Formal Modelling and Automatic Detection of Resource Exhaustion Attacks

Bogdan Groza
Politehnica University and Institute e-Austria
Timişoara, Romania
bogdan.groza@aut.upt.ro

Marius Minea
Politehnica University and Institute e-Austria
Timişoara, Romania
marius@cs.upt.ro

ABSTRACT

Many common protocols: TCP, IPSec, etc., are vulnerable to denial of service attacks, where adversaries maliciously consume significant resources of honest principals, leading to resource exhaustion. We propose a set of cost-based rules that formalize DoS attacks by resource exhaustion and can automate their detection. Our classification separates excessive but legal protocol use (e.g., flooding) from illegal protocol manipulation that causes participants to waste computation time without reaching the protocol goals. We also distinguish simple intruder intervention leading to wasteful execution from DoS attacks proper, which can be repeatedly initiated. Our rules can highlight attacks that are undetectable by the targeted honest agents, or by all protocol participants. We have successfully tested an implementation of the methodology in a validation platform on relevant protocol examples, in what to the best of our knowledge is the first formal automated analysis of DoS attacks.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol verification*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; C.4 [**Performance of Systems**]: *Reliability, availability and serviceability*

General Terms

Security, Verification

Keywords

denial of service, formal modeling, automated verification

1. INTRODUCTION

Protocols that base their security on cryptographic primitives are indispensable nowadays. However, from a computational perspective, not all cryptographic primitives are cheap. While secret-key primitives can be executed in microseconds on modern computers, public-key primitives require a thousand time more computational steps and can cause resource exhaustion even on well equipped servers. The problem is far-reaching: on low computational power devices, such as sensors, mobile phones, embedded devices, etc., the unjustified execution of even simple cryptographic primitives can cause resource exhaustion.

In recent years, many protocols have been found vulnerable to this kind of attacks and modified variants or countermeasures have been proposed. In this paper we focus on the automatic detection of these attacks, where honest participants can be maliciously determined to perform expensive operations, such as public-key encryptions or signatures, while the adversary consumes significantly less resources.

From the point of view of protocol execution, we consider useful to separate resource exhaustion DoS attacks in two main categories:

- Resource exhaustion DoS attacks due to *excessive use*. These are attacks in which there is no abnormal use of the protocol, however the adversary as participant consumes significantly less resources than other principals thus being capable to cause a DoS. Typical examples are attacks on the server side, such as flooding, spam, etc., which do not violate the protocol specification but can exhaust resources of honest principals.

- Resource exhaustion DoS due to *malicious use*. In these attacks the adversary manages to bring the protocol to an abnormal state (principals are not aware of their correct identities, shared keys do not match, etc.) from which the protocol goals cannot be correctly met. Many protocols with such vulnerabilities exist, probably the best known is Lowe's attack [13] on the STS protocol [8], which we will use as one case study.

Of course, in general *denial of service* (proper) requires repetition, and one condition for this to take place is the ability of the intruder to control the initiation of a session. This condition is sufficient in the case of protocols vulnerable to excessive use since there is no abnormal protocol behaviour in this case, and thus an honest principal cannot detect being under attack (the only prevention is to limit the use of the protocol). A commonly used solution to protect the server side are proof-of-work protocols based on moderately hard one-way functions, known as cryptographic puzzles or client puzzles. In this context, several protocols have been augmented with such constructions, including e-mail [9], TCP [12], authentication protocols [3], etc.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

However, we are also concerned with the second, abusive kind of attack. In this case, not only are resources spent, but this is done without achieving the protocol goals. For this case we also express additional conditions under which we can determine whether the attack is or not noticeable by honest principals. We consider these attacks more severe and their automatic detection to be especially relevant.

As for the causes of resource exhaustion attacks, our analysis of existing protocols reveals two main design flaws:

- Unbalanced costs between participants. This can cause DoS inflicted both by adversaries and honest principals and usually affects the server side. A common engineering practice adopted in protocols to overcome this is the aforementioned use of client puzzles.
- Lack of authenticity for the exchanged messages. This allows adversaries to intrude and compromise a protocol at lower cost than that incurred by honest participants. For example, the adversary can inflict an unjustified computational cost on some honest principal, or even steal a principal's computational work.

Conversely, as our case studies show, most protocols that do not exhibit these flaws are also resilient to DoS attacks. Therefore, checking for these two vulnerabilities should be good engineering practice for designing DoS-resilient protocols. However, DoS attacks can be intricate, and protocol designs can thus benefit from formal analysis.

To detect DoS attacks, we have expressed our rules in the ASLan specification language of the AVANTSSAR toolset [4], a successor to the AVISPA project [5], which is analyzed by the back-ends CL-Atse [25], OFMC [6] and SATMC [2]. To enable automatic detection, we need first to integrate costs in protocol transitions and adversary actions and second to define rules which use the cost and state information to identify an attack. We have shown that our rules can be easily integrated into existing models and work effectively with representative case studies: STS [8], JFK [1] and some variants of these protocols, providing an automated analysis of resource exhaustion attacks.

1.1 Related work

The first and best-known approach to formalize DoS attacks is the cost-based framework developed by Meadows [18] who also relates DoS resilience to the fail-stop property introduced by Gong and Syverson [11]. Ramachandran [21] has applied this framework to the JFK protocol and other protocol fragments, and also links the fail-stop property to the non-interference property proposed by Focardi et al. [10]. Later, a more in-depth analysis of JFK was done by Smith et al. [22] who found some attacks to which we refer in the case studies section. All these analyses were done by hand, whereas our focus is on the formalization of attack conditions in rules that can be used in a toolset for automated protocol validation, which to the best of our knowledge has not been done before.

