Formal Approaches to Denial of Service Verification
SE 547 Dr. Riely
March 15, 2005

Abstract

With the continued growth of electronic commerce, voting, emergency, governmental and other informational systems the threat of massive network disruptions has never been more real. Hence, the development of denial of service countermeasures must be expanded. Recent work has shown that modifying known methods for verification (i.e. cryptographic protocol verification) prove useful in developing new ways of verifying availability properties. This paper seeks to articulate some of the current trends in formal approaches to attacking the denial of service problem.

1. INTRODUCTION

In February 2000 [4] and again in January 2003 [16] the Internet fell victim to devastating service disruption attacks. These attacks were massive in scope and were perpetrated by large sets of small distributed systems. These systems launched a distributed denial of service attack that turned out to be quite successful against very large, thought to be well-secured, systems. Both of these attacks underscore the vulnerability of the Internet and in turn, the vulnerability of systems which rely upon it.

However, establishing secure systems, impervious to a denial of service attack, is an open problem which continues to grow in complexity. In fact, the cost of mitigating denial of service attacks is on the rise. As reported in [12] the costs associated with virus and denial of service attacks is outpacing the cost of intellectual property theft. Mirkovik and Reiher [18] have established a taxonomy which illustrates this complexity issue and attempts to define the problem space in a manageable way. Their approach combined with the development of formal models will enhance protocol verification, in terms of availability properties, and provide the necessary foundation for building secure protocols resistant to denial of service.

In [17] Meadows emphasizes this need for constructing protocols with resistance to denial of service attacks and suggests a generalized formal framework. Her framework has been expanded in [1,15,21]. This paper seeks to articulate some of the formal approaches to denial of service prevention and to define what is necessary in a model for reasoning about denial of service. The outline of the paper is as follows. Section 1 defines denial of services and shows how it relates to the notion of liveness. Section 2 describes a

few currently available formal approaches to the problem. Section 3 presents the requirements for a workable denial of service model given the discussion of section 2 and 3. Section 4 draws conclusions on the current state of the problem.

1.1 DOS and DDOS OVERVIEW

The term denial of service was first used in [13]. Gligor defined the concept of denial of service in terms of operating systems and a shared user construct. He defined the concept in terms of a shared resource which has legitimate and illegitimate users. The legitimate users of a given service expect to be able to use a service within a set time parameter. Gligor refers to this as the *Minimum Wait Time* (mwt). The illegitimate users are able to perform a denial of service attack if they can cause legitimate users, who request use of the service, to wait longer that the mwt.

Most of the denial of service attacks today are distributed in nature and are conducted over a network. This variant of the denial of service attack is commonly referred to as a distributed denial of service attack. Large number of slave hosts are used to attack one or more hosts. There is a trend for denial of service attacks to "piggyback" on other types of attacks suck as virus or Trojan Horse attacks. Attackers can utilize one exploit to compromise several machines with the intent of executing a distributed denial of service at some later date. This was precisely how Slammer (Sapphire) [16] was executed.

The attack profile of a denial of service exploit is defined in [6]. The main types of attacks are consumption of scarce resources, destruction or alteration of configuration information and physical destruction or alteration of network components. The last two are straight forward in that if attackers are able to destroy or alter configuration information it is trivial to disrupt network traffic. For example, changing routing

information would easily cause a denial of service. Furthermore, if an attacker could change DNS entries access to specific machines through DNS queries would undoubtedly be disrupted. Physical destruction or alteration is also straight forward. Destroying equipment or simply turning equipment off would definitely lead to a loss of service.

The first attack method is a little more interesting since it could be divided into four distinct categories [6]:

1. Network Connectivity – These attacks usually attack memory resources (network buffers) of the victim. The idea here is to start communication connections with the intent of not completing the exchange. Hence, connections are left open and resources are consumed. This process is continued until all available resources are used. An example of the attack is the "SYN flood" [8].

2. Resource Redirect – In this type of attack an attacker will redirect resources of the victim back to the victim. An example of this type of attack is described in [7]. This attack utilizes UDP, the echo and chargen services to cause a denial of service.

3. Bandwidth Consumption – For this attack an attacker with push large amounts of traffic onto a network. The typical traffic in this scenario is ICMP ECHO requests. Most likely, this type of attack would be distributed in nature.

