Dynamic Analysis of Security Protocols

Alec Yasinsac
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4530

Yasinsac@cs.fsu.edu

Abstract

Security protocols are essential to protecting electronic information, but security protocols are much more complex than many people think. Twenty years of research has failed to ensure security protocol effectiveness. We propose a new paradigm to protect secure enclaves. Using Dynamic Security Protocol Analysis, we will monitor executing security protocols and detect attacks in real-time by comparing ongoing activity to an accumulated knowledge base. The technique is founded on previous research in security protocol verification and on computer and network intrusion detection. Our thesis has several embedded research components. The following items summarize our challenges.

1. Define the methodology to identify malicious behavior
2. Gather distributed security protocol activity information
3. Accumulate the attack detection knowledge-base

Future protection of the Internet will rely on security protocols. We must use dynamic security protocol analysis to protect from attack by sophisticated intruders.

1. Introduction

Secure electronic communication relies on the application of cryptography. Cryptographic techniques are used to provide the confidentiality, integrity, non-repudiation, and authentication services necessary to exploit the capabilities of the Internet in the face of ever more sophisticated intruders. Even with perfect encryption, communication may be compromised without effective security protocols. Unfortunately, security protocols are known to be highly susceptible to subtle errors. To date, we have relied on formal methods to tell us if our security protocols are effective. These methods provide static evaluation that is largely dependent on the skill of the analyst. Further, they provide no complete or measurable level of security of the protocols they evaluate. As a result, security protocols are in operation that have known and unknown flaws. While static, off-line methods are incomplete, we are not aware of any methods or attempts to analyze security protocols on-line, dynamically. In this paper, we propose a new paradigm of on-line security protocol verification, Dynamic Analysis of Security Protocols.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
New Security Paradigm Workshop 9/00 Ballycotton, Co. Cork, Ireland
© 2001 ACM ISBN 1-58113-260-3/01/0002...$5.00

In the past year, the visibility of attacks has increased. Propagation of the *Melissa* and *ILOVEYOU* viruses along with the devastating Distributed Denial of Service Attacks (DDoS) has raised public awareness of problems that computer security experts have known about for years. The vulnerabilities that these attacks illustrate will not be easy to solve. In fact, in the February 15th issue of Crypto-Gram, Bruce Schneider suggests that there is no solution to DDoS attacks under the current Internet architecture. We contend that only the establishment of authenticated enclaves secured by cryptography can protect against such attacks by filtering traffic from unknown sources at enclave boundaries and by limiting the pool of effective *zombies* to the attacker. The cryptography that provides these enclaves will rely on effective security protocols.

Additionally, there has recently been an explosion in the growth of Public Key Infrastructure technology, where centralized or partially centralized services provide addresses and keys for participants desiring to establish secure channels with one another. These trusted services are worthy targets for intruders since successful intrusion would have wide-ranging impact. If a central trusted service can be compromised, it might be possible to use that service as an oracle, to compromise communications between any two participants using that service or to masquerade as any participant with virtually no trace. Such attacks are already manifesting [36], with little corresponding effort to address this threat in a systematic way.

Procter [43] confirms the rapid spread of encryption technology used to protect Internet packets. He points out that: "Encryption sleeves are becoming very common at all levels of the network, from virtual private networks (VPNs) to session encryption, such as secure shell (SSH) and application encryption such as secure socket layer (SSL)

The security of the information provided by encryption services is dependent on security protocols. Extensive work has been done to test [22] and verify [16] security protocols, and significant progress has been made in these areas. Nonetheless, no method provides complete, or even measurable, confidence in security protocols. In fact, based on the nature of security protocols and their environment, it may be impossible to accurately predict their performance through formal analysis or automated testing. In [15] the

authors show how attacks can be constructed through interaction of two simultaneously executing protocols, even though both protocols are "secure" when run independently.

The work on verification of security protocols has been largely theoretical, since cryptographic systems are few in comparison to the overall scope of the Internet. Conversely, Intrusion Detection System (IDS) research has been highly successful in meeting practical, low assurance security requirements in an Internet environment [4, 7, 11, 18, 35]. IDS techniques focus on characteristics of "normal" network traffic, and normal user behavior as identified through network and host activities. Historical data is tracked and modeled by statistical measures providing a baseline to compare new activities against. Two examples of the data considered by IDSs are:

1. The commands a user routinely issues to a host

2. The type of traffic generated by an application on the network (identified by the Internet Protocol (IP) port).

More recently, efforts have been made to extend intrusion detection techniques to a slightly different environment. In [11], Jou, Gong, et al. show how network protocol traffic can be analyzed to protect the network routing infrastructure.

In the same way, we believe that established intrusion detection techniques apply to active analysis of the environment that surrounds trusted services in high assurance systems. In this paper, we provide a framework of how IDS technology can be applied to the security protocol environment, and extend this work by categorizing attacks in order to determine appropriate responses to detected anomalies and intrusions. While there has been intense research in security protocol analysis and verification and equal emphasis on research in intrusion detection, the two fields have not been considered together. We suggest that combining these technologies can create a tool that will automatically detect attacks on trusted security services through identification of anomalies in the security protocol traffic. This technology can also characterize potential intrusions in order to suggest a proper action to take in response to the noted anomaly.

