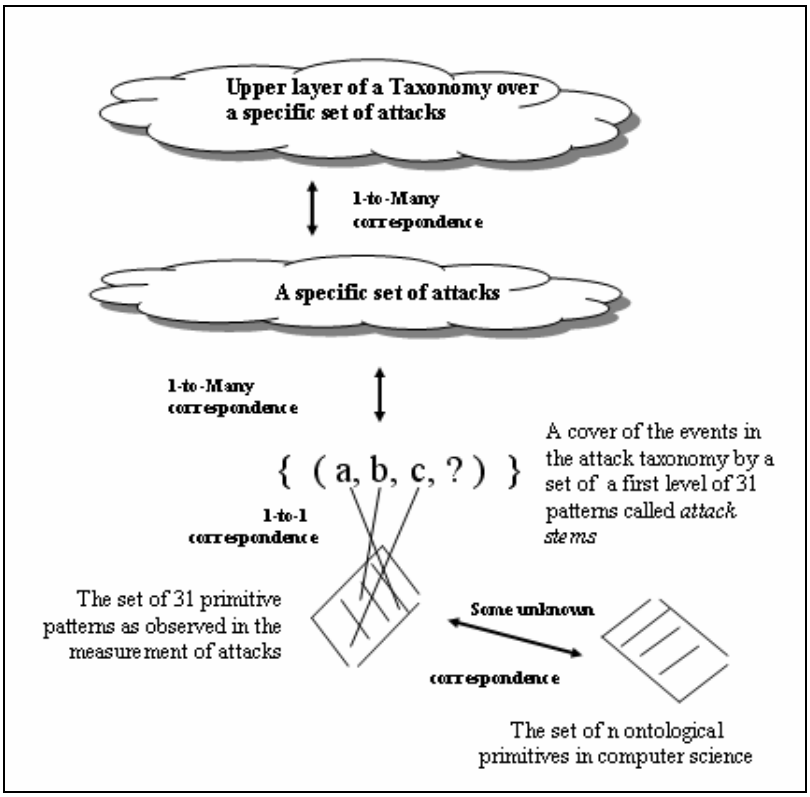

Figure 3 - Structure of the Attack Taxonomy

The 31 primitives, or primes, are derived from a set of observations of threats and vulnerabilities. The threat families are taken from the Profiling Knowledge Base of the Common Criteria (ISO 15408). These threat families are the nuclei (the large circles) of the atoms produced by the SLIP Browser as shown in figures 1 and 2. The

valences (small dots) on the atoms represent the vulnerability families against which the threat family would be credible. The vulnerability families are abstracted from a vulnerability taxonomy developed as part of the Common Vulnerabilities and Exposures list maintained by Mitre Corp.[1] The 31 primes comprise a set of attack stems against which all existing attacks may be described. The description is in the context of a descriptive cover, and may not be universal. However, our future work will address the descriptive aspects of multiple covering ontologies. A type of universality may be obtained as a limiting process.

The current upper taxonomy comprises nine compounds created by the SLIP browser. These compounds represent fundamental attack families at a level of abstraction that permits useful anticipatory analysis of attack types. Over the course of the next two stages of the research, we expect to refine this taxonomy. The current 31 primes and the upper taxonomy are shown in Appendix B.

4 A Hypothesis of Cyber Attack Mechanics

Any sufficiently large network such as the public Internet may be described in terms of the fractal nature of its connection topology, particularly its routers [9], [10]. Both cited groups of researchers concluded that the pattern of routers on the Internet was fractal relative to population centers and that the likelihood of connecting to node increases with the number of links available on the node.

This implies that the traffic between nodes is, as well, fractal. With an increase in nodes the traffic increases in predicable increments. Yook, et al [9] establish a fixed fractal dimension $_f D$ for a scale-invariant fractal set that describes both Internet routers and domains. Cyber attacks, however, may be seen to introduce ambiguities in the traffic flow. We hypothesize that the imposition of a cyber attack upon a large, complicated network disturbs the fractal nature of that network's data flows and causes perturbations that may alter the dimension of the data flows. The larger the attack, the greater become ambiguities. Large attacks, such as large-scale worm infections, produce easily noticeable ambiguities. Smaller attacks produce far more subtle ones. The instrumentation for the measurement of these ambiguities presents a challenge in a practical sense.