4. Consumption of other resources – Attackers would try to consume an inordinate amount resources in terms of processes (CPU) or disk usage. For example, mail servers could become inoperable in the event that excessive numbers of messages are sent to a specific node [cert]. This type of attack also includes account "lockout". Furthermore causing systems to crash by sending unexpected data could also constitute a denial of service.

1.2 Availability and DOS

The central structures to computer security can be described as confidentiality, integrity and availability. Thus, as defined by Bishop in [5] availability is the ability to use the information or resource desired. Hence, attempts to disrupt availability are described as denial of service. The availability of a resource is determined by the time period in which the service is meant to be used.

1.3 TCP vs. UDP

An interesting point to illustrate at this point is that in terms of the network protocol used to initiate a denial of service attack the “SYN Flooding” attack is rather straight forward to defend against. “SYN Flooding” uses weaknesses in the TCP protocol. TCP communication sessions start with an exchange known as the “three-way handshake” [24]. A typical TCP connection begins with a client sending a packet, with the SYN flag set, to the server. The server will respond with a packet setting the SYN and ACK flags and the client will complete the hand-shake by responding with an ACK. In a “SYN Flooding” attack an attacker will send multiple SYN packets with spoofed source address information. The server will send SYN-ACK packets in an attempt to acknowledge the connection requests. However, since the source information is spoofed the connection request will go unacknowledged. As these request mount up the connection buffer is filled and the server is unable to accept new connections from legitimate users.

Most modern firewalls are able to block this type of attack. For example, Cisco Systems Firewall and Routers have a function call “TCP Intercept” which intercepts SYN packets and proxies to the server side [10]. Ckeckpoint has a similar technology entitled “SYN Defender” [9]. This type of mitigation is theoretically possible due to the fact that the availability requirement is converted from a liveness property into a safety property. Liveness and Safety properties will be defined later in section 1.4. However, by converting to a safety property there is a definable threshold for tcp connection requests. Once this threshold is reached the network device empties the buffer and hence, averts a resource exhaustion.

It has recently been suggested that DOS attacks are switching from TCP to UDP [19]. The reason for this is that UPD can be used to simply flood a network with unwanted requests (e.g. ICMP ECHO requests). This type of flood can overwhelm network devices and is commonly referred to as the Smurf Attack [7]. This type of an attack is easy to block by simply turning off broadcast addressing at the perimeter router and setting your firewall to block ICMP ECHO requests.

However, there are legitimate applications in wide use today which utilize UDP including VOIP (Voice over IP) [11] and some video applications, like RealVideo[23]. It is best to filter all requests to UPD services such as ECHO and CHARGEN, but filtering connections to legitimate services is difficult. Thus, attacks aimed at UPD driven applications would be difficult to mitigate and, since UDP is connectionless, the application layer of a protocol would have to deal with the denial of service problem. The reason for this is that the network devices will not have the ability to determine connections based on any of the information provided in the protocol header.

1.4 LIVENESS VS SAFETY PROPERTY

To distinguish the varying security threats there has been a historical tendency to divide them into three categories. Needham describes these in [20]. He defines them as breach of confidence, failure of authenticity and unauthorized denial of service. In fact, the first two threats have been studied profusely, but with denial of service there is a lack of formal research available. The reason for this lies in the fact that there are variables that we are unable to quantify. For example, given a typical network service the number of possible clients is difficult to determine. At any given moment in time a service could

attract any number of requests. Furthermore, the location of these requests given the state of today's Internet is difficult to predict. The communication flow is also difficult to predict and it is not possible to predetermine client behavior. Hence, clients could participate within the protocol in a friendly manner or could seek to disrupt the communication flow by sending invalid or bogus information.

This is precisely what the notion of liveness in [3] seeks to define. Schneider and Alpern imply that liveness properties suggest that "something good" eventually happens during a program execution. Conversely, a safety property is defined by stipulating that a "bad thing" does not occur within program execution. Hence, for a given execution of a program a liveness property will require that eventually a "good thing" occurs. So in the case of denial of service there is no way to decide if the connections coming in are legitimate or not. It is difficult to determine when a liveness property is violated since it is always possible for the "good thing" to happen in the future.

An interesting thing to note here is that in the earlier idea of the "TCP Intercept" and "SYN Defender" the liveness availability is converted to a safety property. The "bad thing" is reached when the half-opened connection threshold is met. In this way, the infinite collection of possible events is bounded. The point at which the "bad thing" occurs is uniquely identifiable and recognizable. For this reason firewalls and routers can be instructed to act upon safety properties, but not liveness properties. With liveness properties there is always the chance that the "good thing" will happen (e.g. SYN-ACK will be ACKed).