The cumulative result is that this research will provide a mechanism for active defense and response to attack for security services. The mechanism will ensure reliable, effective performance of critical security services and will prevent sophisticated attackers from utilizing or masquerading as these security services.

2. Security Protocol Verification

Security protocols are intended to provide secure services. Most often, these services entail establishing a secure channel between two communicating principals. Unfortunately, security protocols are subject to flaws that are not easy to detect. If the protocols underlying the secure channel are flawed, then the security objectives of the participants are undermined, possibly at great financial, physical, or other risk to the participants. [26] first highlighted vulnerabilities of security protocols, and a mountain of research amassed since that time aimed at ensuring the effectiveness of these security essentials.

Protocol verification attempts have generally fallen into one of four categories:

1. Testing tools designed to reduce the search space of possible errors [22]

2. Epistemic logics [2]

3. Theorem proving techniques [12, 33, 28, 24]

4. Symbolic Model Checkers [29,30]

Other, less mature paradigms have emerged based on formal semantics [39], and Brackin has been successful in finding flaws by combining application of security protocol design principles with logical protocol analysis [3]. Bellare and Rogaway proposed provably secure protocols in [42]. Still, each of these techniques are static in the sense that they are performed on protocols in a laboratory environment in symbolic execution only and consider theoretic vulnerabilities based largely on the form of the messages. The results are not based on actual protocol execution, but are founded on a mathematical representation (model) of the protocols used (mathematically) to prove security properties.

Protocol verification results to date are mixed. On the positive side, many protocol vulnerabilities were (and still are being) identified that likely would not have been found without formal methods. In fact, the literature is filled with uncounted errors detected in proposed, production, and contrived protocols. So many attacks were found that they were categorized in 1994 when Syverson produced a taxonomy of protocol attacks [32]. The categories of attacks, and corresponding protocol vulnerabilities, range from misuse of encryption (signing encrypted messages) to dangerous message formatting (components in one message are not clearly distinguishable from components in another message of the same protocol).

Interestingly, clever intruders that utilize multiple concurrent sessions exploit most of the detected vulnerabilities, often when the protocol was written (intentionally or not) to be run independent of any other sessions[1]. As a result, much of the existing security protocol verification research reflects this paradigm, where a protocol is shown to be resistant (or vulnerable) to attacks accomplished by an intruder that simultaneously engages one or more hosts in multiple sessions of the [single] protocol to be verified. Tjaden [34] and Kelsey et al. [15] address the more realistic environment where each principal is executing multiple sessions of many varying protocols. The challenges to static protocol verification in a multi-user, multi-protocol environment are even more daunting.

---

[1] In [19], Lowe demonstrates an attack on a famous protocol by using that protocol in a subtly different context than it was originally intended for [41].

The successes of verification techniques in finding errors notwithstanding, none of these methods have been shown to be effective in showing measurable security in any protocol or class of protocols. So, while no protocols devised to date are proven secure, there is no evidence in the literature that anyone has considered dynamic analysis of real protocol traffic to detect compromise or attack. That is the focus of this paper.

3. The Operating Model

3.1. Definitions.

We consider the vulnerabilities to centralized security services to occur through protocol sessions that provide system-wide secure channels and that shield these channels from attacks from lower layer attacks. Terminology applicable to secure channels is often confused with that associated with normal network traffic. Here, we present a few definitions to distinguish our discussion of messages, sessions, etc.

Principals are participants in a protocol session, distinguishable by a unique identifier. Our focus is on special principals that we refer to as *security servers*, *trusted servers*, and at times *Key Distribution Centers*. The common characteristic of each of these three terms is that they represent principals that provide a security service to other principals, and are, thus, integral components of the security infrastructure, with significant impact on the security for those principals that they serve.

The messages that we are interested in are only those used in a protocol session. Anonymous messages, network overhead, and other traffic not associated with security protocols for trusted services are not messages in our sense. To distinguish the traffic of interest to us, we define a "message" as a tuple of at least four elements:

1. The identifier of a principal representing the source (originator) of the message
2. The identifier of a principal representing the destination (recipient) of the message
3. The message payload which may be a structured element comprised of one or more data elements to be conveyed from the source to the destination.
4. A protocol identifier

A *protocol* is a fixed sequence of messages predefined to principals that either originate or receive the message(s). Each principal may recognize and utilize multiple protocols. A principal recognizes a protocol if the protocol is stored in the principal's private memory. Principal Alice utilizes protocol P if one protocol session exists or existed where Alice is either an originator or recipient of a message with the protocol identification field of P.

A *protocol session* is an instantiation of a protocol. Thus, it is a set of messages that correspond to the form of a protocol, where the generic source and destination identifiers are replaced with actual identifiers and an actual payload replaces the generic payload. Notice that every message that meets our definition is a message of a protocol session. We recognize that there will be many non-protocol transmissions on any network, but for our purposes, we ignore transmissions that are not messages by our definition.

