Towards DoS-resistant Single Sign-On Systems

Valer Bocan, *Member, IEEE* and Mihai Făgădar-Cosma

***Abstract* — Single Sign-On systems are traditionally vulnerable to DoS attack since robustness was considered an afterthought at the time when most protocols were designed. This paper advocates the need to detect and counteract DoS attacks in SSO systems and presents a way to accomplish this through Bayesian inference and threshold puzzles.**

***Keywords* — Attack Detection, Denial of Service, Bayesian Inference, Single Sign-On.**

I. Introduction

MOST companies encounter problems related to network security when several applications request separate user authentication. The situation gets worse when the number of applications grows and specific authentication needs grow correspondingly. The need for connectivity pursued an increased dependency over user identity assurance, fact that has its own set of unique problems with respect to security and privacy.

Identity theft arises when an individual gets the credentials of another individual in order to fraudulently use them, typically for financial gain [1]. Evil doers steal wallets, purses and sometimes letters that contain financial or credit information. Modern thieves however try to steal information that can later be used to commit Internet fraud. A successful theft is a combination of the two methods, meaning that the information is stolen with its physical media and then it is augmented with data available from the Internet.

Identity theft cannot be avoided. There have always been fraud attempts and there will ever be. The measures that any society has to adopt regard securing online financial transactions and augmenting the weakest link in the chain from the client to the service provider (ISP). This link is more often than not represented by authentication, be it simple such as a mere password or advanced, such as the smart-card or biometrics.

With the proliferation of on-line services, the natural need of authenticating the users has arisen. One single individual is supposed to memorize tens of passwords, which is inefficient and unsafe. The users tend to neglect the risks associated with writing the passwords down or choosing a single password for all services. The solution is adopting Single Sign-On systems (SSO henceforth), which facilitates the automatic authentication of a user to each service that is being accessed, provided that the initial authentication between the user and the single sign-on system is successful.

Companies are usually affected by identity theft and they should be the first to adopt security mechanisms aided at limiting the magnitude of the phenomenon. Most companies that do not sell their services online authenticate their customers by checking physical tokens warranted by the government. Such tokens include credit cards, passports, ID cards, however the greatest drawback is that the mechanism is static (revocation is very difficult if at all possible) and mobile (in case of theft it can serve another person with minimal tampering).

SSO systems on the other hand, take advantage of the great flexibility of online communication, minimizing theft risks by using pseudonyms in the dialog between the service provider and the client. Pseudonyms are transparent to the user and they are limited to one service provider. Limiting the use of pseudonyms to one service provider increases the anonymity of such setup. In case the pseudonym is compromised for whatever reason, the change is immediate and transparent for the user, so we can speak about a dynamic mechanism which reacts quickly to changes.

II. DoS Attacks On SSO Systems

Single Sign-On (SSO) systems like the Liberty project [3], [4] are particularly vulnerable to DoS attacks, since they do not rely exclusively on technical means for authentication, but also on legal agreements between the involved parties. By taking advantage of the requirement that every message between the involved entities must be signed, a compromised service provider can flood the network, in particular the identity provider, with a burst of messages under false signature, which appear to be legitimate, as shown in Fig. 1. Since the receiving entity trusts the sender, it will check the signature of all incoming messages, a process which will quickly consume all its available resources. The result will be either a high latency in providing answers to the requests received from other legitimate service providers, or the complete crash of the service [5].

V. Bocan is a Ph.D. student with the Department of Computer Science and Engineering, "Politehnica" University of Timisoara, Romania (phone: +40 722 714798; e-mail: vbocan@dataman.ro).

M. Fagadar-Cosma is with the Digital Signal Processing Laboratories, Department of Computer Science and Engineering, "Politehnica" University of Timisoara, Romania (phone: +40 721 000499; e-mail: mfagadar@yahoo.com).