Besides formalizing attack states, other research has focused on improving protocols to withstand DoS attacks and formalizing criteria that need to be met by a protocol for resilience to DoS. Matsuura and Imai [16] provide a strategy for resilience in three-pass authentications (similar to STS) while later work of the same authors performs an explicit cost calculation of computational resources involved [17]. A related protocol is also proposed by Tseng [24].

As for resilience criteria, the most common principle to avoid DoS is that the initiator of the protocol must consume more resources than the responder. This is the principle behind proof-of-work protocols.

Five criteria for DoS resilience are given in [23]. They include the aforementioned cost comparison indirectly by allowing the responder to react only to legitimate requests, where a request is legitimate only if the initiator has solved a puzzle which consuming more computational resources than the responder. However the criteria stated in [23] are informal and relations between them are not clearly established. For example, a server should "not perform any expensive operations with a client unless" (1) "it is convinced the client is trying to make a legitimate connection" and (2) "it is convinced that the client wants to talk to B and not another server M". However, a request made to a different server than the intended one should not be considered a legitimate attempt, thus criterion 2 is implied by criterion 1. Also criterion 4 states that "a malicious party must use a very large amount of resources" if it wishes to flood the server, however such a condition is too restrictive for many practical protocols where the sufficient condition is that the server will not waste more resources than the client. In particular, Meadows' framework and subsequent analyses use a tolerance relation which a allows a more flexible way to define the limit of resource exhaustion. As the criteria are stated informally, without individual examples for each of them, some imprecision is unavoidable. The authors also give a formal definition of service resilience for secure key agreement protocols. This definition appears problematic as well, illustrating the need to properly track costs incurred by principals. According to [23], a protocol is defined as DoS-resilient if each server only performs expensive operations in a session that follows an "acceptable pre-session" in which the client performs the proof of work. The problem with this definition is that it ignores the server cost in the session, where a resource-exhausting computation may occur even after the server has protected itself in the pre-session. Thus, the definition as stated will fail to correctly identify DoS attacks.

In contrast, we consider that one cannot consider protocol resilient unless it keeps track of the server cost in each session. Our restriction of balanced cost between protocol participants covers criterion 4 from [23], while the first three criteria are covered by our condition for malicious executions. This includes the oft-forgotten criterion (3) that the work of some honest principal cannot be stolen by a malicious adversary.

Regarding proof of work by client puzzles at an algebraic level, this criterion is included by the stricter requirement of unforgeability of client puzzles formalized in [7]. It should be noted that even if a protocol does not allow the work of a honest client to be stolen, the adversary might still be able to forge puzzles for which a honest client will waste computational time to solve. The property of unforgeability requires that an adversary is unable to produce a valid puzzle, thus it offers stronger security, but proving this property requires security reductions that are not meant to be handled symbolically in our framework. Thus, the property that the work of a client cannot be stolen ensures safety for the server side but cannot guarantee protection against resource exhaustion on the client side, a case often neglected by protocol designers. The rules we introduce also detect and flag this case as *malicious use*.

2. FORMALIZATION AND REASONING

2.1 Protocol and cost model

First we define the protocol model for which we formalize DoS attacks. We give a simplified account of the ASLan specification language of the AVANTSSAR toolset [4], on which we base our implementation.

Definition 1. A *symbolic protocol description* is a triple $\mathcal{P} ::= (\mathit{InitialState}, \mathit{TransitionRule}^*, \mathit{AttackState}^*)$ formed by an *initial state*, a set of *transition rules* and a set of *attack states*, where:

i. the *initial state* is a conjunction of ground facts (terms),

ii. a *transition rule* has the form $\mathcal{LHS} \Rightarrow \mathcal{RHS}$ where $\mathcal{LHS}$ and $\mathcal{RHS}$ are conjunctions of facts, and $\mathcal{LHS}$ may also contain negative facts, $\mathtt{not}(\mathit{Fact})$,

iii. an *attack state* is a conjunction of positive and negative facts (like a $\mathcal{LHS}$).

Sending a message, e.g., $A \to B : M_K, sig(inv(pk_A), M)$ may be modelled with two rules as shown in Figure 1 and an initial state: $\mathtt{stateA(a,3,0).stateB(b,3,0)}$. In ASLan, we denote conjunction of facts $F_1 \,.\, F_2$ with a dot.

$$\mathtt{stateA(A,ID,0)} \Rightarrow \mathtt{stateA(A,ID,1).iknows(enc(K,M).sig(inv(pkA),M))}$$

$$\mathtt{stateB(B,ID,0).iknows(enc(K,M).sig(inv(pkA),M))} \Rightarrow \mathtt{stateB(B,ID,1)}$$

Figure 1: Protocol fragment in ASLan

Here `stateA` and `stateB` are facts tracking state progress in the two roles, and `a`, `b` are principals that can play roles A and B in a session with identifier 3. Sending a message M is modelled by making it available to the intruder, `iknows(M)`; conversely, the intruder might place any known term (including the actual message sent) on the communication channel.

Semantics. A symbolic protocol description $\mathcal{P}$ defines a transition system $M = (\mathcal{S}, \mathcal{I}, \to)$, where $\mathcal{S}$ is the set of states, $\mathcal{I}$ is the set of initial states, and $\to \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation, defined as follows.