3.2. Traces.

It is normal practice to specify protocols as an execution trace [38] of actions between principals, with each principal taking turns. These protocols are listed as though the messages are executed sequentially, on a single processor, when they are intended for concurrent execution, in a distributed environment. In practice, any principal may be executing multiple protocols concurrently. In this case, a *trace* is the set of all messages executed by a principal. A trace may be thought of as an interleaving of protocol steps as described in [32] and [13], meaning the execution of the steps of two different protocol sessions are intermixed.

It is important to distinguish between the symbolic execution of a protocol and an actual execution. For existing methods of protocol verification, symbolic protocol execution is examined. That is, protocols are encoded with generic values and readable, symbolic identifiers to facilitate reasoning about the results of the protocols. It is this symbolic version of the protocol that is recognized (as defined above) by principals. Conversely, when protocols are executing, they contain actual data that are not routinely readable to a human. For example, random numbers are largely unrecognizable to the human eye. Fortunately, protocols are executed on computers that can recognize random numbers and other protocol components, and can match the actual messages to their symbolic counterparts, even when executing multiple, complex protocols in a highly concurrent environment.

A protocol trace is the accumulation of actual messages sent and received by a principal in the order that they were sent and received. For a protocol session running alone, the trace is simply the listing of the messages in the protocol. If multiple protocols (or multiple instances of the same protocol) are executing concurrently, the trace will be extensively interleaved. This is a common occurrence in networking; where any large host may be concurrently executing requests from many different sessions for logon, file access, computations, etc. It is this type of interleaved trace that present intrusion detection technology targets.

3.3. The Man-in-the-Middle.

The Man in the Middle model has been around for a long time. In [8] and later [1], the authors formally define such an environment, where every message in the communication system must pass through a powerful intruder. We target this model and the extensions from [37], because it is a powerful model, and it is easy to understand. Some of the characteristics of this model are that:

- All messages to/from every principal (including the KDC) are to/from a single party (i.e. the intruder)
- Principals operate in their own "address space". Personal memory changes only by receiving new

information through the network, or by performing computation on information that is already stored. This personal memory represents the local state of the principal.

- The only knowledge that a principal can acquire about other principals is through the network.

3.4. Challenges of Man in the Middle.

It is clear that assuming such a powerful adversary presents challenges to our ability to detect and respond to intrusions. Among the consequences of our choosing the model are the following specifics.

a. The Intruder knows much more than we do. Unless we make assumptions about the underlying communications [e.g. that we use a broadcast medium] or inject a distributed information gathering mechanism such as roving agents, we cannot see all messages. In fact, we assume to see only a very limited subset of the communications that may be used to generate an attack against us, to the extent that a KDC can rarely expect to see all (or even most) messages exchanged in a protocol. On the other hand, the intruder is assumed to see (and have immediate access) to ALL messages on the network.

b. We are limited by realistic resource assumptions. In order to provide useful information about intrusions, even more critical for response, we must consider response time as a limiting factor for our methods. Conversely, because we assume the intruder is very powerful, they may conduct resource intense activities, such as gathering information over a long period and conducting fast searches over extensive databases in real time.

Assuming the strongest model is essential when addressing the effectiveness of security protocols. Applications using this environment will have a strong assumption of security and, thus, will pass more sensitive and/or valuable information. Additionally, unlike existing network services, because of the necessarily centralized nature of security services, the impact of a single intrusion can have broad scope, compromising a large volume of transactions. The ability to detect these attacks, hopefully before they occur, and to reduce the impact if they are successful is essential to network security, recovery, and deterrence.

4. Intrusion Detection

Intrusion Detection is a fertile research area since the mid-1980's and it continues to be a topic of intense research [7], [35], [17]. A principal technique used in intrusion detection is profiling, detailed in Denning's seminal paper [5]. We will show how monitoring activity to detect and respond to attacks mirrors the environment that Denning addresses.

The problem is deeper than simply detecting all attacks, since we could meet this challenge by simply signaling "possible attack" for every protocol action taken. Of course this would provide no useful functionality, since we could not take effective action based on that feedback. With that in mind, we recognize two measures of our effectiveness as false negatives and false positives.

a. Do we detect all attacks (false negative)? If not, what percent of attacks will we detect? We call this metric: *percentage of attacks detected.* While we can empirically analyze our system utilizing this metric, it is only useful in a laboratory environment, since in an actual environment we cannot know how many attacks that we do not detect.

b. Do we incorrectly detect activities that are not attacks (false positives)? If so, what percent of activities will produce false alarms? This metric is termed: *ratio of false alarms to activities*. This important metric can be analyzed in a laboratory and in production use. We must have the goal of keeping this metric as low as possible to ensure that appropriate actions are taken when attacks are detected.

4.1. [2]Behavior as an Attack Indicator.

