Modeling Computer Attacks: A Target-Centric Ontology for Intrusion Detection

Jeffrey Undercoffer and John Pinkston
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
{junder2, pinkston}@cs.umbc.edu
phone: 410-455-3971
fax: 410-455-3969

Abstract

We have produced an ontology specifying a model of computer attacks. Our ontology is based upon an analysis of over 4,000 classes of computer attacks and their corresponding attack strategies, and is model is categorized according to: system component targeted, means of attack, consequence of attack and location of attacker. Our analysis indicates that non-kernel space applications are most likely to be attacked with the attack originating remotely. These attacks most often result in the attacker gaining root access. We argue that any taxonomic characteristics used to define a computer attack be limited in scope to those features that are observable and measurable at the target of the attack. We present our attack model first as a taxonomy and convert it to a target-centric ontology that will be refined and expanded over time. We state the benefits of forgoing dependence upon taxonomies for the classification of computer attacks and intrusions, in favor of ontologies. We illustrate the benefits of utilizing an ontology by comparing a use case scenario of our ontology and the IETF's Intrusion Detection Exchange Message Format Data Model.

Keywords: Intrusion Detection, Ontologies, Attack Signatures

1 Introduction

Based upon empirical evidence we have produced a model of computer attacks categorized by: the system component targeted, the means and consequence of attack, and the location of the attacker. Our model is represented as a *target-centric* ontology, where the structural properties of the classification scheme is in terms of features that are observable and measurable by the target of the attack or some software system acting on the target's behalf. In turn, this ontology will be used to facilitate the reasoning process of detecting and mitigating computer intrusions.

Traditionally, the characterization and classification of computer attacks and other intrusive behaviors have been limited to simple taxonomies. Taxonomies, however, lack the necessary and essential constructs needed by an intrusion detection system (IDS) to reason over an instance representative of the domain of a computer attack. Unlike taxonomies, ontologies provide powerful constructs that include machine interpretable definitions of the concepts within a domain and the relations between them. Ontologies provide software systems with the ability to share a common understanding of the information at issue in turn enabling the software system with a greater ability to reason over and analyze this information.

Since existing research is limited to taxonomies and because a taxonomy is contained within an ontology we start with taxonomies and build to ontologies.

As detailed by Allen, et. al [1], and McHugh [20], the taxonomic characterization of intrusive behavior has typically been from the attacker's point of view, each suggesting that alternative taxonomies need to be developed. Allen et. al state that intrusion detection is an immature discipline and has yet to establish a commonly accepted framework. McHugh suggests classifying attacks according to protocol layer or, as an alternative, whether or not a completed protocol handshake is required. Likewise, Guha [9] suggests an analysis of each layer of the TCP/IP protocol stack to serve as the foundation for an attack taxonomy.

The Intrusion Detection Working Group of Internet Engineering Task Force (IETF) has proposed the Intrusion Detection Message Exchange Requirements [33] which, in addition to defining the requirements for the Intrusion Detection Message Exchange Format, also specifies the architecture of an intrusion detection system (IDS). The Intrusion Detection Message Exchange

Format Data Model and Extensible Markup Language (XML) Document Type Definition [5] (IDMEF) is a profound effort to establish an industry wide data model which defines computer intrusions. IDMEF has its shortcomings, however. Specifically, it uses XML which is limited to a syntactic representation of the data model. This limitation requires each IDS to interpret and implement the data model programatically. Moreover, XML does not support the notion of inheritance, which means that the data model will not benefit from substitutability – a property allowing a value of a subtype to be used in place of a supertype without prior knowledge of the subtype.

As an alternative to IDMEF, we propose a data model represented by an ontology representation language such as the Resource Description Framework Schema (RDFS) [26]. We illustrate the benefits of using ontologies for IDS's by presenting an example of our ontology being utilized by IDSs supported by *SHOMAR* [29], a framework for distributed intrusion detection services. SHOMAR is an optimization of [28] and [13] architectures that provide secure service discovery and access in heterogeneous network and computing environments.

Generally, IDS's are either adjacent to or co-located with the target of an attack. It is imperative, therefor, that any classification scheme used to represent an attack be *target-centric*, where each taxonomic character is comprised of properties and features that are observable by the target of the attack. Consequently, our taxonomy, and subsequently our ontology, defines properties and attributes in terms of characteristics that are observable and measurable by the target of an attack.

As a basis for establishing our *a posteriori* target-centric attack ontology we evaluated and analyzed over 4,000 computer vulnerabilities and their corresponding attack strategies. Section 2 presents the characteristics of a sufficient taxonomy. Section 3 presents related work in the form of alternative attack taxonomies as well as presenting related work in the area of ontologies for intrusion detection. Section 4 details the empirical data on which our study is based and presents the results of our analysis. Our target-centric attack taxonomy is presented in Section 5. Section 6 details the motivation for abandoning taxonomies in favor of ontologies and Section 6.1 presents our target-centric ontology. Section 6.1.1 provides an example scenario illustrating the utility of the ontology within system of distributed intrusion detection systems. Section 7 describes our implementation and we conclude in Section 8.

2 Characteristics of a Sufficient Taxonomy

At this preliminary stage, a clear understanding of the definition, purpose and objective of a taxonomy is in order. Accordingly, a *taxonomy* is a *classification* system where the classification scheme conforms to a systematic arrangement into groups or categories according to established criteria [31]. Glass and Vessey [7] contend that taxonomies provide a set of unifying constructs so that the area of interest can be **systemically** described and aspects of relevance may be interpreted. The overarching goal of any taxonomy, therefore, is to supply some predictive value during the analysis of an unknown specimen, while the classifications within the taxonomy offer an explanatory value.

According to Simpson [27] classifications may be created either *a priori* or *a posteriori*. An *a priori* classification is created non-empirically whereas an *a posteriori* classification is created by empirical evidence derived from some data set. Simpson defines a taxonomic character as a feature, attribute or characteristic that is divisible into at least two contrasting states and used for constructing classifications. He further states that taxonomic characters should be observable from the object in question.

Amoroso [2], Lindqvist, et. al [18] and Krusl [17] each have identified what they believe to be the requisite properties of a sufficient and acceptable taxonomy for computer security. Collectively, they have identified the following properties as essential to a taxonomy:

Mutually Exclusive A classification in one category excludes all others because categories do not overlap.

Exhaustive The categories, taken together, include all possibilities.

Unambiguous The category is clear and precise so that classification is not uncertain, regardless of who is classifying.

Repeatable Repeated applications result in the same classification, regardless of who is classifying.

Accepted The taxonomy should be logical and intuitive so that it can become generally approved.

Useful The taxonomy can be used to gain insight into the field of inquiry.

Comprehensible The taxonomy should be useful to those with less than expert knowledge.

Conforming The terminology of the taxonomy should comply with established security terminology.