Let $\mathcal{T}$ be the set of all terms, and $\mathcal{F}$ the set of ground facts (Boolean terms). Then the state set $\mathcal{S} = 2^{\mathcal{F}}$ is the powerset of possible ground facts, and the initial states $\mathcal{I}$ are directly defined as a conjunction (set) of ground facts in the protocol description $\mathcal{P}$. Informally, a rule $\mathcal{LHS} \Rightarrow \mathcal{RHS}$ specifies a transition that replaces in the current state the facts in $\mathcal{LHS}$ with those in $\mathcal{RHS}$. Formally, there exists a transition $S \to S'$ iff there exists a rule $PF.NF \Rightarrow R$ (with PF and NF sets of positive and negative facts) and a substitution $\sigma : V \to \mathcal{T}$ (with V the set of variables in PF), such that:

i. $\sigma(PF) \subseteq S$ (positive facts are satisfied)

ii. $(\sigma' \circ \sigma)(NF) \cap S = \emptyset$ for all substitutions σ' such that $(\sigma' \circ \sigma)(NF)$ is ground (i.e., negative facts are false)

iii. $S' = S \setminus \sigma(PF) \cup \sigma(R)$ (positive facts are subtracted and right-hand-side facts are added).

To enable cost evaluation, we extend this definition by adding costs to transition rules as well as to adversary actions.

Definition 2. A *cost-augmented* protocol description $\mathcal{CP}$ is a *symbolic protocol description* where:

i. $\mathtt{cost(P,0)}$ holds for all principals in the *initial state*

ii. *transition rules* have the form
$$\mathcal{LHS}.\mathtt{cost(P,C_1)} \Rightarrow \mathcal{RHS}.\mathtt{cost(P,C_2)}$$
where the fact $\mathtt{cost(P,C_1)}$ denotes that the cost for principal P before the transition is C_1, and $\mathtt{cost(P,C_2)}$ states that the cost after the transition is C_2.

The cost predicate does not identify a particular session of the protocol, since both for the adversary and for legitimate participants, the cost is accumulated over multiple protocol executions.

We need to extend the notion of cost to the abilities of the adversary. The adversary is equipped with the well-known Dolev-Yao abilities over the communication channel: he can intercept, replay, forge or block messages. Besides this, the adversary can use computational abilities, which must be augmented by costs in our model. We consider that costs are induced by the following operations: modular exponentiation, public-key encryption, decryption and signatures, denoted $\mathtt{c_{exp}}, \mathtt{c_{enc}}$, etc. This leads to the protocol rules synthesized in Equations 1–4. If needed, costs can be attached to other primitives and in the same manner to the Dolev-Yao intruder abilities.

$$\mathtt{iknows(X).iknows(Y).cost(i,C_1).sum(C_1,c_{exp},C_2)} \Rightarrow \mathtt{iknows(exp(X,Y)).cost(i,C_2)} \quad (1)$$

$$\mathtt{iknows(K).iknows(X).cost(i,C_1).sum(C_1,c_{enc},C_2)} \Rightarrow \mathtt{iknows(enc(K,X)).cost(i,C_2)} \quad (2)$$

$$\mathtt{iknows(enc(K,X)).iknows(K).cost(i,C_1).sum(C_1,c_{dec},C_2)} \Rightarrow \mathtt{iknows(X).cost(i,C_2)} \quad (3)$$

$$\mathtt{iknows(X).iknows(Y).cost(i,C_1).sum(C_1,c_{sig},C_2)} \Rightarrow \mathtt{iknows(sig(X,Y)).cost(i,C_2)} \quad (4)$$

Until now, we have not specified whether cost has an algebraic value or is modelled symbolically. To fix notions, we use as cost set a monoid, as in Meadows' framework [18]; our approach is also applicable to numeric costs, depending on the capabilities of the employed verifiers.

Definition 3. A cost set is a commutative monoid with operation $+$ over a set S with partial order $<$, such that $x + y \geq x$ for all $x, y \in S$. In particular, we consider the set $S = \{0, low, high, expensive\}$ and the sum defined as $\forall a, b \in S, a + b = max(a, b)$.

We implement this cost structure in our protocol model by defining a fact $sum(X, Y, Z)$ for each pair X, Y where $Z = X + Y$, e.g., $sum(high, low, high)$, etc.

We consider the cost of the attacker and the honest agents directly comparable, by defining a comparison predicate *less*, e.g., $less(cheap, expensive)$, $less(medium, expensive)$. Meadows' framework uses different cost sets for attacker and honest agents, and a relation that defines comparable costs. In our case, the tolerance relation is given by the comparison predicate *less*. We could obtain the same generality by appropriately scaling cost values and/or redefining the comparison. If the cost also depends on the principal, for example, if for the adversary it may be more expensive to write on a channel that is already allocated to a different principal, then these facts can be adapted to include the principal's

identity as well. Thus the tolerance function in the attack condition is flexible and can be adapted according to the requirements of the protocol that is modelled.

2.2 Formalizing the attack condition

It is not straightforward to define and formalize the conditions that characterize a resource exhaustion DoS attack. The main reference so far, the framework of Meadows [18] starts out by extending the definition of fail-stop protocols. The intuition for DoS resilience is that if the adversary successfully interferes in a protocol, his cost at that point must exceed some value related to the cost of the honest agent. We start from the same point of comparing the cost of the adversary and of the honest victim. However, we refine this characterization along several criteria.