Intrusion detection is focused on the behavior of communicating principals. The assumption is that, while it may be disguised, a principal's behavior will reflect their intentions. There are two fundamental behaviors that are used to identify potential intrusions:

a. That which was previously shown to result in compromise

b. That which significantly deviates from the norm

4.2. Signatures.

In the first category we include characteristics of known attacks on cryptographic protocols as well as intuitively dangerous behavior. These attacks may be characterized by sequences of activity traces, similar to methods for virus scanning and for network intrusion detection [4]. The pattern of these sequences produces a *signature* for the known attack. Traces that match these signatures are always suspect, and in some cases may be enough evidence to affect a protective or damage control response in and of themselves, with no corroboration necessary. An example, given in the same reference is that any program that sets UID during execution should be flagged as a high risk. Another example of an activity pattern that is always suspect given by Denning [5] is a high rate of password failures by any user.

Since the famous attack on the Needham and Schroeder protocol in [6], uncounted attacks have been documented on contrived and production protocols. Syverson produced a taxonomy of protocol attacks [32] that may allow abstract construction of signatures for intrusion detection through protocols in much the way that Spafford's taxonomy [14] of

[2] Throughout this paper, we use the terms *behavior* and *activity* almost interchangeably. Behavior is used to reflect observed activity, as well as some intention that preceded the action. Most often, behavior refers to activities that are abnormal rather to that which is inherently dangerous.

network intrusions provides a framework for identifying signatures for network attacks in IDIOT.

Examples of dangerous behavior in a security protocol environment include:

- Simultaneous triangular sessions (A->B, B->C & C->A).
- Sequential triangular sessions (A->B, B->C & C->A)
- Request to encrypt with a public key followed by a request to sign with the same key.
- Simultaneous Group Protocols
- Failed protocol sessions
- Suspended or partially completed protocol sessions
- Repetitive use of one cryptographic key

4.3. Profiles.

Profiling essentially means recording observed activity of a principal over time and producing a data structure that reflects normal activity of that principal. This data structure is called a profile. The fields in the profile contain data that models the activity in some predefined way. We will select models that allow us to accomplish the two goals that we just laid out, of detecting a high percentage of attacks and of producing a low percentage of false alarms.

4.3. Profiling for Abnormal Behavior.

The second category of behavior that we are interested in is anomalous activities. If we assume that intrusions are not routinely accomplished, then it is reasonable to infer that abnormal behavior is more likely to be an intrusion than is normal behavior. Denning states it this way in [5]:

> The model [for the use of profiling] is based on the hypothesis that exploitation of a system's vulnerabilities involves abnormal use of the system; therefore, security violations could be detected from abnormal patterns of system usage.

If we accept this premise, then we can reduce our problem of detecting intrusions to one of detecting abnormal behavior. First, we must categorize behavior in order to be able to distinguish that which is normal from abnormal. For a shared computer environment, Denning categorizes behavior based on activities on objects, where the objects are resources ("...files, programs, messages, records, terminals, printers, and user- or program-created structures."[5]).

Our view of intrusions is based more exclusively on activities; specifically, activities carried out through security protocols. We expect that, after sufficient data has been gathered to reduce the impact skew and after usage has stabilized after initial system usage, normal behavior will be recorded. Thus, we can characterize the behavior of each principal (Alice) based on measurable criteria such as:

4.3.1. Which protocols has Alice utilized by originating the first message? A legitimate principal (or an intruder that has compromised a legitimate principal) that initiates a protocol that they do not normally use may indicate an attempt to generate data that may allow an attack.

4.3.2. Which protocols has Alice utilized as recipient of the first message? A legitimate principal (Alice) that receives an unusual request for service may indicate that the originator is making an attempt to utilize Alice as an oracle.

4.3.3. How frequently does Alice utilize each protocol as originator of the first message? An increase in frequency of use of a protocol could reflect an attempt to generate a value in a data field necessary to construct an attack.

4.3.4. How frequently does Alice utilize each protocol as recipient of the first message? A legitimate principal (Alice) that receives an unusual number of requests for service may indicate that the originator is making an attempt to utilize Alice as an oracle.

4.3.5. What other principals are normal recipients for each protocol that Alice utilizes and where she originates the first message? A sudden change in the targets of requests for service by Alice may indicate that Alice has been compromised and is now being used to gather information for an attack.

4.3.6. What other principals are normal originators for each protocol that Alice utilizes and where she is a recipient of the first message. A sudden change in the sources of requests for service from Alice may indicate that another principal has been compromised and is now being used to gather information for an attack.

4.3.7. In what order does Alice utilize protocols where she is the first message originator? Multiple sudden changes in the order of requests for service from Alice may indicate that Alice is making an attempt to gather information for an attack.

4.3.8. In what order does Alice utilize protocols where she is the first message recipient? Multiple sudden changes in the order of requests for service to Alice may indicate that another principal is making an attempt to gather information for an attack.

4.3.9. How often does Alice exercise an encryption followed by signature? While signing an encrypted message is considered a vulnerable activity, a principal may, after careful consideration, conduct certain ordered activities without concern. A change in this ordering pattern may indicate that an attack is ongoing.

4.4. Trace profiles.

In our earlier definition of traces, we referred to the tendency of protocol analysis to focus on the symbolic execution. We again consider the symbolic execution of protocols, not as derived from a preconceived or contrived execution for test purposes, but the symbolic representation of messages

executed in a production protocol environment. Rather than constructing the interleavings from the protocols, we reconstruct the symbolic trace from the execution trace of the protocols as they occur in the system.