We begin at this stage of the research with the following hypothesis.

Hypothesis of Cyber Attack Mechanics.

> The interaction between cyber attack space and fractal network space results in attack markers that anticipate the existence of a cyber attack. Because attack markers are complex, they may result in halting conditions within the network. These halting conditions can be represented formally and the source of the cyber attack may be deduced.

There is anecdotal evidence in support of this hypothesis from a variety of sources. For example, Nicol, et al [8] describe the behavior of the Code Red version 2 Internet worm as observed through BGP routers on the Internet. The hypothesis addresses very large unbounded networks such as the Internet that exhibit fractal properties either in router placement or centers of user population (domains).

We may represent this hypothesis formally as:

$$a \cdot \{ e_f \Rightarrow \triangle D_{f(e)} \} \Rightarrow \xi$$

where:

(a) ξ is an event marker

(b) $\boldsymbol{e}_f$ is an element of a scale-invariant fractal set describing Internet routers or domains

(c) $\triangle D_{f(\boldsymbol{e})}$ is the change in the fractal dimension of the scale-invariant fractal set containing element $\boldsymbol{e}$

[1] http://cve.mitre.org

We read this as: An event marker results from an attack against an element of a scale-invariant fractal set describing Internet routers or domains such that there is a change in the fractal dimension of that element.

Expanding (Definition 8):

$$(\tau, \nu) \cdot \{ \boldsymbol{e}_{\mathrm{f}} \mid \triangle D_{\mathrm{f}\,(\boldsymbol{e})} \} \Rightarrow \xi$$

We see that an event marker results from an ordered threat-vulnerability pair applied to an element of a scale-invariant fractal set describing Internet routers or domains such that there exists a change in the fractal dimension of the system. The change in measurement triggers mechanisms that attentively select for information about the origin of the cause of the measured change in dimension.

Event markers may be placed as tokens in formal models of the set of scale-invariant fractal sets representing the attack space using a formalism such as Colored Petri Nets. From the simulation of these models the probable source of the attack may be deduced. Identification and measurement of the event marker from a practical perspective present a challenge for the next stages of the research. The event marker represents a categorical abstraction that is invariant with the property that it is not understood in the context of nominal operation of the network.

The notion of a well-formed enterprise is significantly different from that of a large network such as the Internet. We may view the enterprise as being a well-formed system where the states of the elements of the network behave in a manner consistent with that intended by its developers. Variations in state behavior are noticeable, can be measured, and may be attributed to some cause external to the enterprise's nominal operation.

In a network such as the Internet, however, the objectives of the system are to provide connectivity for the enterprises attached to it. The focus, then, is upon the attachment points and the links, or paths, between them. Changes in the states of the elements of the Internet are extremely numerous and frequent, and are unmanaged by any central authority. In the current Internet, variations in state changes then are neither easily noticeable nor easily attributable. Thus, we see that the behavior, and, consequently, its measurement, of an enterprise differ markedly from that of a very large-scale network such as the Internet.

The development of measured behavioral analysis over Enterprises is a first step towards the longer range goal of measurement over the complete Internet.

A corollary to the Hypothesis applies explicitly to smaller networks such as organizational enterprises. There is anecdotal evidence that this corollary is correct [11].

Corollary 1 to the Hypothesis.

> An enterprise network may be represented as a finite state machine. In an enterprise network an attack may be characterized as being an event contrary to the desired operating state of the network. Attack markers within bounded networks are the result of a change in state of one or more nodes of the finite state machine and may lead to a halting condition of those nodes. These halting conditions can be represented formally and the source of the cyber attack may be deduced.

The meaning here is that an event marker results from an incident as defined in Definition 3. Again, event markers may be placed as tokens in formal models of the bounded network using a formalism such as Colored Petri Nets and halting conditions (deadlocks) identified.

We explicitly define the “correct” operating state of the system as its “as designed” state. There is no assumption that the pre-attack state is, in fact, the correct operating state. However, the change of state to an incorrect state (perhaps a *different* incorrect state) must be as the result of the external stimulus. If the pre-attack state was not correct (i.e., not as designed), as long as the state change does not result in a return to a correct operating state, the corollary holds.