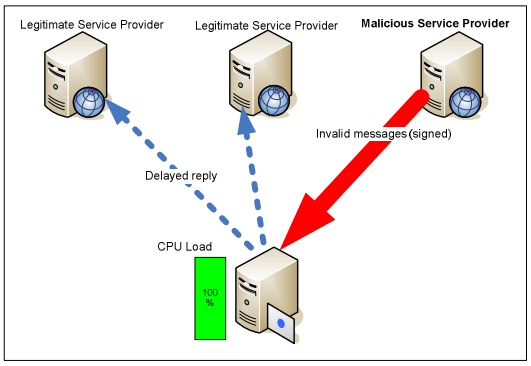

Fig. 1. A DoS attack aimed at a Liberty protocol

Digitally signing each request represents, by itself, a DoS attack. Although SSO systems are governed by legal agreements, this is often not enough to protect the legitimate users against attackers, therefore other measures, of technical nature, need to be taken.

The solution presented in this paper provides active measures of protection and attack detection for SSO systems, based on the analysis of the incoming network traffic parameters. These parameters can be the number of requests received in a certain time interval, the received packet size or the request duration, and are reported by a collection of sensors. The SSO-SENSE detection engine will apply Bayesian inference [6] on these parameters, in order to infer the current system state, identify the alleged DoS attacks and take appropriate measures to mitigate them.

III. Mathematical Foundations

A. The Bayesian inference theory

The Bayesian inference represents a method of reasoning in which probabilities are expressed not as frequencies or proportions, but as degrees of belief [7]. Bayesian inference involves collecting evidence which helps confirm or deny a certain hypothesis. Certainty is never reached, but as evidence accumulates, the degree of belief associated to a given hypothesis changes accordingly.

The mathematical theory behind the Bayesian inference mechanism relies on Bayes' theorem, which is used to compute the degree of belief of a certain hypothesis in the light on new information or evidence. Considering a set of mutually exclusive hypotheses, H_1, H_2, ..., H_N, and a new evidence E, which adds knowledge to the problem-solving process, we can compute the conditional (or *posterior*) probability of hypothesis H_1 for example, using Bayes' theorem, as follows:

$$P(H_1 \mid E) = \frac{P(E \mid H_1) \cdot P(H_1)}{\sum_{i=1}^{N} P(E \mid H_i) \cdot P(H_i)}. \quad (1)$$

The term $P(H_i)$ represents the *prior probability* of H_i, while $P(E \mid H_i)$ specifies the *conditional probability* of seeing the evidence E if hypothesis H_i is known to be true. In case multiple, statistically independent evidences E_1, E_2, ..., E_M, are used, they can be combined one by one using Bayes' theorem.

As it can bee seen from equation (1), Bayesian inference also requires *a priori* knowledge of the probability distribution of the hypotheses, in the form of an array of values: $P(H_1)$, $P(H_2)$, ..., $P(H_N)$ [8]. The hypotheses can be used to represent one of the possible states of a system, provided that they are mutually exclusive and complete. This way, probability $P(H_i)$ can be used to express the degree of belief that the system is in state H_i, if no other evidence has been obtained so far. As new evidence is obtained, the *posterior probability* will be used to adjust this degree of belief, for each of the available states.

Taking in consideration the presented aspects, we have decided to use Bayesian inference as a mean of determining the state of a SSO system, based on evidence provided by a set of software sensors [9], which are able to measure the parameters of the Liberty protocols [4]. This way, we can decide if the SSO system is in a normal state, or is currently the subject of a DoS attack.

B. Threshold Puzzles

Client puzzles [10]–[12] are an efficient way of preventing DoS attacks, by forcing the client to allocate its resources before the server whose services are being requested. This way, every moment during the execution of the authentication protocol, the execution cost at the client will be greater than that of the server. Before the server allocates resources for the connection, it will ask the client to solve a puzzle, which is essentially the brute-force inversion of a hash function like MD5 or SHA. Thus, the client is unable to guess the correct result, and has to solve the problem in order to find it. Only after receiving the correct result from the client, the server will start to allocate resources and service the connection.