2. FORMAL APPROACHES

As we have seen there is a direct need for a formalized model for discerning denial of service vulnerability. The following discussions reviews three approaches to the denial of service problem. The first is a framework devised by Catherine Meadows which will try to determine costs for actions within a given protocol run. By analyzing these costs Meadows suggests that it is possible to determine if a protocol is susceptible to a denial of service attack. The next group discussed formulates a logic-based approach which extends the operation views of Meadows. They take Duration Calculus and Interval Logic and apply this to real-time system analysis. By claiming that availability requirements can be modeled the group attempts to show how a formal proof could be constructed used to show that these availability requirements hold under attack from attackers of a given strength. Finally, the last group surveyed examined the Just Fast Keying protocol and presents a theorem for denial of service protection requiring round-trip communication before committing to high cost computation. The theorem is articulated in the Applied Pi Calculus.

2.1 COST BASED and FAIL-STOP

In [17] Meadows has developed a framework for reasoning about denial of service n terms of network services. We will use a simplified notation in that a defender or service will be defined as S and the attacker will be defined as E. Thus, a denial of service is defined in Meadows as follows:

> If S is capable of falling victim to a denial of service attack from E within a protocol P, then there exists an interaction between S and E, which is allowed by P, where E uses more of $S's$ resources than S has available.

This is the classic definition of a denial of service attack in terms of over-consumption of scarce resources.

Throughout the paper Meadows uses an Alice-and-Bob notation for describing protocols. Given the following two definitions it is possible to build annotated protocol specifications in such a way as to capture the necessary points for discussion.

> **Definition 1** An Alice-and-Bob specification is a sequence of statements in the form $A \rightarrow B : M$ where A and B are processes or services and M is a message.

> **Definition 2** An annotated Alice-And-Bob specification is a sequence of statements in the form $A \rightarrow B : T_1,...\mathrm{T_k} \,||\, M \,||\, O_1,...,O_n$ where A and B are processes or services, M is a message, and $T_1,...,T_k$ and $O_1,...,O_n$ are sequences of operations performed by A and then by B.

The first sequence with T are the operations performed by leading up to the production of the message, M , and then sending the message to B. The second sequence with O are the operations that B performs in processing and verifying the message.

Within this framework Meadows has defined the notion of event. Events are distributed across three categories or classifications. The first grouping is described as Normal events. These events occur between the sender and receiver within the protocol structure. There are also verification events which happen at the receiver station only. These events are used to verify the message. Furthermore, there are accept events which are also occur only at the receiver's end. These events are precisely performed when B accepts the message.

The other idea which Meadows needs for enabling her framework to be able is an expanded version of fail-stop from Gong and Syverson [14]. Gong and Syverson developed the idea of a fail-stop protocol which simply implies that a protocol is fail-stop when an improper event is detected. Thus, Meadows says that if an attack interfering with

a message sent in a single step has the ability to cause all following messages to not be sent then the protocol is fail-stop. Meadows takes this idea and adds to it a specification for costs by defining a function Γ which maps actions within a given protocol to costs. Therefore, a protocol is considered fail-stop if a service cannot be made to participate in an exchange up to and including an event X unless the attacker can deliver effort greater than $\Gamma(A)$. This is precisely the fail-stop condition.

The goal is to decide if there are sequences of events within a given protocol which produce the outcome of the defender having a higher cost than the attacker. If this condition exists then it would imply that the attacker could efficiently produce and send messages to the receiver who, in turn, would have to process these messages. Since the receiver's cost is higher than the sender's there exists the scenario for a denial of service. This is a reasonable assumption since the attacker could continue to send these messages to the receiver at a faster rate than the receiver could process and respond to them.

To model within this framework it is necessary to define a cost set and assign a mapping which defines how events are related to costs. For this analysis Meadows has supplied a few basic definitions. These include a cost set, C, which contains cost definitions like expensive, medium, cheap and free. Furthermore, there is a cost function, δ, which maps every event to a single cost. Finally, there is a message processing cost function defined as δ'. This is defined by Meadows as the cost of processing a message up to a failed verification event. There is the possibility that the cost sets for the service and attacker could be different.