Objectivity The features must be identified from the object under observation where the attribute being measured should be clearly observable.

Determinism There must be a clear procedure that can be followed to extract the feature.

Repeatability Several people independently extracting the same feature for the object must agree on the value observed.

Specificity The value for the feature must be unique and unambiguous.

Upon review of the above list we believe that a sufficient and acceptable taxonomy must be: **Mutually Exclusive, Exhaustive, Unambiguous, Useful, Objective, Deterministic, Repeatable** and **Specific**.

3 Related Work

As previously stated, most of the existing research in the area of the classification of computer attacks is limited to taxonomies. Accordingly, this section is subdivided, with Subsection 3.1 presenting related work in the area of taxonomies for intrusion detection and Subsection 3.2 presenting related work in the area of ontologies for intrusion detection.

3.1 Related Work: Taxonomies

There are numerous attack taxonomies proposed for use in intrusion detection research. Howard [12] provides a *"Complete Computer and Network Attack Taxonomy"*, classifying attacks according to: *Attackers, Tools, Access, Results* and *Objectives*. Within Howard's taxonomy the types of attackers include: *Hackers, Spies, Terrorists* and *Teenagers*, while the objectives include: *Political Gain* and *Financial Gain*. These characteristics, however, are not discernible by analyzing an instance of the intrusive event. Specifically, an IDS does not have the means of discerning whether an attacker is a terrorist or a teenager or if the attacker's objective is financial gain or curiosity.

During the 1998 and 1999 DARPA Off Line Intrusion Detection System Evaluations [10, 19] Weber [15] provided a taxonomy classified by *Initial Privilege Level, Method of Transition to a New Privilege Level and New Privilege Level*. Kendall includes *Social Engineering* [1] in the *Method of Transition* category, however, detecting off-line human interaction is beyond the scope of an IDS, hence that specific taxonomic character is not discernible by objectively observing the attack.

Neumann and Parker [22] categorize computer attacks into nine classes: *External, Hardware, Masquerading, Setting Up Subsequent Misuse, Bypassing Access Controls, Active Misuse of Resources, Passive Misuse of Resources, Misuse Resulting From Inaction and Use as an Indirect Aid in Committing Other Misuse*. According to Neumann and Parker, their taxonomy is not mutually exclusive, hence it too falls outside the bounds of our definition of a sufficient and satisfactory taxonomy.

Lindqvist and Jonsson [18] state that they *"focus on the external observations of attacks and breaches which the system owner can make"* consequently they create a taxonomy in terms of *intrusion techniques* and *intrusion results*. Their categories of intrusion techniques are: *Bypassing Intended Controls, Active Misuse of Resources and Passive Misuse of Resources* and their categories of intrusion results are: *Exposure, Denial of Service and Erroneous Output*. They provide two examples of passive misuse of resources – *"automated searching using a personal tool"* and *"automated searching using a publicly available tool"*. Here too, these taxonomic characters are not discernible by objective observation of the attack because the knowledge of the tool's origin is beyond the scope of the target or an IDS running on behalf of the target.

In their *"Taxonomy of Security Faults"*, which defines a classification scheme for security faults in the Unix operating system, Aslam et. al [3] group vulnerabilities according to *Emergent Faults, Environment Faults, Coding Faults and Other Faults*. They define *Coding Faults* as faults introduced during software development and include errors in programming logic, missing or incorrect requirements and design errors. They conclude by stating that *" a database of vulnerabilities using this classification was implemented and is being used in the production of tools that detect and prevent computer break-ins"*. Although this taxonomy may be quite useful in the security analysis of software, it does not lend itself to an IDS because the IDS cannot discern between intended program behavior and program behavior that is the consequence of incorrect requirements.

In a recent paper, McHugh et. al [21] characterize two alternative attack perspectives; the target's view and the attacker's view. Accordingly, they characterize these views as being focused on the following manifestations:

Victim View

1. What happened?
2. Who is affected and what?
3. Who is the intruder?
4. Where and when did the intrusion originate?
5. How and why did the intrusion happen?

Attacker View

1. What is my objective?
2. What vulnerabilities exist in the target system?
3. What damage or other consequences are likely?
4. What exploit scripts or other attack tools are available?
5. What is my risk of exposure?

As detailed by McHugh, these are two completely different perspectives where the target of the attack does not have the information required to answer the questioned posed in the Attacker's viewpoint. Accordingly, they should not be included in a taxonomic classification used by an IDS.

[1] Social engineering is a term that describes a non-technical kind of intrusion that relies heavily on human interaction and often involves tricking other people to break normal security procedures. A social engineer runs what used to be called a "con game".

3.2 Related Work: Ontologies

There is little, if any, published research formally defining ontologies for use in Intrusion Detection.

Raskin et. al [25] introduce and advocate the use of ontologies for information security. In arguing the case for using ontologies, they state that an ontology organizes and systematizes all of the phenomena (intrusive behavior) at any level of detail, consequently reducing a large diversity of items to a smaller list of properties.

4 Vulnerabilities and Attack Strategies: Empirical Analysis

In gathering data for our study, we relied upon the *CERT/CC Advisories* maintained by the "Computer Emergency Response Team/Coordination Center" of Carnegie Mellon University's Software Engineering Institute and the "*Internet Catalog of Assailable Technologies"* (ICAT) maintained by the National Institute of Standards. Both provide a listing of known computer vulnerabilities and exploits. CERT obtains its data from computer incident reports made by the public at large. CERT, after a forensic examination of the reported incident, and providing the incident has wide spread impact, posts an advisory. ICAT is a compilation of vulnerabilities derived from multiple sources, including but not limited to: CERT, Internet Security Systems (ISS), Bugtraq, Microsoft and Security Focus.

Currently, the ICAT meta-base contains 4,160 entries and is classified according to severity, loss type, vulnerability type, exposed system component, etc. The ICAT classification scheme is not mutually exclusive. Therefore, for our study, we only considered 4,048 entries from the ICAT data set. Furthermore, we reclassified many of the ICAT entries to ensure that each sub-category was mutually exclusive and non-ambiguous. For example, ICAT lists the exposed component of the *Land* [2] attack as both the network protocol stack and the operating system as well as stating that multiple vulnerabilities are responsible for enabling the *Land* attack: "Input Validation Error", "Buffer Overflow", "Boundary Overflow" and an "Exceptional Condition Handling Error". CERT, however, states that *Land* is an attack comprised of a SYN packet in which the source address and port are the same as the destination address and port, resulting in an input validation error.

CERT has issued 286 advisories since its inception in 1985, and we have included all of these in our study. We compared the statistics derived exclusively from the CERT advisories with those derived from ICAT (which includes CERT) for continuity between the two data sets.

[2] The *Land* attack is an IP Denial of Service Attack where a SYN packet in which the source address and port are the same as the destination address and port.