This notion accounts for errors occurring in the network that are not the result of a malicious attack. These errors, as demonstrated in Definition 2, can result in a security incident. Experientially we have seen many security incidents that were not the result of malicious intent. In fact, the understanding of an attack or incident has nothing whatever to do with the intent behind it. Referring to Definition 4 we see that a threat is *"…natural disaster, an unintentional act by an individual that causes harm or a malicious act by an individual or group of individuals"*. Thus the point of this corollary is to describe in pure terms generation of an attack marker without regard to the intent behind its generation.

While we see no theoretical reason at this stage of the research why this corollary could not be applied on larger networks such as the Internet, from a practical perspective it would be nearly impossible to measure the state of every element possibly involved in an attack. Additionally, we know from observation that a large-scale attack may involve many interacting elements (such as zombies in a distributed denial of service attack) complicating the process of assessing state changes in individual network elements. Therefore, we hypothesize that the appropriate level of abstraction for very large networks is the fractal dimension of the data flows on that network.

5 Conclusions and Future Work

At this stage of the research we conclude that it is feasible to characterize cyber attacks and their contributing elements formally. We have provided a core set of formally sound definitions, a notation a cyber attack taxonomy, and a set of formal tools from which a theory of cyber attack mechanics may be developed and tested based upon our hypothesis. We further conclude that this characterization has forensic value in that it may enable forensic investigations of anomalistic events on both large-scale networks such as the Internet and on smaller enterprise networks where networks states can be known.

5.1 Second Stage: Application of the cyber attack taxonomy and formal proof of the foundational hypothesis developed in this stage of the research

During this stage we will refine the cyber attack taxonomy through testing of the assumptions used in the current proof of concept. We will use the taxonomy and the tools reported here to refine and develop a formal proof of the hypothesis into a theory of cyber attack mechanics. An important part of this stage's work involves validation that data flows within a very large network such as the Internet exhibit some consistent fractal characteristics as does the topology, as well as demonstrating the quantification of those characteristics.

Additionally, we will identify the specific characteristics of the event marker and develop a method to instrument a network to identify and measure it. During this stage we expect to be able to demonstrate a forensic method for identifying the probable path of an attack over a large network such as the Internet using formal modeling and induction.

5.2 Third Stage: Development of a theoretical artificial immune system that can defend against cyber attacks

During this stage, we apply the theory of cyber attack mechanics and the concept of the event marker to the theoretical specification of an artificial immune system that can defend an enterprise network against attack independently of any specific device on the enterprise. Such an artificial immune system cannot have any dependencies upon specific knowledge of attack patterns (such as anti-virus pattern files) and must defend through knowledge of the enterprise self and not-self. We hypothesize that the event markers represent pathogens that can be recognized by the enterprise organism and resisted.

References

1. Prueitt, Paul S. Foundations of Knowledge Management in the 21st Century, Chapter 1, Published on the Web at < http://www.bcngroup.org/area3/pprueitt/kmbook/Chapter1.htm> 15 Sept, 1999 and revised slightly 19 March 2004. Accessed 13 November 2004.
2. Stephenson, Peter. "A Formal Model for Information Risk Analysis Using Colored Petri Nets", CPN 2004 5th Annual Workshop on Colored Petri Nets - Proceedings, 8 – 11 October 2004, Aarhus, Denmark
3. Howard, John D. An Analysis Of Security Incidents On The Internet 1989 – 1995, PhD Dissertation, 7 April 1997
4. Jones, Andrew. "Identification of a Method for the Calculation of Threat in an information Environment" published internally, QinetiQ, Inc. April 2002 and updated 2004.
5. Amoroso, Edward G. Fundamentals of Computer Security Technology, Prentice Hall PTR, Upper Saddle River, NJ 1994
6. Prueitt, Paul S. "SenseMaking Experiment with SLIP Analytic Conjectures", Published on the Web at <http://www.ontologystream.com/SLIP/files/SLIP-AC.htm>, 6 November 2001. Accessed 25 November 2004.
7. Prueitt, Paul S. "Community Building Workshop supporting Anticipatory Human-centric Information Production (HIP) Challenge Problem" Published on the Web at <http://www.ontologystream.com/beads/nationalDebate/115.htm>, 14 September 2004. Accessed 25 November 2004.
8. Liljenstam , Michael, Yougu Yuan, BJ Premore, David Nicol. "A Mixed Abstraction Level Simulation Model of Large-Scale Internet Worm Infestations" *Proceedings of the Tenth IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer Telecommunication Systems* Fort Worth, TX, October, 2002
9. Yook , Soon-Hyung, Hawoong Jeong, Albert-Laszlo´ Barabasi. "Modeling the Internet's Large-Scale Topology" Proceeding of the national Academy of Science , 13382–13386 October 15, 2002, vol. 99 no. 21
10. Lakhina, Anukool, John W. Byers, Mark Crovella, Ibrahim Matta. "On the Geographic Location of Internet Resources" Proc. of ACM SIGCOMM Internet Measurement Workshop, 2002
11. Stephenson, Peter. "Modelling of Post Incident Root Cause Analysis", International Journal of Digital Evidence Fall 2003 Volume 2 Issue 2