The puzzle can be viewed as a pair $<N_S, k>$, where N_S represents a random number generated by the server, and k is the puzzle difficulty. In order to solve the puzzle, a client C generates a new random value, N_C. Using this value, the client is able to prevent an attacker to compute the puzzle before him and send the result to the server. Then, the client applies repeatedly a dispersion function to a quantity Y, and the puzzle is considered solved when the first k bits of this quantity, marked with X, are equal to 0, as seen in equation (2).

$$h(C, N_S, N_C, X) = Y. \quad (2)$$

The computational time required to solve the puzzle increases exponentially with the parameter k. When k is set to 0, no effort will be required to solve the puzzle. If, instead, it is set to 128 for MD5 or 192 for SHA, the puzzle will be unsolvable, as inverting an entire dispersion function is computationally impossible.

The server changes N_S periodically, and keeps a list of the used $N_S - N_C$ pairs, so that the old solutions cannot be reused. Still, if an attacker has a huge computational power, it may solve the puzzles very fast, and thus the DoS attack will not be stopped. In this case, the server will increase the puzzle difficulty, but this could prevent the legitimate clients from accessing the service.

Threshold puzzles [13] solve the above-mentioned problem by adding two new features to the traditional puzzles: an upper margin for the difficulty level, so that the clients will not be forced to solve impossible puzzles, and a minimum response time for the solution which prevents an attacker from solving the puzzle too quickly.

IV. Measurable Characteristics of the Liberty Protocols

This section details the measurable quantities in Liberty protocols that will be used as input for the Bayesian inference engine, which, in turn, will determine the degree of safety associated with the current state of the SSO system.

The measurable characteristics of Liberty protocols can be differentiated in three main categories. The first category named *SSOMessageContent*, is related to the protocol suite itself, whilst the second, *NetHealth*, contains network-related characteristics. The third, *SystemHealth,* refers to system parameters like the CPU load.

The first measurable characteristic contained in the *SSOMessageContent* category is related to the execution order of the protocols in the Liberty protocol suite. The protocol executions must follow an approximate order, and a break in this order could raise a question regarding a possible attack on the SSO system.

For example, a *Single Logout* request should not be received by a service or an identity provider if no *Single Sign-On and Federation* request has been previously received [5], as shown in Fig. 2.

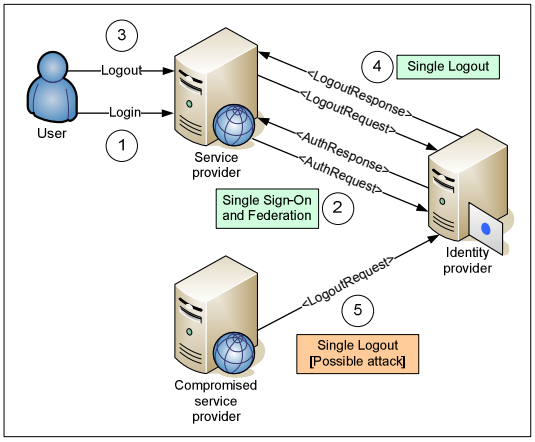

Fig. 2. Detecting possible attacks on SSO systems based on protocol execution order.

Another characteristic which could be used to detect attacks in the Liberty protocol suite is the content of the requests. For each protocol, the request has a set of attributes which specifies how it should be handled by the receiving entity. A large variation of these attributes in the requests received from a certain client could also indicate a possible attack against the server. For example, the <*AuthRequest*> message from the *Single Sign-on and Federation* protocol has an attribute which asks the identity provider to issue an anonymous, temporary identifier on behalf of the client, when exchanging data with service providers. Too many <*AuthRequest*> messages with this attribute set, received from the same client, could indicate a possible DoS attack on the identity provider.

The network-related characteristics are contained in the *NetHealth* category, and consist in parameters of the received network traffic [14], which have been already used with success in detecting DoS attacks [9], [15]. These parameters include the elapsed time between two successive requests, the request size, or the number of accesses per second received from a certain client. In the latter case, the client will be identified by his IP address, even if it may be forged.