Given this notion of events and cost functions a final relation is needed to determine if the outcomes are acceptable within the model or not. Meadows provides for

this in her definition of a tolerance relation which is a subset of pairs of costs crossed with intruder capabilities. Hence, given a set C of costs and a set G of attacker capabilities (set of actions available to the attacker) the subset consists of a pairs of (c, g). Thus, it is stipulated that for a pair (c', g') to be within the subset it follows that $c' \leq c$ and $g' \geq g$. This tolerance relation shows the efforts, in terms of cost c, that is used to protect against an attacker of strength g. In real terms the idea of effort could translate into money or equipment outlays used to protect against a particular attacker sophistication. The tolerance relation defines the granularity of the analysis system.

The steps to assess a protocol's denial of service security is divided into four sections. The first is to define a cost function and decide on what the attacker capabilities will be. Things like forging return addresses, reading messages and breaking encryption are all examples of attacker capabilities. The next step is to build a tolerance relation. The tolerance relation is really how much you are going to spend to provide a specific level of security. Furthermore, it is a statement as to what level of insecurity are you willing to tolerate at a given amount of cost.

Step three is the actual calculating and comparison step. It is accomplished by calculating the attacker capability function Γ. Meadows defines this process in the following manner:

> **Part 1**: If E_1 is an event immediately preceding a verification event E_2, in a line of a protocol, then $(\delta'(E_2), \Gamma(E_1))$ is in the tolerance relation.
> **Part 2 :** If E is an accept event, then $(\Delta(E), \Gamma(E))$ is in the tolerance relation.

This implies that if the communication is able to reach the point where E_2 fails or succeeds then the cost is $\delta'(E_2)$. Thus, no intruder who has capability $\Gamma(E_1)$ could have

disrupted or "interfered" with M after event E_1. In Part 2 the $\Delta()$ is fully defined in Definition 13 of [17], but it simply means that if the event to be measured is an accept event the cost is calculated by summing the costs of all preceding events leading to the accept. These two steps provide a way to verify if the protocol has the ability to force a denial of service upon the service provider before and after the verification of the message.

It must be noted that this framework has a few flaws. The definition of cost may be too vague. Furthermore, using fail-stop protocols as the base model is not robust enough. Fail-stop protocols do not generalize very well to real-world protocols. Constructing the attacker capability function and the tolerance relations are also very difficult and can be interpreted in multiple ways. It is also impossible to model protocols that do not have a finite number of definable steps which do not vary.

Meadows has tried to capture the essence of a denial of service attack based upon relative costs to the receiver versus the capabilities of an attacker. This is a general framework which could be embellished to reason about specific protocols and, in fact, others have done so in [1,15,21]. However, this framework is useful in that it provides a generalized base for reasoning about denial of service in a way that could lead to the development of tools for model checking protocols for availability properties. In fact, Meadows suggests that existing tools could be enhanced to include this framework.

2.2 INTERVAL LOGIC

Hansen and Sharp have used interval logics to analyze availability properties in [15]. In [21] they have expanded on this notion and provided a concrete example of using

interval logics with Meadows costs framework and Paulson's inductive approach in [22] to devise several definitions and theorems on availability. Their work is moving towards a development of a verification tool which will verify availability properties. They show several results by formulating availability requirements within the cost structure of Meadows and assigning the costs in terms of execution times for the verification events. Furthermore, their approach makes use of a timed model which can be used for reasoning about availability properties as the amount of time a service is available to answer requests and whether or not a service has the ability to detect malicious activity and recover within a given time period.

In general, the types of attacks that are used include passive and active attacks. Both attack distinctions are defined in terms of the attackers. Passive attacks are executed by attackers that have the ability to read, but not change or inject, messages into the network. Active attacks are perpetrated by attackers with the ability to change or inject messages into the network.

Paulson's work in [22] is not applicable to the verification of availability since it does not work for timed events. However, the combination of Meadows and Paulson make it possible to formulate verifiable proofs in the area of availability. The protocol specifications are written in a similar Alice-and-Bob specification that was employed by Meadows. Hence, $A \rightarrow B : M$ implies that Message M is sent from A to B.

They have also devised a notion of a packet. Packets are used to send messages across networks and this is defined by $(p_s, i_s) \rightarrow (p_r, i_r) : M$. Each principal has both a persona (defined by p) which could be false and an identity (defined by i) which is true in all cases. It is possible to fake a persona, but not an identity.