APPENDIX A – Notation

In addition to standard logic and set notation, we use the following:

i) E is a set of elements that define a system

ii) $\boldsymbol{e}$ is a specific element where $\boldsymbol{e} \in E$

iii) S is a system where $\{\boldsymbol{s}_1 \ldots \boldsymbol{s}_n\}$ is an enumerated set of states of a selected number of elements of the system

iv) P is the set of all security policies on an enterprise network

v) $\boldsymbol{p}$ is a particular policy where $\boldsymbol{p} \in P$

vi) D is a security policy domain

vii) B is the set of external stimuli applied to S

viii) β is an external stimulus where $\beta \in B$

ix) I is the set of all possible computer security incidents

x) $\boldsymbol{i}$ is a security incident where $\boldsymbol{i} \in I$

xi) V is the set of all possible vulnerabilities in a system

xii) $\boldsymbol{v}$ is a specific vulnerability where $\boldsymbol{v} \in V$

xiii) T is the set of all threats, including natural disasters, malicious and unintentional acts by individuals, that can cause harm

xiv) τ is a specific threat where $\tau \in T$

xv) M is the et of all possible impacts where an impact represents a change of state in some set of elements $\{\boldsymbol{e}_1 \ldots \boldsymbol{e}_n\}$

xvi) μ is a specific impact where $\mu \in M$

xvii) R is the set of all information systems risks

xviii) ρ is a specific information systems risk where $\rho \in R$

xix) A is the set of all attacks, successful or not where $A = \{< \tau_i, \boldsymbol{v}_n, \tau_j > \ldots < \tau_{()}, \boldsymbol{v}_{()}, \tau_{()} >\}$

xx) $\boldsymbol{a}$ is a particular attack where $\boldsymbol{a} \in A$

xxi) Ψ is the set of all successful attacks where $\Psi \subseteq A$

xxii) ψ is a particular successful attack where $\psi \in \Psi$

xxiii) $\Rightarrow$ implies, yields or leads to, as in $a \Rightarrow \psi$

xxiv) iff if and only if

xxv) $\equiv$ is the same as

xxvi) $\cdot$ applied to, as in $\beta \cdot \boldsymbol{e}$

xxvii) ξ is an event marker used to identify a cyber event

APPENDIX B – Cyber Attack Taxonomy

B.1 The 31 Cyber Attack Primes

1. User_Send
   - Access_Err
   - Conf_Err
   - Config_Err
   - Input_Err
   - Logic_Err
2. User_Modify
   - Access_Err
   - Conf_Err
   - Config_Err
   - Input_Err
   - Logic_Err
3. User_Misuse_Avl_Resc
   - Input_Err
   - Logic_Err
4. User_Err_Slf_Protect
   - Config_Err
   - DOS
   - Input_Err
   - Logic_Err
5. User_Err_Misuse_Avl_Resc
   - Config_Err
   - DOS
6. User_Err_Integrity
   - Config_Err
   - Input_Err
   - Logic_Err
7. User_Err_Inaccess
   - Config_Err
   - Input_Err
   - Logic_Err
8. User_Err_Conf
   - Access_Err
   - Config_Err
   - Input_Err
   - Logic_Err
9. User_Collect
   - Access_Err
   - Audit_Err
   - Conf_Err
   - Config_Err
   - Logic_Err
10. User_Abuse_Conf
    - Access_Err
    - Conf_Err
    - Config_Err
    - Logic_Err
11. Spoofing
    - Access_Err
    - Audit_Err
12. Repudiate_Transact
    - Audit_Err
13. Repudiate_Send
    - Audit_Err
14. Repudiate_Receive
    - Audit_Err
15. Power_Disrupt
    - DOS
    - Phys_Acc
16. Malicious_Code
    - Client_Side
    - Config_Err
    - DOS
    - Logic_Err
17. Hack_Social_Engineer
    - Phys_Acc
18. Hack_Phys
    - DOS
    - Phys_Acc
19. Hack_Msg_Data
    - Access_Err
    - Config_Err
    - Input_Err
    - Logic_Err
20. Hack_Masq
    - Access_Err
    - Input_Err
    - Logic_Err
21. Hack_Crypto
    - Conf_Err
22. Hack_Comm_Eavesdrop
    - Access_Err
    - Conf_Err
    - Logic_Err
23. Hack_Avl_Resc
    - DOS
    - Logic_Err
24. Hack_AC
    - Access_Err
    - Audit_Err
    - Config_Err
    - Input_Err
    - Logic_Err
25. Failure_DS_Comp
    - Config_Err
    - DOS
26. Dev_Flawed_Code
    - Access_Err
    - Audit_Err
    - Client_Side
    - DOS
27. Component_Failure
    - DOS
    - Phys_Acc