The first classification relates to the way in which the intruder interferes with protocol execution, causing the DoS attack. We distinguish attacks due to *excessive* use, in which the adversary only acts as an ordinary participant, but can deplete the resources of honest principals by repeated execution at lower cost. On the other hand, there are instances of *malicious* use where the adversary interferes with the protocol, resulting in at least one message reception differing from normal use (more formally, we consider violation of injective agreement [14]). Previous frameworks, such as that of Meadows do not distinguish along this dimension, since a common definition for interfering with a message is used.

A second dimension relates to forced repetition by the adversary. Strictly speaking, we consider that *denial of service* occurs only if the adversary can repeatedly force a resource-depleting execution, which requires him to be the initiator in the protocol (regardless of whether malicious capabilities are used). To be able to force this repeated resource depletion, the adversary may need to rely on a previous session started by an honest participant. To model this we will also consider the execution of multiple sessions. For the case of *malicious* executions we do not strictly require the adversary to be able to repeat the attack in order to cause DoS, i.e., to be the initiator of the protocol. This is because we consider that not allowing *malicious* executions is a good engineering practice and protocols that do not meet this requirement should be spotted, even if the adversary may not succeed more than once in mounting such an attack.

A third criterion concerns the detectability of the attack by the victim or some other honest participant. This is important, since undetectable attacks cannot be countered by other means than indiscriminately limiting protocol usage.

Thus, the simplest case of resource exhaustion is due exclusively to excessive use: a responder B is vulnerable to a DoS attack if there is some state where the cost accumulated by B is higher than the one accumulated by the initiator A of the protocol, a role that can be potentially played by an adversary. This is a violation of the design rule that the initiator should commit more resources than a responder. In this case, the intruder does not need special abilities, but can simply play the role of A, initiating sessions at will. To identify the principal initiating the protocol we augment in the protocol model the first transition of the initiator role (assumed to be named A) with the fact $\texttt{initiate}(\texttt{A})$. The attack condition is then expressed as follows:

Attack condition 1 (DoS due to excessive use). A protocol is vulnerable to a resource exhaustion DoS attack on some principal P if, in a session initiated by the adversary, an execution state is reached where the cost accumulated the adversary is less than the one accumulated by P, i.e.,

$$\texttt{dos\_exc}(\texttt{P}) := \texttt{initiate}(\texttt{i}).\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (5)$$

The semantics of an attack condition written in this way involves an implicit existential quantification over all variables appearing on the right-hand side.

At this point, the adversary can simply stop executing protocol steps. Repeated protocol executions can lead to resource exhaustion for P, who must observe the responder role, incurring higher cost than the adversary in each execution.

The relation above corresponds to the simple case of a single protocol session; thus, the costs can be tracked per protocol principal rather than per execution instance. To model multiple sessions run in parallel, which also covers the case of a potential distributed DoS (DDoS), we need to track the costs cumulated from sessions initiated by the adversary. This can be achieved by using two distinct transition rules, one for sessions that are initiated by the adversary (cumulating cost) and one for case of sessions between honest participants (where no costs need to be tracked).

$$\mathcal{LHS}.\texttt{initiate}(\texttt{i},\texttt{ID}).\texttt{cost}(\texttt{P},\texttt{C}_1).\texttt{sum}(\texttt{C}_1,\texttt{c}_{\texttt{step}},\texttt{C}_2) \Rightarrow \mathcal{RHS}.\texttt{cost}(\texttt{P},\texttt{C}_2) \quad (6)$$

$$\mathcal{LHS}.\texttt{initiate}(\texttt{A},\texttt{ID}).\texttt{not}(\texttt{equal}(\texttt{i},\texttt{A})) \Rightarrow \mathcal{RHS} \quad (7)$$

We now define the case when a protocol is *maliciously* used, without attaining its intended purpose, wasting computational resources of honest participants (at lower cost to the attacker), even though the protocol might not necessarily be initiated by the intruder or repeatable by it.

To formalize this kind of attack we need to track the correspondence between sent and received messages, and express whether they pair up to a proper protocol run. For this purpose, we augment each send transition with a fact $\texttt{send}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID})$ meaning that message M is sent by agent S at protocol step labeled L in session ID, intended for receipt by R. We also augment every receive transition with a fact $\texttt{recv}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID})$ to denote that message M is received at protocol step L in session ID by receiver R as coming from sender S. For correct protocol runs (injective agreement), it must hold that every $\texttt{recv}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID})$ is preceded by a matching $\texttt{send}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID})$. Conversely, if for some value M this agreement is violated, there is an attack on the protocol functionality (notably, authentication) with respect to principal R:

$$\texttt{tampered}(\texttt{R}) := \texttt{recv}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID}).\texttt{not}(\texttt{send}(\texttt{S},\texttt{R},\texttt{M},\texttt{L},\texttt{ID})) \quad (8)$$

(the other variables are implicitly quantified existentially). In the following rules, we will use `tampered` as a macro with the above definition. With this, we can express the malicious execution of a protocol as follows:

Attack condition 2 (DoS due to malicious use). A protocol is vulnerable to *malicious use* for participant P if it can reach a state where the adversary cost is lower than the cost of P, and P accepts a value which differs from the prescribed protocol execution, i.e.,

$$\texttt{dos\_mal}(\texttt{P}) := \texttt{initiate}(\texttt{i}).\texttt{tampered}(\texttt{P}).\\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (9)$$

The attack conditions for excessive use (5) and malicious use (9) have the same cost comparison, and differ only in the extra condition: malicious use implies tampering with a normal protocol run. We may formulate (9) less restrictively by removing the condition $\texttt{initiate}(\texttt{i})$. As discussed before, even if the run is not initiated by the attacker and thus not repeatable at will, a malicious use signifies bad protocol design, and should be avoided.