The last category, *SystemHealth*, contains parameters related to the system itself, such as the CPU or memory load. As shown in Fig. 1, these parameters can be used to detect if the system resources are running dangerously low, which may be the possible cause of an attack targeted against the SSO system.

V. The SSO–Sense Detection Engine

Based on the Bayesian inference theory, we have designed the SSO-SENSE detection engine, oriented towards traffic monitoring and detection of DoS attacks in Single Sign-On systems. The proposed detection engine aims at providing better security for SSO systems, which are particularly vulnerable to DoS attacks since the employed protocols require every message to be signed.

The SSO-SENSE engine, illustrated in Fig. 3, infers the current state of the system for each connection, by gathering evidence from a set of sensors which are similar to the performance counters used by the Windows operating system [16]. Each characteristic of the Liberty protocol suite mentioned in section IV will be measured by a dedicated sensor, and the result will be used as input for the Bayesian inference mechanism. The output from the inference engine will be a new level of belief, named *AttackProbabilityLevel*, associated with the hypothesis that the system is under attack.

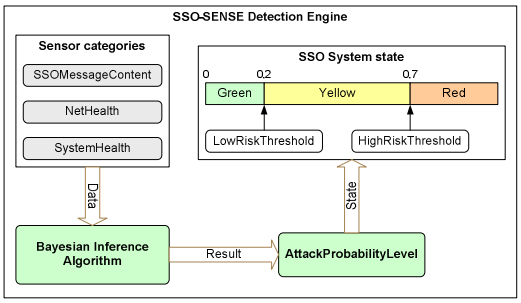

Fig. 3. The SSO-SENSE attack detection engine.

The *AttackProbabilityLevel*, is a floating-point number between 0 and 1. A value of 0 specifies that the system is operating safely, without being the subject of an attack, while a value closer to 1 specifies that the SSO system is under a heavy attack, which could lead to service failure. The *AttackProbabilityLevel* is used to decide what countermeasures to take in case of an attack, based on two defined threshold values: *LowRiskThreshold* and *HighRiskThreshold*. We propose the default values to be

0.2 and 0.7 respectively; however these may be tweaked at the time of actual implementation. We can identify three possible system states, as follows: the *Green state*, in which the system operates in relatively safe conditions, the *Yellow state*, when countermeasures are being deployed against the possible attacker, and the *Red state*, in which the attacker's requests are completely blocked by the server.

A. The Green system state

When the SSO system is in the *Green state*, the *AttackProbabilityLevel* is below the *LowRiskThreshold*. Therefore, we can conclude that the system is operating under normal, safe conditions. Taking any preventive action below this threshold could lead to a high number of false alarms [17], thus affecting the performance of the provided service.

B. The Yellow system state

The Yellow state is reached when the *AttackProbabilityLevel* takes values in the interval [*LowRiskThreshold*, *HighRiskThreshold*]. This is where the system begins to take countermeasures against what seems to be a possible attack. We have employed a novel technique, based on the threshold puzzle technology, in order to reduce and control the rate of requests coming from the eventual attacker.

In order to implement the SSO-SENSE detection engine, we intend to modify the Liberty protocol suite, ensuring that every <*AuthnRequest*> message from the client is followed by a <*PuzzleRequest*> message sent by the server [5]. After the client answers with a correct <*PuzzleResponse*> message, the server will commit its resources and begin checking the signature of the <*AuthnRequest*> message.

When the system is in *Yellow state*, the puzzle complexity will be proportional to the *AttackProbabilityLevel* value. If this value is higher, the puzzle complexity will be increased by the server, so that the client will require more time to solve it. Thus, while solving the puzzle, the client will be unable to flood the server with more requests, as they will not be taken in consideration.

C. The Red system state

The connection enters the critical *Red state* when the system is under a heavy attack, characterized by an associated *AttackProbabilityLevel* higher than the *HighRiskThreshold*. In this case, the server will forcibly close the connection, in order to save resources for the other well-behaved clients.