The purpose of our analysis is to identify the means of attack that are most frequently employed (i.e. as manifested at and experienced by the target), the most likely consequence of an attack (i.e. as experienced by the target), the component of the target that is most often targeted by an attack and the most common location from whence the attack originated.

We present our analysis by plotting the means, consequences and location of attack against each of the four identified system components (network, kernel-space, application and other) targeted during an attack.

4.1 Means of Attack

Figure 1 contains an accounting of the means of attack (against the each of the four targeted components)as derived from the ICAT meta-base. as well as a graph illustrating this data. Similarly, Figure 2 presents an accounting an an illustration of the data derived from the CERT advisories. Both figures show that applications are the primary target of attacks and the kernel is a secondary target. Both figures also show that exploits are the overall most common means of attack.

4.2 Consequences of Attack

Figure 3 contains an accounting of the consequences of attack as derived from the ICAT meta-base data and a graph illustrating this data. Likewise, Figure 4 presents the data derived from the CERT advisory data. Again, applications and the kernel, respectively, are the most frequently targeted system components. However, the ICAT data shows a denial of service to be the most likely consequence while the CERT data shows root access to be the most likely consequence of an attack.

The disagreement between the two data sets is attributable to the selection process of each. The ICAT data set contains information regarding all attack types, whereas CERT only publishes a security alert for attacks that are widespread and of serious consequence.

4.3 Location of Attack

Figure 5 contains an accounting of the location of attack as derived from the ICAT meta-base and a graph illustrating this data. Figure 6 contains a similar accounting but is derived from the CERT advisory data. Both data sets are in agreement, showing that most attacks originate from a remote location.

Our analysis of the CERT and ICAT data, shows that exploits are the most common overall means of attack and are directed against applications from remote locations. According to the CERT data, *root* access is the most common consequence of an exploited vulnerability while the ICAT data shows that a denial of service to be the most common consequence. As previously stated, this discrepancy (root access vs. denial of service) is a

result of CERT issuing advisories only in cases where those vulnerabilities are of great and widespread consequence, with root access being the gravest of consequences. Whereas ICAT does not discriminate according to impact or severity and collects vulnerability advisories from multiple sources. Accordingly, the severity level of the vulnerabilities in the CERT advisories are *High* whereas this is not necessarily the case with the ICAT data.

Both ICAT and CERT show that processes running in kernel-space to be the second most likely component to be attacked.

5 Target-Centric Taxonomy

Because an IDS has no knowledge of the attacker's motivation or the tools employed to conduct the attack we believe that to be successful, the IDS needs to focus on evaluating the information which is readily available. Therefore, our taxonomy is classified according to features and characteristics directly observable at the target. Our feature set is predicated upon the result of our analysis. Our target-centric taxonomy follows:

1. **Target of Attack**. The system component that is the target of an attack.
   (a) Network. The attack is inclusive of the layers of the protocol stack, but does not leave the protocol stack. For example, a *SynFlood* attack "half-opens" multiple TCP sessions in an attempt to exhaust system resources.
   (b) Kernel-Space. A process executing as part of the operating system, either compiled into the kernel or a module that is loaded into and executed by the kernel. An example of an attack of this type is a heap overflow attack directed against the cfsd_calloc function of Solaris' cachefsd (the NFS/RPC file system cachefs daemon). It the attack is successful it allows the remote attacker to execute arbitrary code by using a request with a long directory and cache name.
   (c) Application. An application running outside of kernel space. The application could be running with root privileges or with user privileges. The recently discovered *Apache Web Server Chunk Handling* vulnerability illustrates an attack within this category.
   (d) Other. Any component not included above. Examples include printers and modems. Specific examples include the Alcatel Speed Touch ADSL modem and the 3Com HomeConnect cable modem both of which are vulnerable to denial of service attacks via malformed data packets.
2. **Means of Attack**. The method that was used by the attacker as is manifested at and experienced by the target. This category is further divided as:
   (a) Input Validation. An input validation vulnerability exists if some malformed input is received by a hardware or software component and is not properly bounded or checked. This category is further classified as: a boundary overflow.
      i. Buffer Overflow. The classic buffer overflow results from an overflow of a static-sized data structure.
      ii. Boundary Condition Error. A process attempts to read or write beyond a valid address boundary or a system resource is exhausted.
      iii. Malformed Input. A process accepts syntactically incorrect input, extraneous input fields, or the process lacks the ability to handle field-value correlation errors.
   (b) Exploits. General exploits are vulnerabilities such as race conditions or undefined states in a hardware or software component that lead to performance degradation and/or system compromise. Exploits include:
      i. Exceptional Condition. An error resulting from the failure to handle an exceptional condition generated by a functional module or device.
      ii. Race Condition. An error occurring during a timing window between two operations.
      iii. Serialization Error. An error that results from the improper serialization of operations.
      iv. Atomicity Error. An error occurring when a partially-modified data structure is used by another process; An error occurring because some process terminated with partially modified where the modification should have been atomic.
   (c) Configuration. Vulnerabilities that result from some mis-configuration or lack of proper configuration.
3. **Consequences of Attack**. The end result of the attack. This category is further divided as:
   (a) DoS. The attack results in a Denial of Service to the users of the system.
   (b) User Access. The attack results in the attacker having access to some services on the target system.
   (c) Root Access . The attack results in the attacker having complete control of the system.

(d) Loss of Confidentiality. The attack results in the users of the system loosing privacy of their data.

(e) Other. The attack results in the compromise of data integrity or other undesirable characteristic.

4. **Location of Attack**. The location of the attacker. Indicated by whether the attacker is connected via the network or local host. These are defined as:

   (a) Remote. The attacker does not need to be "virtually" present at the target.

   (b) Local. The attacker needs to be "virtually" present at the target.

   (c) Remote/Local. The attacker may be either local or remote to the target.

6 From Taxonomies to Ontologies: The case for ontologies

In [23], Ning et. al propose a hierarchical model for attack specification and event abstraction using three concepts essential to their approach: *System View, Misuse Signature* and *View Definition*. Their model is based upon a thorough examination of attack characteristics and attributes. However, their model is encoded within the logic of their proposed system. Consequently, it is not readily interchangeable and reusable by other systems.

Similarly, the Intrusion Detection Working Group of the Internet Task Force has defined the *Intrusion Detection Message Exchange Format Data Model* (IDMEF) [5] to describes a data model to represent information exported by IDS's and by individual components of distributed IDS's. Although the IDMEF specification states: "... the Intrusion Detection Message Exchange Format is intended to be a standard data format that automated intrusion detection systems can use to report alerts about events that they deem suspicious" it also specifies the architecture of an Intrusion Detection System and models some attacks. IDMEF uses the Extensible Mark-up Language (XML) [30] to encode the data model, consequently, due to XML's limitations, the data model is not contained within the XML declarations but rather in the logic of how the particular IDS interprets the XML declarations.