Symbolic trace information may be extracted from state data maintained by the host representing each principal as the protocols are executed, or a sophisticated listener monitoring communications on the network may infer it. Since we are concerned with activities that correspond to a trusted principal, we can assume that the necessary state information will be available to translate actual messages into symbolic form in real time.[3]

Because we can recover the symbolic representation of protocols as they execute, we can construct profiles of protocol usage base on their symbolic characteristics. For example, we can record the symbolic representation of every protocol that executes on the monitored computer and record statistical information about the sessions, and about each message. We can determine which protocol that the monitored principal participates in. We can determine who the monitored principal communicates with and can gather statistics regarding the time and sequencing of application of these protocols. These statistics can be translated into the model information discussed in the next paragraph.

4.5. Statistical Models.

The metrics described in paragraph 4.1 can be represented in statistical models that Denning describes [5, pp122-3]. For example, the metrics described in paragraphs 4.3.4 and 5 can be analyzed using the operational model, mean, and standard deviation as given by Denning in par 5.2.1 and 2 and by the multivariate model from Denning's 5.2.1.3. The Markov Process Model as given in Denning's 5.2.4 can measure the metrics we describe in paragraphs 4.3.1, 2, 5, and 6. The Time Series Model Denning presents in paragraph 5.2.5 are applicable to the events we describe in paragraphs 4.3.3, 4.3.4, 4.3.7, 4.3.8, and 4.3.9.

Such modeling will serve to improve both the percentage of attacks detected and the ration of alarms to activities, similar to the results of intrusion detection systems.

A sample profile for measuring Trent's activity may include a three dimensional array, where one dimension represents each protocol that Trent recognizes, another represents each other principal that Trent communicates with, and the third distinguishes whether Trent was the originator or recipient of the first message.

5. Categorization of Attacks.

5.1. Taxonomies of attacks.

[3] We consider evaluation of actual protocol messages and the explicit program actions that result to be an uncharted research arena, with roots in classic intrusion detection methodology. We leave that discussion for another time.

We now turn from our focus from detecting attacks on trusted principals to categorizing attacks for the purpose of formulating appropriate responses. This is a classic problem in intrusion detection, including deciding what to do when an attack is detected, and what to do when behavior is encountered that is neither categorized as an attack, nor as normal behavior. We earlier pointed out taxonomies for intrusions into computer systems [14] and for attacks against protocols [32]. The latter is of particular importance to us, and assists us in categorizing behavior for selecting an appropriate response.

Syverson partitions attacks against protocols into two major categories of external and internal attacks. These are further decomposed into categories and sub-categories. These categories and responses follow:

- Interleaving attacks (including replays) requiring contemporaneous execution of more than one protocol. An appropriate response to detection of contemporaneous execution of two protocols that are vulnerable to such an attack would be to suspend or cancel one session or the other.
- Replay attacks that need not require contemporaneous execution of more than one protocol. The proper response in this case would be dependent on the state of the principals involved when the attack is detected. If the attack is detected during execution of the reference session, keys may be updated, certifications revoked, and existing sessions may be aborted. If the attack is detected during the attack session, the attack session would be aborted.
- Message deflection attacks. If message deflection is detected, there are two responses required. First, the principal that was the intended destination for the deflected message must be notified and damage control actions taken. Second, the principal that received the message should be notified and the protocol session, if it is still active, aborted.
- Message reflection attacks. The impact of message reflection is centralized to one principal that is the originator and recipient of the message. Again, the proper response depends on the timing of the detection. If the attack session is still under way, it should be aborted. If the attack session has ended, the victim should be notified of the details of the attack and should initiate local damage control activities.

5.2. Other categorizations of attacks. The above taxonomy provides a comprehensive view of protocol vulnerabilities from the perspective of interleavings of messages. We take another perspective of these vulnerabilities to consider the intent of attackers and discuss responses related to these intentions. Once an attack is detected, at least three goals must be considered when constructing a response:

a. Assess and correct the damage of the compromise.

b. Prevent further compromise.

c. Catch and prosecute the perpetrator.

In the following discussion, we consider Alice and Bob to be uncompromised principals, Trent is a principal that provides trusted services, and Mallory is a malicious attacker.