As with excessive use, such an attack might appear only after a correct protocol run, for example, when the adversary can initiate a new session and reuse previous values (captured from participants, or used in a previous session). To model this, participant cost must be added over multiple sessions. This is not straightforward because we should not include costs from sessions where the adversary does not interfere, as these are not part of the attack.

For this purpose, we need to track the cost of a principal separately for each session it participates in. We define a predicate $\texttt{scost}(\mathit{Agent}, \mathit{Cost}, \mathit{ID})$ that keeps track of the cost in a particular session, identified by the additional parameter ID. Each protocol rule is now split into three different rules, depending on whether the session has already been interfered with by the intruder. This is represented by the fact $\texttt{bad}(\mathit{ID})$, initially false for each session. As long as the intruder does not intervene, the cost $\texttt{c}_\texttt{step}$ for the current step is added to the per-session cost for the executing agent (rule 10). When a session is interfered with (either it is initiated by the intruder, or there is a receipt of a tampered message), the per-session cost so far and the cost of the current step are added to the accumulated cost for the principal, and the fact $\texttt{bad}(\mathit{ID})$ is asserted (rule 11). Finally, for a session already marked as bad, the cost of the step is directly added to the accumulated participant cost (rule 12).

$$\mathcal{LHS}.\texttt{not}(\texttt{bad}(\texttt{ID})).\texttt{send}(\texttt{S},\texttt{P},\texttt{M},\texttt{L},\texttt{ID})\\ .\texttt{scost}(\texttt{P},\texttt{C}_\texttt{ID},\texttt{ID}).\texttt{sum}(\texttt{C}_\texttt{ID},\texttt{c}_\texttt{step},\texttt{C}'_\texttt{ID}).\\ \Rightarrow \mathcal{RHS}.\texttt{recv}(\texttt{S},\texttt{P},\texttt{M},\texttt{L},\texttt{ID}).\texttt{scost}(\texttt{P},\texttt{C}'_\texttt{ID},\texttt{ID}) \quad (10)$$

$$\mathcal{LHS}.\texttt{not}(\texttt{bad}(\texttt{ID})).\texttt{not}(\texttt{send}(\texttt{S},\texttt{P},\texttt{M},\texttt{L},\texttt{ID}))\\ .\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{scost}(\texttt{P},\texttt{C}_\texttt{ID},\texttt{ID})\\ .\texttt{sum}(\texttt{C}_\texttt{P},\texttt{c}_\texttt{ID},\texttt{C}_1).\texttt{sum}(\texttt{C}_1,\texttt{c}_\texttt{step},\texttt{C}'_\texttt{P})\\ \Rightarrow \mathcal{RHS}.\texttt{recv}(\texttt{S},\texttt{P},\texttt{M},\texttt{L},\texttt{ID}).\texttt{bad}(\texttt{ID}).\texttt{cost}(\texttt{P},\texttt{C}'_\texttt{P}) \quad (11)$$

$$\mathcal{LHS}.\texttt{bad}(\texttt{ID}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{sum}(\texttt{C}_\texttt{P},\texttt{c}_\texttt{step},\texttt{C}'_\texttt{P})\\ \Rightarrow \mathcal{RHS}.\texttt{bad}(\texttt{ID}).\texttt{cost}(\texttt{P},\texttt{C}'_\texttt{P}) \quad (12)$$

Note that in rule 10 the `send` guard is needed (to express correct execution) only in a receive step (with `recv`)on the RHS); otherwise, the rule is written without both facts.

The rules for detecting excessive use (5) and malicious use (9) do not change in the multiple session setting; only updating the cost per principal changes by splitting the rules for protocol steps, as defined above.

As argued before, it is worth distinguishing the above defined vulnerabilities are detectable by the protocol participants. The problem is more severe if, while being depleted of resources, an honest agent cannot tell that this is happening maliciously and merely sees an excessive protocol execution.

Attack condition 3 (Undetectable resource exhaustion). Both excessive and malicious executions are especially dangerous if they are not detected by honest protocol participants as different from normal executions. In this case there is no means of protection other than a potentially unnecessary blanket limitation of executions that affects honest use as well. Participant P cannot detect an attack if it successfully reaches the final state. For repeatable attacks initiated by the intruder, and respectively, for malicious executions, this can be expressed as:

$$\texttt{dos\_exc\_nd}(\texttt{P}) := \texttt{initiate}(\texttt{i}).\texttt{count}(\texttt{P},0).\\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (13)$$

$$\texttt{dos\_mal\_nd}(\texttt{P}) := \texttt{tampered}(\texttt{P}).\texttt{count}(\texttt{P},0).\\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (14)$$

We use $\texttt{count}(\mathit{Agent}, \mathit{Num})$ as a fact to keep track of the number of sessions in which a principal is still active.

Even though the protocol abuse might not be detectable by the participant P under DoS attack, it might be detectable by some other participant Q. This makes it possible to correct the protocol, by adding an extra confirmation from Q to P, which would then also allow attack detection by P.