Because IDMEF is specified in an XML Document Type Definition (DTD) [8] it does not convey the semantics, relationships, attributes and characteristics of the objects which it represents. Moreover, XML does not support the notion of inheritance.

In commenting on the IETF's IDMEF, Kemmerer and Vigna [14] state *"it is a but a first step, however additional effort is needed to provide a common ontology that lets IDS sensors agree on what they observe"*.

According to Davis et. al [6] knowledge representation is a surrogate or substitute for an object under study. In turn, the surrogate enables an entity, such as a software system, to reason about the object. Knowledge representation is also a set of *ontological* commitments specifying the terms that describe the essence of the object. In other words, *meta-data* or data about data describing their relationships.

Frame Based Systems are an important thread in knowledge representation . According to Koller, et al., [16] Frame Based Systems provide an excellent representation for the organizational structure of complex domains. Frame Based Languages, which support Frame Based Systems, include RDF, and are used to represent ontologies. According to Welty et. al [32] at its deepest level an ontology subsumes a taxonomy. Similarly, Noy and McGuinness [24] state the process of developing an ontology includes arranging classes in a taxonomic hierarchy.

The relationship among data objects may be highly complex, however at the the finest level of granularity, the *Knowledge Representation* of any object may be represented as an *RDF* (Resource Description Framework) statement [4] which formally defines the RDF model as:

1. A set called *Resources*.
2. A set called *Literals*.
3. A subset of Resources called *Properties*
4. A set called *Statements*, where each element is a triple of the form:
   $\{sub, pred, obj\}$
   Where *pred* is a member of Properties, *sub* is a member of Resources,
   and *obj* is either a member of Resources or a member of Literals.

Figure 7 illustrates the basic RDF model.

Additionally, the relationship between a set of objects may be described graphically (as in Figure 9), as a series of *N-triples*, or by an RDF statement.

While RDF defines a model for describing relationships among objects in terms of properties and values, the declaration of these properties and their corresponding semantics are defined in the context of RDF as an RDF schema (RDFS) [26]. In applying RDFS to the problem of intrusion detection the power and utility of RDFS is not simply in representing the attributes of the attack, but rather the in the fact that we can express the relationships between collected data and use those relationships to deduce that the particular data represents an attack of a particular type.

Moreover, specifying an ontology decouples the data model representing an intrusion from the logic of the intrusion detection system. The decoupling of the data model from the IDS logic, specifying it as an ontology, enables non-homogeneous IDS's to share data without a prior agreement as to the semantics of the data. To effect this sharing, the ontology is made available and if

the recipient does not understand some aspect of the data it obtains the ontology in order to interpret and use the data.

Ontologies therefore, unlike taxonomies, provide powerful constructs that include machine interpretable definitions of the concepts within a specific domain and the relations between them. In our case the domain is that of a particular computer or a software system acting on the computer's behalf in order to detect attacks and intrusions. Ontologies may be utilized to not only provide IDS's with the ability to share a common understanding of the information at issue but also further enable the IDS with improved capacity to reason over and analyze instances of data representing an intrusion. Moreover, within an ontology characteristics such as cardinality, range and exclusion may be specified and the notion of inheritance is supported.

6.1 Target Centric Ontology

Figure 8 presents a high level graphical illustration of our target-centric ontology that is built upon our taxonomy. An ellipse is used to denote a subject and object while an arc represents the predicate (relationship). Note the addition of the node labeled *Input* which is a superclass of the taxonomic items *Component*, *Means* and *Location*. Accordingly, an intrusion is **comprised** of some input **resulting** in some consequence, while the input is **directed** towards a a system component, **received** from some location and **causes** some means of by inducing some system behavior. Figure 9 presents our complete ontology in graphical form. Instances of data are represented at the leaves of the graph.

IDMEF, in contrast to an ontology represented by RDF, must work within the constraints imposed by XML, which only provides a syntax for communicating that an attack of a particular type has occurred. IDMEF does not directly contribute to or facilitate the detection and reasoning process. Specifically, once the attack has been detected, and its type, source and target identified, IDMEF only provides a format for communicating information concerning the event. How this information is interpreted and used is solely dependent upon the meaning imposed by the receiver of the information – which may or may not be the same as was intended by the originator of the communication. This is not the case with an ontology. The benefit of the ontology is that everyone that uses the ontology imparts the same semantic meaning on instances of the ontology. Moreover, an ontology is easily extensible as new attack types can be added as subclasses.

The following example of a distributed attack illustrates the utility of our ontology within a system of distributed intrusion detection systems.

6.1.1 The Attack

The Mitnick attack is multi-phased consisting of a Syn/Flood attack, TCP sequence number prediction and IP spoofing. The attack incorporates yet another attack, Syn/Flood, to effect a denial of service attack on a specific host that has a trust relationship with target of the attack. In the following example **Host B** is the ultimate target and **Host A** is trusted by **Host B**. The attack is as follows:

1. The attacker initiates a Syn/Flood attack against **Host A** to prevent **Host A** from responding to **Host B**.

2. The attacker then attempts to open multiple TCP connections to the target, **Host B** in order to be able to predict the values of TCP sequence numbers generated by **Host B**.

3. The attacker then pretends to be **Host A** by spoofing **Host A**'s IP address and sends a Syn packet to **Host B** in order to establish a TCP session between **Host A** and **Host B**.

4. Because its input queue is full due to the half open connections caused by the Syn/Flood attack, **Host A** cannot send *RST* message to **Host B** in response to the Syn message sent by the attacker purporting to be from **Host A**/.

5. Using the calculated TCP sequence number of **Host B** (recall that the attacker did not see the Syn/ACK message sent from **Host B** to **Host A**) the attacker sends an *Ack* with the predicted TCP sequence number packet in response to the *Syn/Ack* packet sent by **Host B**.

6. **Host B** is now in a state where it believes that a TCP session has been established with a trusted host **Host A**. The attacker now has a one way session with the target, **Host B**, and can issue commands to the target.

Figure 10 illustrates this attack.

6.1.2 Detecting the Attack

Consider an environment of distributed intrusion detection services where the specific IDS architecture (component type, name, function, etc.) is abstracted by the SHOMAR framework [29]. Moreover, consider that all components of the architecture use our ontology which not only specifies the data model for attacks.

Suppose that an IDS has either "learned" or has been presented with an instance of an ontology characterizing normal behavior during TCP connection establishment (i.e.: *Three Way Handshake*). It is important to note that this "normal behavior" instance of our ontology expresses the temporal relationship between the receipt of a

Syn the transmission of an *Ack/Syn* and the receipt of the *Ack* establishing the connection as well as the ordering of TCP packet and fragment numbers.