There are also three types of events in terms of the network communication. They are defined as send, receive and block. Send events are when one principal sends a message to another principal and a receive event is the act of reading the packet from the network. The block event is the act of removing the packet from the network.

The use of Interval Logics provides a foundation for reasoning about the timing requirements of systems and how processes are scheduled. In [15] the real-time modeling is using functions of time that are based on real numbers and return boolean values such that a system is in state X at time t iff $X(t) = 1$. Hence, this type of function is referred to as a state variable.

Furthermore, these state variables can be combined into state expressions. These state expressions can have a notion of duration over an interval. This logic enables the reasoning of timing processes of a system in a mathematically sound way. Therefore, proofs can be built leading to methods for verification.

Hansen et al. have developed several availability requirements from this logic. In using Duration Calculus and Interval Logic they are able to formulate liveness properties. The reason for this is that by trying to articulate liveness properties which express the idea that "something good will eventually happen" is difficult without a notion of expanding modalities or the ability to reason about program traces which inductively constructed.

Their availability requirement examples utilize a definition for good and bad processes. Trusted entities or the good processes are defined as γ, and the bad processes of hostile intruders are defined as β. Thus, the following assumption is given:

$$\pi = \beta \cup \gamma \text{ and } \beta \cap \gamma = \emptyset$$

There is no overlap between good and bad processes. The two sets are disjoint.

An example of an availability requirement is given as follows.

$$\forall i \in \gamma.(\Diamond_r \int Run_i = C_i)$$

This is described as "for all good processes, for some interval of time from present to future they will all complete". Each process $i \in \gamma$ is given a portion of the processing time C_i which is the measure of requested processing time for process i. Run_i is defined as a state variable: $Run_i : Time \rightarrow \{0,1\}$. Furthermore, it is in a running state under the following logic.

$$Run_i(t) = 1 \text{ iff process } i \text{ is running at time } t$$

A further requirement is stated as a server's ability to respond to normal requests for a fraction of a set time interval. In their logic it is described as:

$$\Box(\ell \geq T \Rightarrow (\sum_{i \notin \gamma} \int Run_i) \leq (1-x) \cdot \ell)$$

This states "that for all subintervals" the server is available to answer requests in given large time length T. It is shown that there will be an accumulation of requested processor time from several untrusted processes, but for a fraction x, the server should be available.

It is also necessary to model the attackers since, as the authors point out, attackers will consume an arbitrary amount of processing time dependent on their strength. Hence, for a credible availability analysis the model must have a method to model variable strength attackers. This includes a notion of verification and, subsequently, rejection time which is precisely the time required to verify and reject bad messages. A process of high strength has the ability to request more processor time as compared to a lower strength process.

Hence, this logic can be used to estimate service availability. Hansen and Sharp illustrate this in [15] with the understanding that a precise lower and upper bound for the running time of good processes is achievable. By adding the cost structure of Meadows, apply this timing logic directly and taking into account that a service could act as either a sender or receiver, Hansen and Sharp have defined the total computation time as the total time it takes for the protocol to successfully complete or an error is detected. They have shown that for a multiple step protocol a set of expressions can be derived which can be used to show computation time for good and bad processes with varying strength.

They have also developed in [15,22] the proof support in an Isabelle/HOL environment. With the formalisms of Paulson in terms of performing induction over traces of protocol executions they have established a verification tool to examine protocols for availability requirements. Pilegaard, Hansen and Sharp provide several proofs encoded in Isabelle/LSILHOL in [22].

2.3 APPLIED PI CALCULUS AND AVAILABILITY

In [1] Abadi, Blanchet and Fournet have used there applied pi calculus to reason about denial of service in relation to the Just Fast Keying (JFK) [2]. They use the applied pi calculus to model JFK and illustrate several security properties. From this they have generalized a theorem for ensuring denial of service protection.

The authors surmise that the main problem with denial of service prone services is that a server will end up accepting a first message for protocol initiation without authenticating the sender. However, to prevent a denial of service a service must ensure

that incoming connections are legitimate and not devote processing resources until the authenticity of the initiator is verified.

In [1] the authors have developed a theorem for articulating this notion of denial of service resistance. The theorem specifies that resources should only be committed after a round-trip communication session has been established by the initiator. The theorem has two related properties which are used to constrain the protocol so that both the initiator and the responder do not perform an expensive computation or state saving procedure. The theorem is listed below.