Thus, it becomes relevant to characterize the DoS attacks in which excessive or malicious of the protocol is not detectable by *any* of the protocol participants, as such protocols are completely vulnerable, without detection:

$$\texttt{dos\_exc\_all}(\texttt{P}) := \texttt{initiate}(\texttt{i}). \bigwedge_{\texttt{Q}\in\texttt{agents}(\mathcal{P})} \texttt{count}(\texttt{Q},0).\\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (15)$$

$$\texttt{dos\_mal\_all}(\texttt{P}) := \texttt{tampered}(\texttt{P}). \bigwedge_{\texttt{Q}\in\texttt{agents}(\mathcal{P})} \texttt{count}(\texttt{Q},0).\\ \texttt{cost}(\texttt{i},\texttt{C}_\texttt{i}).\texttt{cost}(\texttt{P},\texttt{C}_\texttt{P}).\texttt{less}(\texttt{C}_\texttt{i},\texttt{C}_\texttt{P}) \quad (16)$$

Cost of verification steps. One key modification in modelling a protocol for DoS analysis in our framework is that verification steps done by principals also need to be modelled as protocol transitions. As a result, it becomes easy to model proof-of-work protocols which involve the verification of some cryptographic puzzle before continuing the communication with some principal.

Coordinated attackers. The case of coordinated attackers, where certain protocol messages are reused between attackers, is also considered in [22]. In their proposal the cost of a particular reused operation is divided by the number of attackers, for example, if the cost required by some computation is expensive and there are n attackers that can reuse this computation then the cost will be $\frac{1}{n} \times \mathit{expensive}$. Our approach relies on symbolic computation and therefore we cannot use algebraic operations such as divisions or multiplications (although the back-ends that we use can be modified to handle them as well). Still, the reuse of already computed values works with the rules defined for the multiple sessions as these values are already added to the intruder knowledge (thus the intruder will not have additional cost when it reuses them). Thus, in a coordinated attack, attackers that reuse values will have their cost set to null for that particular computation.

Bounding the attack cost. In practice, a model checker may consume a lot of time to verify the existence of an attack. It becomes critical to restrict the arbitrary possible actions of an intruder to those that might lead to an attack. In this case, if one determines the maximal cost $maxcost(\mathcal{CP}, P)$ for principal P over protocol $\mathcal{CP}$ (by exploring the protocol without the intruder), then any DoS attack on principal P needs to have an intruder cost lower than $maxcost(\mathcal{CP}, P)$. Thus, the adversary actions can be limited up to this cost. Bounding the search in this manner can be done for both single and multiple sessions.

3. CASE STUDIES

As a first case study we consider the well-known Station-to-Station protocol [8], also relevant because of its similarities to the Internet Key Exchange protocol (IKE), which is part of IPSec and commonly used in practice. The first attacks on it were reported by Lowe [13], and it has been used as case study by Meadows [18]. The protocol and Lowe's attack are depicted in Figure 2. In this attack A tries to talk to B, but Adv succeeds in impersonating B to A and uses this to initiate his own session with B.

$A \rightarrow B : \alpha^x$
$B \rightarrow A : \alpha^y, Cert_B, E_k(sig_B(\alpha^y, \alpha^x))$
$A \rightarrow B : Cert_A, E_k(sig_A(\alpha^x, \alpha^y))$

A → Adv(B):	α^x
Adv → B:	α^x
B → Adv:	$\alpha^y, Cert_B, E_k(sig_B(\alpha^y, \alpha^x))$
Adv(B) → A:	$\alpha^y, Cert_B, E_k(sig_B(\alpha^y, \alpha^x))$
A → Adv(B):	$Cert_A, E_k(sig_A(\alpha^x, \alpha^y))$

Figure 2: Station to Station protocol. Lowe's attack

This attack is sometimes considered minor since Adv can not actually recover the key shared between A and B. However, both A and B consume computation time performing public-key operations for a protocol run that is of no use to them. In the context of our rules this case study is relevant as both attack conditions, for excessive and malicious use, hold. First, this is an attack on B due to excessive use because he consumes more resources than Adv while Adv is the initiator of the session. Second, this is a case of malicious use as B receives a value from Adv that was actually sent by A. For A, the second attack rule also holds, we have again a case of malicious use, as A receives a value from $Adv(B)$ that was actually sent by B for the communication with Adv.

After modelling this protocol in ASLan two of the backends found attack traces. The CL-Atse model checker [25] reported both attacks from Lowe [13] and verified that the final state of A is reached, thus making the attack undetectable by A. The OFMC model checker [6] reported a distinct attack: an adversary initiating a protocol with some honest principal can replay α^x to honest principals, thus saving himself from performing new computations. If one considers that values like α^x have a uniform distribution then the adversary can initiate conversations with B by simply sending some random value, or use trivial values (1, 2, etc.) for the exponent, thus consuming significantly less resources than honest principals.

The three-pass handshake proposed by Matsuura and Imai in [16] is also based on the Diffie-Hellman key agreement but is secure against the previous attacks because the identities of the principals are included in the exchanged messages, making it infeasible for an adversary to use a message intended for another principal. Indeed, the previous attack on STS would not be feasible if these identities were included in the signature. Matsuura and Imai later proposed versions of the ISAKMP/Oakley and IKE key agreement [17] which are resilient to DoS; here, the identities of principals were included as well. Later, Mao and Paterson [15] in their study of IKE drew again attention to the good engineering principle stated previously by Abadi and Needham: if the identity of the principal is relevant for the message then it is prudent to include this identity in the message. The same principle makes the protocol proposed by Tseng [24] secure against the attack.

The second case study is the Just Fast Keying (JFK) protocol proposed by Aiello et al. [1], a protocol with provable security and specifically designed to withstand DoS attacks. The protocol was first analyzed by Ramachandran [21] and later in more detail by Smith et al. [22]. Both analyses were done using the cost-based framework of Meadows. The protocol was deemed DoS-resilient in [21], provided that malicious messages are handled properly, in particular by using caching to deal with replay messages and thus amortizing the cost from expensive verifications. In [22] two weaknesses against DoS were outlined, the first comes from the reuse of the Diffie-Hellman exponential and the second involves coordinated attackers. Moreover, the paper proposed a new variant that includes client puzzles to increase its resistance. This variant is depicted in Figure 3: the modifications from the original JFK consist in adding k as the difficulty level of the puzzle and sol as the solution of the puzzle, i.e., k indicates the number of leading zeros that $H(token||sol)$ must produce.