VI. Conclusions and Future Work

We have shown that single sign-on protocols, particularly Liberty, are vulnerable to denial of service attacks through resource starvation since they have not been designed to withstand such attacks. We propose a way to detect the attacks with a Bayesian engine called SSO-SENSE and a way to foil the attacks by engaging a traditional DoS countermeasure, in the form of threshold puzzles.

Since Bayesian inference is already being successfully used to fight spam, we believe that the proposed schema is efficient, albeit that is yet to be determined in practice. We plan to implement these theoretical considerations in software in order to prove that a DoS-resistant SSO system is able to withstand attacks and provide normal service to legitimate clients. Also, we intend to further refine the measurable characteristics of the Liberty protocols.

References

[1] Liberty Alliance Project, 2004, *Whitepaper on Liberty Protocol and Identity Theft*, Available: http://www.projectliberty.org.

[2] Computer Emergency Response Team, 2000, *CERT Advisory CA–2000.01 Denial of service developments*, Available: http://www.cert.org/advisories/CA-2000-01.html.

[3] Liberty Alliance Project, 2003, *Liberty ID-FF Architecture Overview*, Available: http://www.projectliberty.org.

[4] Liberty Alliance Project, 2003, *Liberty ID-FF Protocols and Schema Specification 1.2*, Available: http://www.projectliberty.org.

[5] V. Bocan, "Single Sign-On Systems under Denial of Service Attacks", PhD. report, Dept. of Computer Science and Eng., "Politehnica" Univ. of Timisoara, Timisoara, Romania, 2004.

[6] E. Charniak, "Bayesian Networks without Tears", *AI Magazine*, Winter 1991, pp. 50–63.

[7] Wikipedia Enciclopedia, 2004, "Bayesian inference", Available: http://en.wikipedia.org/wiki/Bayesian_inference.

[8] C. Siaterlis, B. Maglaris and P. Roris, "A novel approach for a Distributed Denial of Service Detection Engine", Dept. of Elect. and Computer Eng., National Technical Univ. of Athens, Athens, Greece, 2003.

[9] C. Siaterlis and B. Maglaris, "Towards Multisensor DataFusion for DoS Detection", in *SAC'04*, Nicosia, Cyprus, 2004.

[10] A. Juels and J. Brainard, "Client puzzles: A cryptographic defense against connection depletion attacks", in *Proc. of the NDSS*, 1999.

[11] D. Dean and A. Stubblefield, "Using Client Puzzles to Protect TSL", in *Proc. 10th USENIX Sec. Symposium*, 2001, Available: http://www.csl.sri.com/users/ddean/papers/usenix01b.pdf.

[12] V. Bocan, "Threshold Puzzles: The Evolution of DoS-resistant authentication", *Periodica Politehnica, Transactions on Automatic Control and Computer Science*, vol. 49, 2004.

[13] T. Aura, P. Nikander and J. Leiwo, "DOS-resistant authentication with client-puzzles", in *Proc. Cambridge Sec. Protocols Workshop 2000*, Cambridge, UK, 2000.

[14] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies", in *Proc 1st ACM SIGCOMM Internet Meas. Workshop*, New York, 2001, pp. 69–74.

[15] Y. Kim, W. C. Lau, M. C. Chuah and H. J. Chao, "PacketScore: Statistics-based Overload Control against Distributed Denial-of-Service Attacks", in *IEEE INFOCOM 2004*, Hong Kong, 2004.

[16] M. Groeger, February 2005, "An Introduction to Performance Counters", *The Code Project*, Available: http://www.codeproject.com/dotnet/perfcounter.asp.

[17] R. B. Blazek, H. Kim, B. Rozovskii and A. Tartakovski, "A novel approach to detection of "denial-of-service" attacks via adaptive sequential and batch-sequential change-point detection methods", in *Proc. 2001 IEEE Workshop on Information Assurance and Security*, West Point, 2001, pp. 220–226.