- Compromise secrecy. This is the canonical attack. Alice and Trent need to share information privately. Mallory wants to know the information and constructs an attack that will divulge the message meaning to her. Terminating the session, changing session or key exchange keys, identifying and gathering appropriate log files may be appropriate responses.
- Compromise integrity. Mallory may attack the system in order to provide inaccurate information to Alice or Trent. When such an attack is detected, data from the attacked session should be validated. Audits from previous sessions should be held.
- Compromise nonrepudiation. If Mallory can sign messages as if she were Alice, then she can incorrectly attribute actions or information to Alice. Detection of such an attack should result in correction of any signatures generated during the attack session and should initiate review of records of previous transitions involving signature by Alice. Depending on the nature of the attack, long term key change may be in order.
- Compromise availability. Mallory may desire to prevent Alice from receiving one or more messages while preventing Alice from recognizing that the message(s) have been delayed or destroyed undelivered. The appropriate response to a denial of service attack is to restore the secure channel and notify other principals of the loss of service so that any lost transmissions may be recreated.
- Attempt to masquerade as Alice to Bob. If Mallory can convince Bob that she is Alice, she can compromise secrecy, integrity, and nonrepudiation between Alice and Bob. Response to detection of such an attack is dependent on its success. If the masquerade has been successful, affected participants should be notified and long-term keys changed. At a higher level, the nature of the attack should be evaluated and the security vulnerability removed. Participants should be notified of the vulnerability until it is resolved.
- Attempt to masquerade as Alice to Trent. If Mallory can convince Trent that she is Alice, she can compromise secrecy, integrity, and nonrepudiation between Alice and all other principals.
- Attempt to masquerade as Trent. If Mallory can convince all other principals that she is Trent, then Mallory can compromise secrecy, integrity, and nonrepudiation between all principals. Because of the widespread ramifications, these are the most dangerous masquerade attacks. Response to an attempt to masquerade as a trusted service must first focus on controlling the damage.
- Attempt to use Alice as an oracle. If Mallory can devise a general method of utilizing Alice as an oracle, then Mallory can masquerade as Alice to any other principal, compromising secrecy and nonrepudiation.
- Attempt to use Trent as an oracle. If Mallory can devise a general method of utilizing Trent as an oracle, then Mallory can masquerade as Trent.

6. Protocol-oriented, State-based Attack Recognition

To date, the only known provably secure protocols are zero knowledge protocols [9] that do not generally have broad applicability. Since we expect any protocol that protects trusted services has flaws (that are either known or unknown) we select a protocol with a known flaw to illustrate how an attack on a cryptographic protocol can be dynamically detected using intrusion detection-like technology. The protocol we present is the canonical Needham and Schroeder Public Key Protocol [26] with the public-key acquisition steps omitted, given in Figure 1a, with a known attack given in Figure 1b. This is a parallel session attack and an insider attack, because it is only possible for an intruder (Mallory) to accomplish the attack when a principal (Alice) begins a session with Mallory, where Mallory is known and trusted by Alice. It also requires that Alice and Mallory know some third party (Bob) that trusts both Alice and Mallory.

As we noted earlier, the concurrent execution of security protocols by a group of principals can be represented by a serialized, interleaved execution trace of the steps in each protocol session. This trace provides the environment for our dynamic protocol analysis. For this illustration, we show how we might dynamically detect the attack on the Needham and Schroeder Public Key Protocol (NSPKP) by identifying its signature. We recognize that NSPKP has many signatures, but for the purpose of simplicity, we start by focusing on the source and destination of the messages. The signature that we will recognize is given in Figure 2.

Needham and Schroeder Public Key Protocol

1. A -> B: $[n_a]PK_b$
2. B ->A: $[n_a,n_b]PK_a$
3. A -> B: $[n_b]PK_b$

Figure 1a

1. A -> M: $[n_a]PK_m$
1'. M_a -> B$[n_a,A]PK_b$
2'. B -> A: $[n_a,n_b]PK_a$
2. M -> A: $[n_a,n_b]PK_a$
3. A -> M: $[n_b]PK_m$
3'. M -> B: $[n_b]PK_b$

Figure 1 b

We model the behavior of this trace with the state transition machine given in Figure 3. The machine represents a protocol signature by mapping the sender/receiver pair for a message into one of the possible segments of a trace. We assume valid participants are members of some population P. In our illustration, A, B, M $\in$ P. For example, from the start state, the machine will transition into S_1 when a valid principal begins a session with another valid principal. When the originating principal sends a message to another

(different) principal, the machine transitions into S_2. When the recipient of the second message sends a message to the originator of the first message, the machine transitions into S_3 and so on.

In the simplest case, where the steps are executed in exactly the sequence given in Figure 3, without any other activities in between, this model will detect an attempt by Mallory to defeat the NSPKP. You may notice that recognition of the execution of messages in this sequence does not *guarantee* that an attack has occurred. Fortunately, dynamic attack detection does not rely on guarantees. Beaconing on suspicious behavior is often sufficient to provide the necessary level of protection. Moreover, we humbly leave the issue of false positives for our future work section.

There are many positive qualities of this simple model. One may consider that transition through the table reflects the likelihood that an attack is under way. When the machine is in the Start state (there are no active messages that meet the format), there is no chance that an attack is under way. When the first recognized message is received, it is still unlikely that it is the first message of an attack sequence, but it is more likely than when no messages were being processed. A more significant jump in likelihood may occur when the second message meeting the format is detected. Similarly, as the machine reaches larger state numbers, the likelihood that an attack is underway increases. This monotonic property may be used to signal probability of attack to a system monitor or combined with other sensors to give a network threat picture.