Suppose that some system **Host A** under observation by the IDS has Σ pending TCP connections in time Δ, where Σ and Δ represent quantitative and temporal thresholds specified in the ontology. Furthermore, suppose that the system has responded with *Syn/ACk* messages but has failed to receive the *Ack* completing the handshake leaving its input queue full. Referring to Figure 9 where the leaves of the tree labeled *TCP Layer, TCP Packet* and *Denial of Service* represent specific instances of data, the IDS now has objects representing the system condition as follows:

- Multiple half open TCP connections.
- A degradation of memory resources because the input queue is full.

This information representing the system condition will result in **Host A**'s IDS reasoning engine inferring that system's condition is a specific instance of our ontology defining a *Syn/Flood* attack. Information about this instance may now be made available to all other IDS within the *Shomar* distributed IDS framework. Specifically, a message to all IDSs in coalition with **Host A** stating the IP Address::Port Number of **Host A** is the target of a denial of service attack and that the attack started at some specific time. Due to the shared ontology each and every IDS in receipt of this message will have a clear understanding of not only its meaning but of its implications.

Now suppose that **Host B** has experienced several connection requests (that in actuality were an attempt to determine its TCP sequence numbers) wherein it immediately responded with *RST* messages. As this behavior is aberrant facts about it are asserted into **Host B**'s knowledge base. Suppose that **Host B** has either established or is about to establish a connection with IP Address::Port Number purporting to belong to **Host A** and that the connection began after the denial of service attack against **Host A** started and before **Host A**'s IDS engine has reported it ending. As our ontology defines an instance stating that the consequence of a denial of service attack is that any communications established with the target of the attack are indicative of a *Mitnick* attack, the IDS operating on behalf of **Host B** will reason that it is also the target of a larger, more encompassing attack.

This example demonstrates the semantic power expressed by the ontology, specifically it conveys the implications that one sequence of events (the *Syn/Flood* attack) may have on another set of events.

7 Implementation

As a proof of concept we are using XSB Prolog to reason over our instances of our target centric ontology. Accordingly, we converted the RDF Schema representing the ontology and RDF statements representing instances of our ontology, into *N-Triples*. The *N-Triples* representing the ontology were asserted into a Prolog knowledge base as rules and the *N-Triples* representing the instances were asserted as facts. Additionally, at the top most level of the knowledge base we asserted a rule set defining how the Prolog reasoner is to interpret the *N-Triples*. The following illustrates the RDFS representation of the class *Syn/Flood attack*:

```
<rdfs:Class rdf:about="&Intrusion_;Syn-Flood"
  rdfs:label="Syn-Flood">
  <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>

<rdf:Property rdf:about="&Intrusion_;memory_degredation"
  a:maxCardinality="1"
  a:range="float"
  rdfs:label="memory_degredation">
  <rdfs:domain rdf:resource="&Intrusion_;Syn-Flood"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&Intrusion_;victim"
  a:maxCardinality="1"
  a:range="boolean"
  rdfs:label="victim">
  <rdfs:domain rdf:resource="&Intrusion_;Syn-Flood"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>

<rdf:Property rdf:about="&Intrusion_;time"
  a:maxCardinality="1"
  rdfs:label="time">
  <rdfs:domain rdf:resource="&Intrusion_;Syn-Flood"/>
  <rdfs:range rdf:resource="&rdfs;Literal"/>

<rdf:Property rdf:about="&Intrusion_;IP_address"
  a:maxCardinality="1
  a:minCardinality="1"
  a:range="cls"
  rdfs:label="IP_address">
  <a:allowedParents rdf:resource="&Intrusion_;IPAddress"/>
  <rdfs:domain rdf:resource="&Intrusion_;Syn-Flood"/>
  <rdfs:range rdf:resource="&rdfs;Class"/>
</rdf:Property>

<rdf:Property rdf:about="&Intrusion_;p__pend_concts"
  a:maxCardinality="1"
  rdfs:label="perc_pending_connections">
  <rdfs:domain rdf:resource="&Intrusion_;Syn-Flood"/>
  <rdfs:range rdf:resource="&rdfs;Resource"/>
</rdf:Property>
```

Likewise a RDF representation of an instance of a *Syn/Flood* attack is illustrate as follows:

```
<Intrusion:TCPLayer rdf:about="&Intrusion;Syn-Flood"
 Intrusion:IP_Address="198.162.10.12"
 Intrusion:Time="23:12:34:12 10-10-2002"
 Intrusion:Victim="1"
 Intrusion:p_pend_concts="40"
 Intrusion:memory_degredation="20.0"
 rdfs:label="Syn-Flood"/>
```

Continuing with our example of Hosts **A** and**B**. The IDS responsible for **Host A** monitors attributes such as pending network connections, memory usage, open connects, etc. and uses this information to reason over its

knowledge base as to the "state" of **Host A**. As we have defined an instance of Syn/Flood to exist if pending network connections exceed a certain threshold in conjunction with rapid memory degradation. On detecting **Host A**'s state to be representative of a Syn/Flood attack, the IDS broadcasts this information through the distributed IDS architecture. The message is the same as the RDF instance depicted above.

Because both IDSs share a common ontology, the semantic meaning of the RDF message is meaningful and relevant to the IDS responsible for **Host B** which converts it into *N-Triples* and asserts it into its knowledge base. In turn **Host B**'s IDS queries its knowledge base, to ascertain if the message poses and implications. Pseudo code representative of the query follows:

```
 ∀ predicates P that match:
class = SYN-Flood ∧ Victim = 1, return X
IP = getIP(X, ip), T = getTime(X, time)
compare-My-Connections-Time-IP(T, IP,
MY-CONNECTIONS, Y)
```

If the query returns any connections initiated from **Host A** to **Host B** that began after **Host A** fell victim to the Syn/Flood attack, then **Host B**'s IDS may infer that it is the target of a *Mitnick* attack and take appropriate action.

Once **Host A** is no longer a victim of the Syn/Flood attack the responsible IDS will broadcast this information throughout the distributed system and the instance of that attack will be de-asserted from **Host B**'s knowledge base.

8 Conclusion and Future Work

We have analyzed vulnerability and intrusion data derived from CERT advisories and NIST's ICAT metabase resulting in the identification of the components (network, kernel-space, application and other) most frequently attacked, the means of attack, the consequences of the attack and the location of the attacker. Our analysis shows that non-kernel space (non operating system) applications, running as either root or user, are the most frequently attacked and are attacked remotely. The most common means of attack are exploits other then buffer overflows and other forms of deliberately malformed input data. According to CERT advisories issued in response to severe vulnerabilities, *root* access is the most common consequence of an exploit whereas the ICAT data shows *denial of service* to be the most common consequence.

Our analysis was conducted in order to identify observable and measurable taxonomic characteristics of computer attacks and intrusions. Accordingly, we developed a taxonomy characterized by *System Component, Means of Attack, Consequences of Attack* and *Location of Attacker*. We have stated the case for replacing simple taxonomies with ontologies for use in IDS's and have presented an initial ontology specifying the class *Intrusion*.