$I \rightarrow R : N'_I, g^i, ID'_R$
$R \rightarrow I : N'_I, N_R, g^r, grpinfo_R, ID_R, S_R[g^r, grpinfo_R], token, k$
$I \rightarrow R : N_I, N_R, g^i, g^r, token,$
$\quad \{ID_I, sa, S_I[N'_I, N_R, g^i, g^r, ID_R, sa]\}^{K_e}_{K_a}, sol$
$R \rightarrow I : \{S_R[N'_I, N_R, g^i, g^r, ID_I, sa], sa'\}^{K_e}_{K_a}, sol$

Figure 3: JFK protocol with client puzzles

We have modelled this protocol and no DoS attack was found for the side of the responder R. However, although probably not a concern of the protocol designers, the model checkers reported attacks for the side of the initiator I. Namely, an adversary may initiate sessions with responder R and forward the puzzles that he receives to some honest principal I; the result is that the adversary can continue his communication without wasting computational time for the puzzle, while principal I consumes time to solve a puzzle that was not intended for him. This is a case of malicious use that happens because the puzzle $token$ is not bound by the responder with a signature to the identity of I. The same happens with the value of the difficulty level k which is not signed and can be increased by an adversary to make I spend large amounts of time to solve a more difficult puzzle. Figure 4 shows the relevant parts of an attack trace reported by OFMC; the same attack trace was reported by CL-Atse. In the attack trace, fresh values are written in upper-case and are suffixed by a counter, a general convention in the

$(a,1) \rightarrow i : h(NA(1)).g^{XA(1)}.idb1$
$i \rightarrow (b,1) : h(NA(1)).g^{XA(1)}.idb1$
$(b,1) \rightarrow i : h(NA(1)).NB(2).g^{XB(2)}.idb.sig(inv(pkB), g^{XB(2)}).$
$h(hkb.g^{XB(2)}.NB(2).h(NA(1)).ipi).k$
$i \rightarrow (a,1) : h(NA(1)).NB(2).g^{XB(2)}.idb.sig(inv(pkB), g^{XB(2)}).$
$h(hkb.g^{XB(2)}.NB(2).h(NA(1)).ipi).k$
$(a,1) \rightarrow i : NA(1).NB(2).g^{XA(1)}.g^{XB(2)}.$
$h(hkb.g^{XB(2)}.NB(2).h(NA(1)).ipa).$
$enc_{(h((g^{XB(2)})^{XA(1)}.h(NA(1)).NB(2).1))}[ida.sa.$
$sig_{inv(pkA)}[h(NA(1)).NB(2).g^{XA(1)}.g^{XB(2)}.idb.sa]].$
$h_{(h((g^{XB(2)})^{XA(1)}.h(NA(1)).NB(2).1))}[ida.sa.$
$sig_{inv(pkA)}[h(NA(1)).NB(2).g^{XA(1)}.g^{XB(2)}.idb.sa]].$
$sol(h(hkb.g^{XB(2)}.NB(2).h(NA(1)).ipi))$

Figure 4: Attack trace reported by OFMC

model checkers that we used. The identities of the honest participants are a and b instead of i and r in order to avoid confusion with the intruder which is by default i in the backends that we used. The trace reflects the situation when the adversary forwards the message of some honest I to the responder R and gets a response that includes a puzzle built on its own ip which will be useless for I to solve. The cost of the adversary is zero while the cost of I is *expensive*. An attacker may further profit from the puzzles solved by deceived honest participants to perform a distributed attack on responder R while avoiding the cost of solving the puzzle on its own. To overcome this, unforgeable puzzles [7] should be used or otherwise the protocol should ensure by design that the work of some honest principals cannot be stolen by a malicious adversary, a principle also stated in [23].

All previous work on formal interpretation and modelling of resource exhaustion concerns the same kind of key agreement protocols and all case studies available in the cited literature are, except for the workload, very similar. Therefore the same paradigm can be always employed: ensure that all values are authentic (which includes freshness), and that the client performs more work than the server (potentially adding some adjustable proof of work on the client side).

However, this paradigm is not suitable for protocols of a different nature. A relevant example is the well-known TESLA protocol proposed by Perrig et al. [20], with the problem of DoS attacks considered later in [19]. This protocol is of particular interest due to its use in a constrained environment (sensor networks) where even simple symmetric cryptographic primitives can become a source of resource exhaustion. Therefore, in this case the usual definition of cost, that considers public primitives while symmetric primitives are cheap, must be changed. Of particular interest is the attack noted in [19] as DoS on the key chain. TESLA uses an array of keys generated by the successive composition of a one-way function f over some initial key K_0, i.e., $f(K_0), f^2(K_0), \ldots, f^i(K_0)$. Each new key K_{New} is checked for authenticity by the receiver by verifying that $f^j(K_{New}) = K_{Last}$ for some j, where K_{Last} is the last authentic key received. Now, if an adversary may mislead a receiver to believe that a key is received from a distant future, the receiver may perform too much computation which will eventually exhaust its resources. The problem here is not that the keys sent by the sender are not authenticated, but that checking for authenticity becomes tedious when the keys are distant from the last known authentic key. In this case, the solution is to disallow the receiver to check for more applications of the one-way function than the time elapsed from the last authentic key, divided by the disclosure delay. Thus, good engineering is the solution and not a generic paradigm as was used or proposed in many key agreements protocols. However, modelling a protocol such as TESLA in ASLan is not straightforward as the language does not allow explicit definition of features such as the keychain or the time-triggered nature of the protocol. The AVISPA library of protocols contains a model for TESLA, but restricted to only three rounds and too limited to check for DoS attacks.