> **Theorem 1 (Protection from DOS) Let** $A \in C$. [1]
> 1. Let $S_{\$}$ be S with an additional output $\bar{\$}\langle N_I \rangle$ before the Diffie-Hellman computation k_I in I_0^A.
>
> For any trace $S_{\$} \xrightarrow{\eta} S'$, for each output $\bar{\$}\langle N_I \rangle$, there are distinct, successive actions $init^A(\_,\_), \bar{c}\langle 1(N_I,\_)\rangle$, and $c\big(2\big(N_I,\_,\_,\_,\_\big)\big)$.
>
> 2. Let $S_{\$}$ be S with an additional output $\bar{\$}\langle N_I, N_R \rangle$, before the Diffie-Hellman computation k_R in R_3^A.
>
> For any trace $S_{\$} \xrightarrow{\eta} S'$, for each output $\bar{\$}\langle N_I, N_R \rangle$, there are distinct, successive actions $c\big(1\big(N_I,\_\big)\big), \bar{c}\big\langle 2\big(N_I, N_R,\_,\_,\_\big)\big\rangle$, and $c\big(3\big(N_I, N_R,\_,\_,\_,\_,\_\big)\big)$.

I_0^A and R_3^A refer to processes that are communicating in the JFK protocol. The first is the initiator and the second is the responder (server). For the entire JFK specification see [1]. Property 1 refers to the message setup before the initiator computes the Diffie-Hellman shared key and Property 2 refers to the setup before the responder computes the Diffie-Hellman shared key. Both sides gain an increased level of denial of service protection. Furthermore, each N is a freshly created nonce used to verify the authenticity of each message.

Given a JFK specific configuration, defined by S, which is an implementation of the protocol, in this case JFK. With Property 1 we see that the initial nonce (N_I) is sent out and the three successive actions are that an initiation message is sent out followed by the sending of a message with the nonce on channel c. The very next action is that the nonce should be received back on channel c. Hence, a round-trip communication has taken place from the initiator's point of view.

With Property 2 we see that both the initiator's nonce and the responder's nonce are sent out. The three actions associated with this from the responder's point of view begins with receiving the nonce from the initiator on channel c. Then the responder sends out a message with both nonces and waits for a reply confirming a round-trip communication on channel c. The freshly created nonces are used to confirm a round-trip communication from the responder's point of view and thus, both sides have denial of service protection. In [1] the Properties 1 and 2 have been automatically been verified with the automatic proof verifier ProVerif.

For protocols establishing a round-trip communication is necessary for guaranteeing denial of service protection since neither side should commit to expensive processing before establishing that the communication is coming from a legitimate principal.

3. REQUIRMENTS FOR MODELING DOS

Thus, from the discussion of section 2 it is evident that to model denial of service the cost framework of Meadows can be built upon in both operational and logic-based approaches to reason about availability properties. Defining attackers and legitimate

processes within a timed structure is also important since most of the modern protocols utilize some timing constraints.

From all of these definitions and theorems is evident that since availability is defined as a liveness property it is difficult to formulate a general model. However, they seem to collectively capture the notion that to verify availability it is necessary to separate out the parts of a protocol that are expensive from the parts that are easily computed. Hence, if a service can be tricked into setting aside resources for verifying and computing based upon easily constructed false initial messages then a denial of service conditional is not only probably, but imminent.

4. CONCLUSIONS

Meadows showed that by defining a cost structure and evaluating a protocol using these defined costs one could find parts of a protocol that could produce a denial of service. Hansen et al. took these findings of Meadows and extrapolated them to include a logic-based argument for verifying availability properties. Their discussion was formulated using Duration Calculus and Interval Logic. This logic-based approach enabled them to reason about inductively defined traces of a protocol a developed several availability requirements. Finally, Abadi et al. proved a theorem on denial of service protection which established that principals within a protocol should not commit resources before verifying that a protocol initiation is from a legitimate source. This was done using their Applied Pi Calculus and the JFK protocol.

Collectively these definitions and theorems show that to reduce or eliminate the risk of denial of service a protocol must enable end points to verify legitimate traffic from

false requests. However, although these definitions are useful in their specific contexts it will still be difficult to apply the results of the above authors in a general application. Defining cost functions within Meadows' framework is indeed difficult. Furthermore, the approach from Hansen, Pilegaard and Sharp is mathematically sound, but requires a great depth of understanding in the area of modal and interval logic with the ability to use formal proof verification tools. It will be difficult to abstract protocols and specific implementations into their framework.