We have produced a *target-centric* intrusion ontology that is based upon our *a posteriori* taxonomy. The ontology is represented in RDFS and instances of the ontology are represented in RDF. Out ontology is available at: http://security.cs.umbc.edu/Intrusion.rdfs. We have converted our ontology into N-Triples and have asserted it into a Prolog knowledge base and use Prolog to reason over our rules and assertions to determine the the cause of a given state, which Prolog deductively determines to be a Syn/Flood attack.

Although we have presented our target-centric ontology in terms of RDF, this does not preclude the use of DAML+OIL (DARPA Agent Mark Up Language and Ontology Interface Layer) [11]. DAML+OIL builds on RDF and RDF Schema, extending these languages to include richer modeling primitives.

Currently, we are in the process of identifying unique attributes and characteristics of the identified attack types.

References

[1] Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, and Ed Stoner. State of the Practice of Intrusion Detection Technologies. Technical Report 99tr028, Carnegie Mellon - Software Engineering Institute, 2000.

[2] Edward G. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall PTR, 1994.

[3] Taimur Aslam, Ivan Krusl, and Eugene Spafford. Use of a taxonomy of security faults. In *Proceedings of the 19th National Information Systems Security Conference*, October 1996.

[4] W3C The World Wide Web Consortium. Resource description framework (rdf) model and syntax specification, February 1999.

[5] D. Curry and H. Debar. Intrusion detection message exchange format data model and extensible markup language (xml)document type definition. draft-ietf-idwg-idmef-xml-07.txt, June 2002. expires December 19, 2002.

[6] Randall Davis, Howard Shrobe, and Peter Szolovits. What is knowledge representation? *AI Magazine*, 14(1):17 – 33, 1993.

[7] Robert L. Glass and Iris Vessey. Contemporary application-domain taxonomies. *IEEE Software*, pages 63 – 76, July 1995.

[8] XML Schema Working Group. XML Schema. http://www.w3c.org/XML/Schema, 2000.

[9] Biswaroop Guha and Biswanath Mukherjee. Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions. In *IEEE Networks*, pages 40 – 48. IEEE, July/August 1997.

[10] Joshua W. Haines, Lee M. Rossey, Richard P. Lippman, and Robert K. Cunningham. Extending the darpa off-line intrusion detection evaluations. In *DARPA Information Survivability Conference and Exposition II*, volume 1, pages 77 – 88. IEEE, 2001.

[11] J. Hendler. DARPA Agent Markup Language. http://www.daml.org, 2000.

[12] John Howard. *An Analysis of Security Incidents on the Internet*. PhD thesis, Carnegie Mellon University, 1997.

[13] Lalana Kagal, Jeffrey Undercoffer, Anupam Joshi, and Tim Finin. Vigil: Enforcing Security in Ubiquitous Environments . In *Grace Hooper Celebration of Women in Computing 2002*, 2002.

[14] Richard A. Kemmerer and Giovanni Vigna. Intrusion detection: A brief history and overview. *Security and Privacy a Supplement to IEEE Computer Magazine*, pages 27 – 30, April 2002.

[15] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, 1999.

[16] Daphne Koller and Avi Pfeffer. Probabilistic Frame-Based Systems. In *Proceedings of the Fifteenth National Conference on Artifical Intelligence*, pages 580 – 587, Madison, Wisconsin, July 1998. AAAI.

[17] Ivan Krusl. *Software Vulnerability Analysis*. PhD thesis, Purdue, 1998.

[18] Ulf Lindqvist and Erland Jonsson. How to systematically classify computer security intrusions. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 154 – 163. IEEE, May 1997.

[19] Richard Lippmann, David Fried, Isaac Graf, Joshua Haines, Kristopher Kendall, Davind McClung, Dan Weber, Seth Webster, Dan Wyschogrod, Rober Cunningham, and Marc Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Offline Intrusion Detection Evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition,2000*, pages 12 – 26, January 2000.

[20] John McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, November 2000.

[21] John McHugh, Alan Christie, and Julia Allen. Intrusion detection implemenation and operational issues. CERT, January 2001.

[22] Peter G. Neumann and Donn B. Parker. A summary of computer misuse techniques. In *Proceedings of the 12th National Computer Security Conference*, pages 396 – 407. National Institute of Standards, October 1989.

[23] Peng Ning, Sushil Jajodia, and Xiaoyang Sean Wang. Abstraction-based intrusion in distributed environments. *ACM Transactions on Information and Systems Security*, 4(4):407 – 452, November 2001.

[24] Natalya F. Noy and Deborah L. McGuinnes. Ontology development 101: A guide to creating your fisrt ontology. Stanford University.

[25] Victor Raskin, Christian F. Hempelmann, Katrina E. Triezenberg, and Sergei Nirenburg. Ontology in information security: A useful theoretical foundation and methodological tool. In *Proceedings of NSPW-2001*, pages 53 – 59. ACM, ACM, September 2001.

[26] RDF. Resource description framework (rdf) schema specification, 1999.

[27] George Gaylord Sumpson. *Principals of Animal Taxonomy*. Columbia University Press, 1961.

[28] Jeffrey Undercoffer, Filip Perich, Andrej Cedilnik, Lalana Kagal, and Anupam Joshi. Centarusu2: A Secure Infrastructure for Service Discovery and Delivery in Pervasive Computing. *MONET: Special Issue on Security*, 2002.

[29] Jeffrey Undercoffer, Filip Perich, and Charles Nicholas. Shomar: An open architecture for distributed intrusion detection services. Technical Report TR CS-02-14, University of Maryland, Baltimore County, September 2002. available at: http://security.umbc.edu/pubs/shomar.pdf.

[30] W3C. Extensible markup language.

[31] inc WEBSTERS, editor. *Merriam-Webster's Collegiate Dictionary*. Merriam-Webster, Inc., tenth edition, 1993.

[32] Chris Welty. Towards a semantics for the web. Vassar College, 2000.

[33] M. Wood and M. Erlinger. Intrusion detection message exchange requirements. draft-ietf-idwg-requirements-08, August 2002.