A -> M
M_a -> B
B -> A
B -> A
A -> M
M_a -> B

Figure 2

	Transition	Old State	New State
1	A -> M	Start	S_1
2	A[4] -> B	S_1	S_2
3	B -> A	S_2	S_3
4	M -> A	S_3	S_4
5	A -> M	S_4	S_5
6	A -> B	S_6	Beacon

Figure 3

While this simple model has some relevant semantic meaning, it is also obvious that it also has problems. For example, it is possible that, under this model, an actual attack sequence may be masked by legitimate sequences. This is illustrated by the case where after starting a session with Mallory, Alice starts a legitimate session with Bob (or Debbie, etc.), followed by Mallory's masquerade message as Alice. The machine would be out of synchronization with the actual attack, masked by Alice's valid message. We could spawn a new machine every time such conditions occur. It is also possible to remove this hindrance by slightly extending the model.

The root of that problem with our simple illustration was that the signature does not contain sufficient specificity. We can increase the specificity of our recognizer by keeping track of the protocol session for each message. This allows us to prevent the masking we noted above in a systematic way by spawning only a single new machine when we recognize that a possible third session in the proper format exists, as shown in Figure 4.

We illustrated this new paradigm with a very simple example, then extended the example slightly to show the power that can be attained with only a little additional information. We do not believe that the two illustrations exhaust the utility of information available to a dynamic protocol analysis mechanism. In the illustrations given here, we do not consider any information passed as the payload of the message, and note that it is on payload information that traditional protocol verification base their entire analysis. While getting at that information in a dynamic environment will be more than a trivial challenge, its potential is evident.

	Message	Session	Old State	New State
	A -> M	1	Start	S_1
	A -> B	2	S_1	S_2
	A -> C	3	S_2	Spawn
	B -> A	2	S_2	S_3
	M -> A	1	S_3	S_4
	A -> M	1	S_4	S_5
	A -> B	2	S_6	Beacon

Figure 4

7. Future Work

This effort is in its early stages. While experimental data is not available to support this thesis, the theoretical foundations are rock solid. The ability to analyze execution traces in real time was shown for operating system and network intrusion detection systems [4, 18, 35, 7]. The ability to describe existing attack is shown in Syverson's taxonomy [32], and the ability to detect previously unknown attacks using profiling techniques described by Denning [5] has been proven [18].

Many challenges lay ahead. We will first tackle the problem that we illustrate with the example in Section 6 of detecting

[4] We model this step as originating from Alice because the intruder is considered all powerful and can, thus, pose as any other valid principal as far as sending and receiving messages is concerned. We also address issue of recording this information for distributed principals in the "Future Work" section.

known attacks using a signature-based approach. In order to fully implement this paradigm, we must accomplish reliable remote protocol execution tracing capability. Many are addressing how to gather distributed information through agent-based mechanisms [27] and mobile code [10]. Others are focused on protecting remote agents [31]. We will first consider an integrated environment where principals cooperate so the necessary information may be easily gathered. We give a detailed description of this environment in [40]. We also intend to consider broadcast-only transmission environments that will become more prominent for secure enclaves as wireless technology advances. Finally, there is a clear application for intelligent, mobile agents to gather information that would complement other information sources.

We have addressed many issues in this research direction in this paper in detail, and have addressed several others in general terms. We recognize that there are many open questions (theoretical and practical) that we have not addressed that must be resolved before dynamic analysis of security protocols becomes reality. Among them:

- How will we protect the detection system itself?
- How will we prevent tool misuse?
- How will we systematically formulate appropriate responses for the numerous different behavior categories?
- Will this IDS work within performance and storage constraints?

We have already begun to establish the laboratory tools necessary to test and implement this mechanism. We are producing a compiler that will translate protocols specified in a language created for protocol verification into executable modules that we can implement on closed network security laboratory.

We are further encouraged in this initiative by the allowable imprecision of the result. Our goal will not be 100% detection of assaults on security protocols with zero false positives. Rather, we will endeavor to produce a mechanism that meets or exceeds the qualitative standard set in recent intrusion detection system evaluations [18].

Finally, we believe that dynamic, on-line analysis of security protocols will provide a foundation for providing complete network situational awareness and active response to attacks and suspicious activity. As we mention in Section 6, this technique can recognize when a complete attack sequence has occurred, or when a partial attack sequence has occurred. We believe that this characteristic will allow appropriate responses to be systematically constructed for each global state. Responses will be generated based on factors such as the complexity, likelihood, scope, potential damage, etc. of the attack as well as other aspects of situational awareness (e.g. the current network threat posture).

8. Conclusion

We present a method to provide active defense for distributed security services. We have shown how proven intrusion detection technology combined with knowledge gained by formal analysis of security protocols can be applied to this problem. Our method involves addressing behavior relative to protocol activation and use rather than considering activities against objects, **s** is conducted in classic intrusion detection. We categorized protocol-based attacks and showed how this method can be used to increase situational awareness and threat picture by considering classes and characteristics of attacks and of suspicious behavior.

Until recently, the requirement for trusted services essentially resided with the federal government and a few large corporations, where key exchange was most often carried out by courier, with the key material stored on paper tape or diskette. Present technology demands extension of the protection provided by cryptography. This necessitates extension of key distribution and, thus, authentication services. These centralized services are attractive targets for sophisticated intruders. The method we prescribe offers to protect this vital link to our security infrastructure.

9. Acknowledgements.

Many thanks to Paul Syverson and Chenxi Wang for their insights on an early version of this paper, and to the anonymous referees and workshop participants, whose comments were very helpful.