4. CONCLUSIONS

Resource exhaustion is a highly relevant class of DoS attacks if an economic viewpoint of security protocols is considered. We have proposed and formalized a set of rules that can be used to automatically detect potential resource exhaustion DoS attacks. Moreover, we have provided a classification of different attack situations: excessive vs. malicious protocol use depending on whether special intruder capabilities are needed; and intruder-controlled initiation vs. one-time malicious execution, depending on whether the intruder can force a repeated attack. In addition, we have characterized the critical cases when attacks are undetectable by the honest protocol participants. We have used these rules, implemented in the AVANTSSAR framework, and shown that this can be effective on two known representative case studies, STS and JFK. To the best of our knowledge this is the first formal automated analysis of DoS attacks.

So far, we have only modelled the computational aspect of selected cryptographic primitives, but one could easily insert cost associated to memory consumption or other resources. As future work, we will consider assigning costs to the capabilities of the Dolev-Yao intruder; however, this would require modelling a custom intruder or changes in the model checkers. We will also consider, as suggested already in [18], deriving the cost annotations automatically from the protocol model, which would significantly ease the usability of our approach and allow its more extensive practical use.

None of the previously published formal models and analyses use detailed algebraic costs, as Maatsura and Imai have done for individual protocols [17]. Indeed, to decide resource exhaustion the simple cost structure proposed by Meadows [18] is sufficient. However, for a more detailed analysis, automating the calculation in [17] is a relevant future goal.

Acknowledgments

This work is supported in part by FP7-ICT-2007-1 project 216471, AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures.

5. REFERENCES

[1] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Just fast keying: Key agreement in a hostile network. *ACM Transactions on Information and System Security*, 7(2):242–273, 2004.

[2] A. Armando and L. Compagna. SAT-based model-checking for security protocols analysis. *International Journal of Information Security*, 7(1):3–32, 2008.

[3] T. Aura, P. Nikander, and J. Leiwo. DOS-resistant authentication with client puzzles. In 8^{th} *International Workshop on Security Protocols (SP'00)*, LNCS vol. 2133, pages 170–177. Springer, 2001.

[4] The AVANTSSAR project. `http://avantssar.eu/`.

[5] The AVISPA project. `http://avispa-project.org/`.

[6] D. A. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *Int'l. J. of Information Security*, 4(3):181–208, 2005.

[7] L. Chen, P. Morrissey, N. Smart, and B. Warinschi. Security notions and generic constructions for client puzzles. In *Advances in Cryptology – ASIACRYPT*, LNCS vol. 5912, pages 505–523. Springer, 2009.

[8] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[9] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology: CRYPTO*, LNCS vol. 740, pages 139–147. Springer, 1993.

[10] R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non interference. In *DERA/RHUL Workshop on Secure Architectures and Information Flow*, 1999.

[11] L. Gong and P. Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Dependable Computing for Critical Applications 5*, pages 79–100, 1998.

[12] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Network and Distributed Systems Security Symposium*, pages 151–165, 1999.

[13] G. Lowe. Some new attacks upon security protocols. In 9^{th} *IEEE Computer Security Foundations Workshop*, pages 162–169, 1996.

[14] G. Lowe. A hierarchy of authentication specifications. In 10^{th} *IEEE Computer Security Foundations Workshop*, pages 31–44, 1997.

[15] W. Mao and K. G. Paterson. On the plausible deniability feature of Internet protocols. Published online, 2002.

[16] K. Matsuura and H. Imai. Protection of authenticated key-agreement protocol against a denial-of-service attack. In *International Symposium on Information Theory and Its Applications (ISITA)*, pages 466–470, 1998.

[17] K. Matsuura and H. Imai. Modification of internet key exchange resistant against denial-of-service. In *Pre-Proceedings of Internet Workshop (IWS 2000)*, pages 167–174, 2000.

[18] C. Meadows. A cost-based framework for analysis of denial of service in networks. *Journal of Computer Security*, 9(1/2):143–164, 2001.

[19] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS)*, pages 35–46, 2001.

[20] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[21] V. Ramachandran. Analyzing DoS-resistance of protocols using a cost-based framework. Technical Report DCS/TR-1239, Yale University, 2002.

[22] J. Smith, J. M. González Nieto, and C. Boyd. Modelling denial of service attacks on JFK with Meadows's cost-based framework. In 4^{th} *Australasian Information Security Workshop*, pages 125–134, 2006.

[23] D. Stebila and B. Ustaoglu. Towards denial-of-service-resilient key agreement protocols. In 14^{th} *Australasian Conference on Information Security and Privacy*, LNCS vol. 5594, pages 389–406. Springer, 2009.

[24] Y.-M. Tseng. Efficient authenticated key agreement protocols resistant to a denial-of-service attack. *International Journal of Network Management*, 15(3):193–202, 2005.

[25] M. Turuani. The CL-Atse protocol analyser. In 17^{th} *Int'l. Conference on Term Rewriting and Applications*, LNCS vol. 4098, pages 277–286. Springer, 2006.