Component	Input Valid	Exploit	Config
Network	17	35	2
Kernel-Space	284	479	98
Application	1272	1367	255
Other	69	158	12

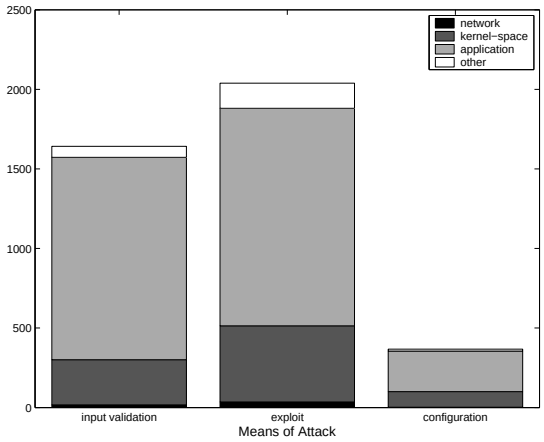

Figure 1. Means of Attack from the ICAT Meta-base

Component	Input Valid	Exploit	Config
Network	3	5	0
Kernel-Space	38	52	4
Application	74	68	18
Other	2	4	0

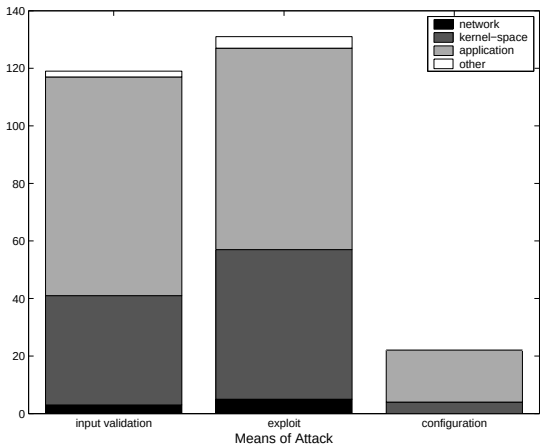

Figure 2. Means of Attack from the CERT/CC Advisories

Component	DoS	User	Root	Conf	Oth
Network	33	4	4	6	7
Kernel-Space	198	60	354	84	165
Application	1056	200	633	721	284
Other	74	10	43	41	71

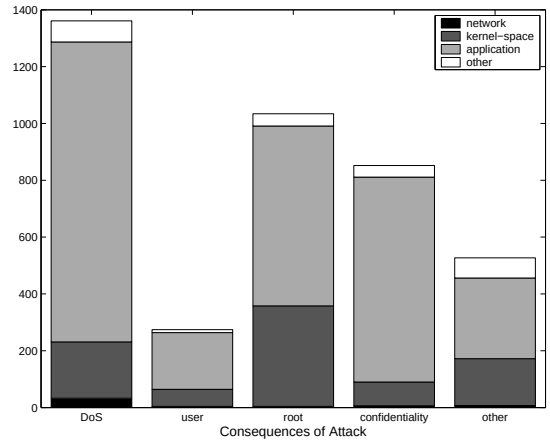

Figure 3. Consequences of Attack from the ICAT/Meta-base

Component	DoS	User	Root	Conf	Oth
Network	8	0	0	0	0
Kernel-Space	15	6	56	7	8
Application	22	22	83	20	15
Other	2	1	1	1	1

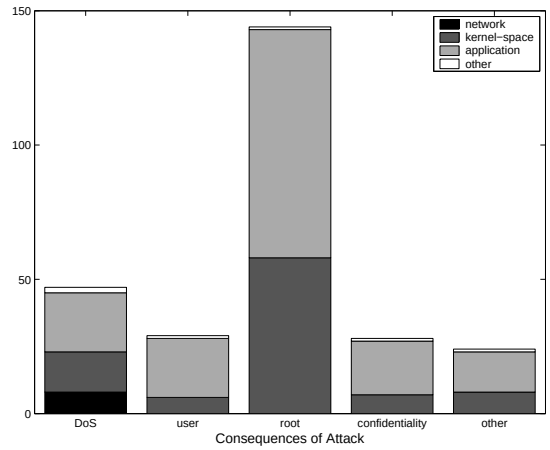

Figure 4. Consequences of Attack from the CERT/CC Advisories

Component	Remote	Local	Remote/Local
Network	50	8	4
Kernel-Space	254	551	56
Application	1905	741	240
Other	137	82	20

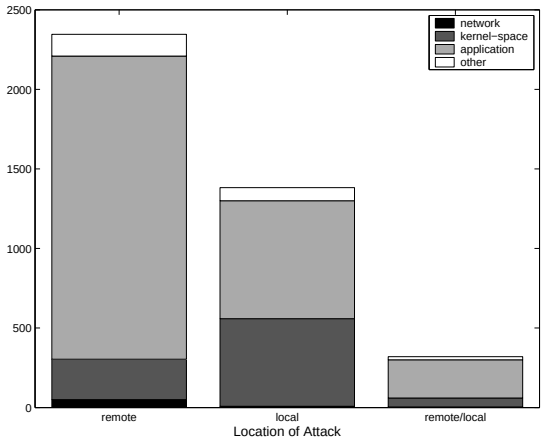

Figure 5. Location of Attack from the ICAT Meta-base

Component	Remote	Local	Remote/Local
Network	7	0	1
Kernel-Space	36	48	8
Application	101	53	8
Other	2	2	2