28. Admin_User_Priv
    - Access_Err
    - Phys_Acc
    - Audit_Err
    - Conf_Err
    - Config_Err
    - DOS
    - Input_Err
    - Logic_Err
29. Admin_Hostile_Modify
    - Access_Err
    - Phys_Acc
    - Audit_Err
    - Conf_Err
    - Config_Err
    - DOS
    - Input_Err
    - Logic_Err
30. Admin_Err_Omit
    - Access_Err
    - Phys_Acc
    - Audit_Err
    - Conf_Err
    - Config_Err
    - DOS
    - Input_Err
    - Logic_Err
31. Admin_Err_Commit
    - Access_Err
    - Phys_Acc
    - Audit_Err
    - Conf_Err
    - Config_Err
    - DOS
    - Input_Err
    - Logic_Err

B.2 The 9 Upper Taxonomy n-aries

1. <**Access_Err**, User_Send, User_Modify, User_Err_Conf, User_Collect, User_Abuse_Conf, Spoofing, Hack_Msg_Data, Hack_Masq, Hack_Comm_Eavesdrop, Hack_AC, Dev_Flawed_Code, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit, User_Send>

2. <**Conf_Err**, User_Send, User_Modify, User_Collect, User_Abuse_Conf, Hack_Crypto, Hack_Comm_Eavesdrop, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

3. <**Config_Err**, User_Send, User_Modify, User_Err_Slf_Protect, User_Err_Misuse_Avl_Resc, User_Err_Integrity, User_Err_Inaccess, User_Err_Conf, User_Collect, User_Abuse_Conf, Malicious_Code, Hack_Msg_Data, Hack_AC, Failure_DS_Comp, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

4. <**Input_Err**, User_Send, User_Modify, User_Misuse_Avl_Resc, User_Err_Slf_Protect, User_Err_Integrity, User_Err_Inaccess, User_Err_Conf, Hack_Msg_Data, Hack_Masq, Hack_AC, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

5. <**Logic_Err**, User_Send, User_Modify, User_Misuse_Avl_Resc, User_Err_Slf_Protect, User_Err_Integrity, User_Err_Inaccess, User_Err_Conf, User_Collect, User_Abuse_Conf, Malicious_Code, Hack_Msg_Data, Hack_Masq, Hack_Comm_Eavesdrop, Hack_Avl_Resc, Hack_AC, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

6. <**DOS**, User_Err_Slf_Protect, User_Err_Misuse_Avl_Resc, Power_Disrupt, Malicious_Code, Hack_Phys, Hack_Avl_Resc, Failure_DS_Comp, Dev_Flawed_Code, Component_Failure, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

7. <**Audit_Err**, User_Collect, Spoofing, Repudiate_Transact, Repudiate_Send, Repudiate_Receive, Hack_AC, Dev_Flawed_Code, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

8. <**Phys_Acc**, Power_Disrupt, Hack_Social_Engineer, Hack_Phys, Component_Failure, Admin_User_Priv, Admin_Hostile_Modify, Admin_Err_Commit, Admin_Err_Omit>

9. <**Client_Side**, Malicious_Code, Dev_Flawed_Code>