Abadi, Blanchet and Fournet have an edge on the others in that they have applied their technique to an actual protocol. Their theorem for protection from denial of service could be generalized to other protocols since the idea of establishing round-trip communication before resource commitment could be added to other protocols. Future work undoubtedly requires a combination of the results surveyed in this paper.

References

[1] M. Abadi, B. Blanchet, C. Fournet, "Just Fast Keying in the Pi Calculus", Programming Languages and Systems: 13th European Symposium on Programming, ESOP 2004, Springer-Verlag (March 2004).

[2] W. Aiello, et al. "Efficient, DoS-resistant, Secure Key Exchange for Internet Protocols", Proceedings of the 9th ACM conference on Computer and Communications Security, ACM Press, 2002.

[3] B. Alpern, F. B. Schneider, "Defining Liveness", Technical Report 85-650, Cornell University, Ithica, New York, 1985.

[4] L. Arent, D. MuCullagh, "A Frenzy of Hacking Attacks", Wired Online, February 2000.
http://www.wired.com/news/business/0,1367,34234,00.html

[5] M. Bishop, "Computer Security: Art and Science', Addison-Wesley, ISBN: 0201440997, 2003.

[6] CERT CC. Denial of Service Attacks.
http://www.cert.org/tech_tips/denial_of_service.html

[7] CERT CC. Smurf Attack.
http://www.cert.org/advisories/CA-1998-01.html

[8] CERT CC. TCP SYN flooding and IP spoofing attacks.
http://www.cert.org/advisories/CA-1996-21.html

[9] CHECK POINT. SYN Defender
http://www.checkpoint.com/press/1996/synattack.html

[10] CISCO Systems. TCP Intercept
http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/intercpt.htm

[11] CISCO Systems. VOIP and UDP
http://www.cisco.com/en/US/tech/tk652/tk701/technologies_white_paper09186a008009294d.shtml
http://www.cisco.com/en/US/products/hw/routers/ps221/prod_quick_installation_guide09186a00800dc749.html

[12] CSI/FBI, "Ninth Annual CSI/FBI Computer Crime and Security Survey", Computer Security Institute, 2004.

[13] V. Gligor. "A Note on the Denial-of-Service Problem", Proceedings of the 1983 Symposium on Security and Privacy, IEEE Computer Society Press, 1983.

[14] L. Gong, P. Syverson, "Fail-stop Protocols: An Approach to Designing Secure Protocols", Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications, 1995.

[15] M. R. Hansen, R. Sharp, "Using Interval Logics for Temporal Analysis of Security Protocols", Proceedings of the 2003 ACM workshop on Formal methods in security engineering, ACM Press, 2003.

[16] IEEE Security and Privacy, "Inside the Slammer Worm" http://www.computer.org/security/v1n4/j4wea.htm

[17] C. Meadows, "A Cost-based Framework for Analysis of Denial of Service in Networks", Journal of Computer Security, Volume 9, Number 1-2, 2001.

[18] J. Mirkovic, P. Reiher, "A Taxonomy of DdoS Attack and DdoS Defense Mechanisms", ACM SIGCOMM Computer Communications Review, Volume 34, Number 2, April 2004.

[19] J. Nazario, Arbor Networks, Ann Arbor, Michigan. Trends in Denial of Service http://monkey.org/~jose/presentations/ddos.d/

[20] R. M. Needham, "Denial of Service: An Example", Communications of the ACM, Volume 37, Number 11, 1994.

[21] H. Pilegaard, M. R. Hansen, R. Sharp, "An Approach to Analyzing Availability Properties of Security Protocols", Nordic Journal of Computing, Volume 10, Number 4, 2003.

[22] L. C. Paulson, "The Inductive Approach to Verifying Cryptographic Protocols", Journal of Computer Security, Volume 6, Number 1-2, 1998.

[23] Real. RealVideo and UDP http://service.real.com/firewall/adminfw.html

[24] W. R. Stevens, B. Fenner, A. M. Rudoff, "Unix Network Programming: The Sockets Networking API", Volume 1, Third Edition, Addison-Wesley, ISBN: 0131411551, 2004.