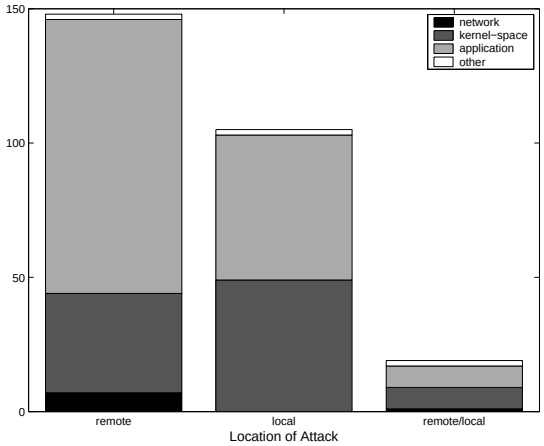

Figure 6. Location of Attack from the CERT/CC Advisories

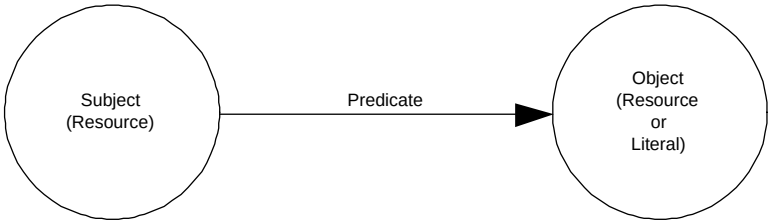

Figure 7. RDF Graph

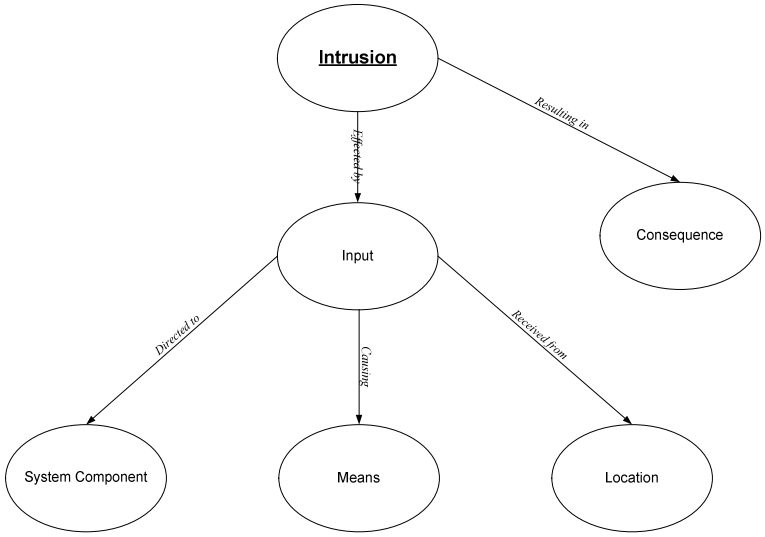

Figure 8. High Level Illustration of the Target-Centric Attack Ontology

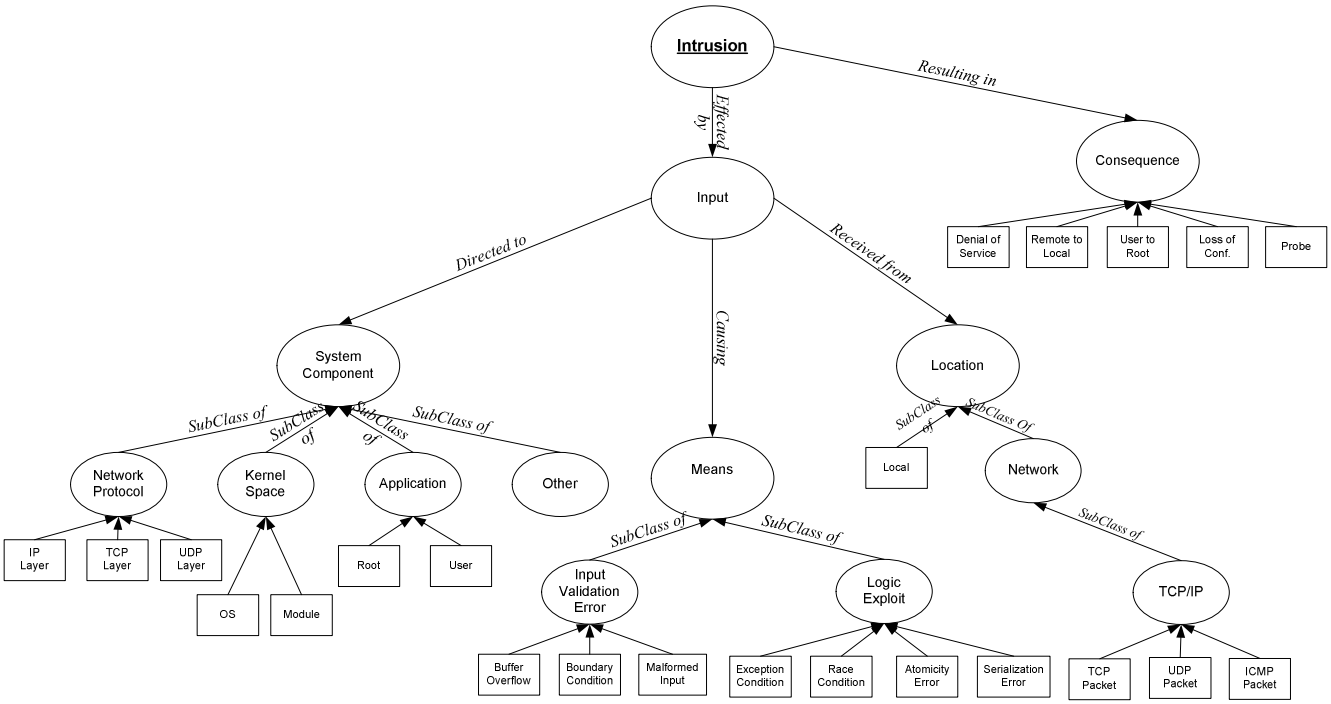

Figure 9. Graphical Presentation of the Target-Centric Attack Ontology

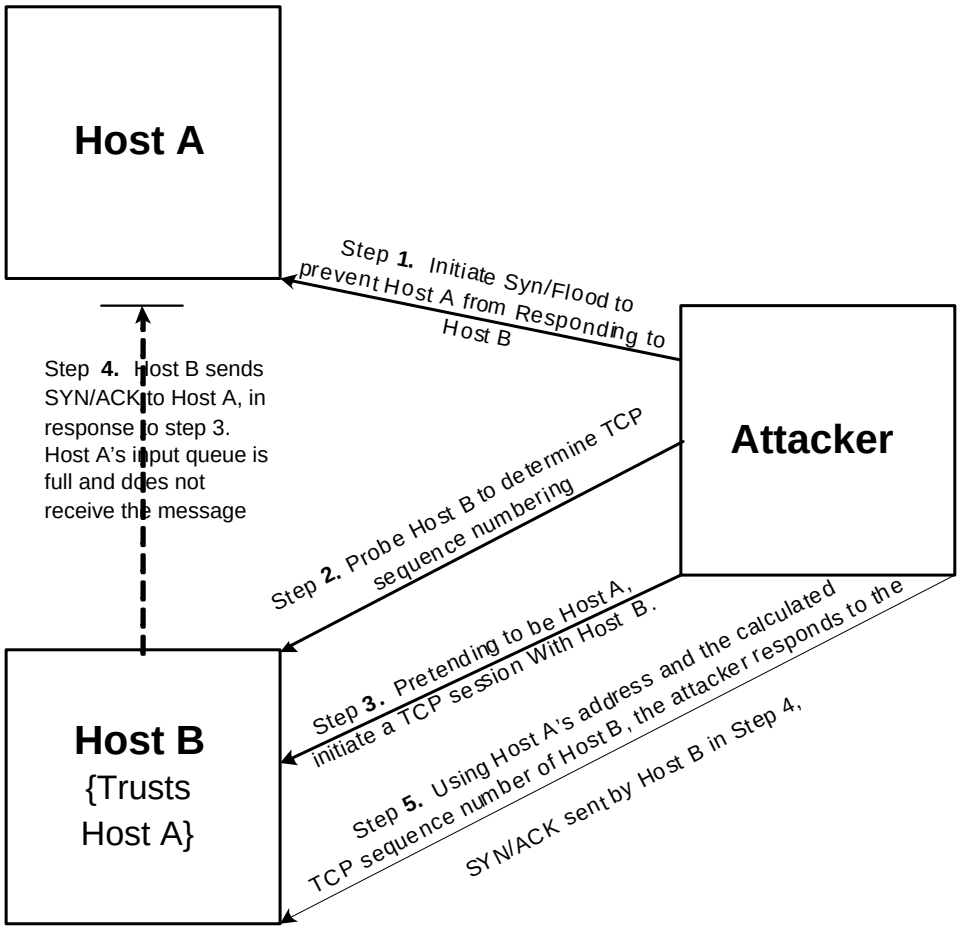

Figure 10. Illustration of the Mitnick Attack