10. Bibliography

[1] Martin Abadi and Mark R. Tuttle, "A Semantics for a Logic of Authentication", *Tenth Annual ACM Symp on Princ of Dist Computing*, Montreal, Canada, August, 1991

[2] Burrows, M., Abadi, M., and Needham, R. M. "A Practical Study in Belief and Action", In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning about Knowledge* (Asilomar, Ca., Feb. 1988) M. Vardi, Ed. Morgan Kaufmann, Los Altos, Calif., 1988, pp. 325-342

[3] S. Brackin, "Automatically Detecting Most Vulnerabilities in Cryptographic Protocols", in The *DARPA Information Survivability Conference and Exposition*, January 2000, Vol.1, pp 222-36

[4] Crosbie, M.; Dole, B.; Ellis, T.; Krsul, I.; Spafford, E, "IDIOT - Users Guide", Technical Report TR-96-050, Purdue University, COAST Laboratory, Sept. 1996

[5] Dorothy E. Denning, "An Intrusion-Detection Model", From *1986 IEEE Computer Society Symposium on Research in Security and Privacy*, pp 118-31

[6] D. E. Denning and G. M. Sacco, "Timestamps in key distribution protocols," *Communications of the ACM*, vol. 24, no. 8, Aug 1981, pp. 533-536

[7] Daniels and Spafford, "Identification of Host Audit Data to Detect Attacks on Low-level IP", *Journal of Computer Security*, Volume 7, Issue 1, 1999

[8] Dolev, D., and Yao, A.C. "On the security of public key protocols". *IEEE Trans. Inf. Theory* IT-29, 2(Mar. 1983), pp. 198-208. Also Stan-CS-81-854, May 1981, Stanford U.

[9] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", *Advances in Cryptology*, Proc. of Crypto~'86 (Lecture Notes in Computer Science 263), Editor A. Odlyzko, Springer-Verlag, pp. 186--194, Santa Barbara, California, U. S. A., August 11--15, 1987

[10] Gregory, D; Shi, Q.; Merabti, M., 'An Intrusion Detection System Based upon Autonomous Mobile Agents", pp. 586-591, *14th International conference on Information security*, 1998 Aug : Vienna

[11] Y. Jou, F. Gong, C. Sargor, X. Wu, S. Wu, H. Chang, and F. Wang, "Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure", *DARPA Information Survivability Conference and Exposition 2000*, Jan 25-27, 2000, Vol. 2, pp 69-83

[12] R. A. Kemmerer, "Using Formal Methods to Analyze Encryption Protocols," *IEEE Journal on Selected Areas in Communications*, vol. 7, mo. 4, pp. 448-457, May 1989

[13] Rajeshekar Kailar and Virgil D. Gligor, "On Belief Evolution in Authentication Protocols", In *Proceedings of the Computer Security Foundations Workshop IV*, PP 103-16, IEEE Computer Society Press, Los Alamitos, CA, 1991

[14] Sandeep Kumar and Eugene Spafford, "A Taxonomy of Common Computer Security Vulnerabilities Based on their Method of Detection", Technical Report, Purdue University, 1995

[15] J. Kelsey, B. Schneier, and D. Wagner, "Protocol Interactions and the Chosen Protocol Attack", *Security Protocols, 5th, International Workshop April 1997*, Proceedings, Springer-Verlag, 1998, pp.91-104

[16] R. Kemmerer, C. Meadows, and J. Millen, ""Three Systems for Cryptographic Protocol Analysis", The *Journal of Cryptology*, Vol. 7, no. 2, 1993

[17] Ulf Lindqvist and Phillip A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)", *1999 IEEE Computer Society Symposium on Security and Privacy*, pp 146-61

[18] R.P. Lippman, D.J. Fried, I.Graf, J.W. Haines, K.R. Kendall, D. McCllung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, M.A. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation", *DARPA Information Survivability Conference and Exposition 2000*, Jan 25-27, 2000, Vol. 2, pp 12-26

[19] Gavin Lowe, "An Attack on the Needham-Schroeder Public Key Authentication Protocol", *Information Processing Letters*, 56:131-133, 1995

[20] Gavin Lowe, "Breaking and Fixing the Needham-Schroeder Public Key Protocol Using FDR", In *Proceedings of TACAS*, Vol. 1055 of *Lecture Notes in Computer Science*, pp 147-166, Springer-Verlag, 1996.

[21] Gavin Lowe, "Casper: A Compiler for the Analysis of Security Protocols", *Proceedings of 10th IEEE Computer Security Foundations Workshop*, 1997. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[22] Millen, J.K., Clark, S. C., and Freedman, S. B. "The interrogator: Protocol security analysis". *IEEE Trans. Sofw. eng.* SE-13, 2(Feb. 1987), pp. 274-288

[23] Catherine Meadows, "Formal Verification of Cryptographic Protocols: A Survey," *Advances in Cryptology - Asiacrypt '94*, LNSC 917, Springer-Verlag, 1995, pp. 133-150

[24] Catherine Meadows, "Analysis of the Internet Key Exchange Protocol using the NRL Protocol Analyzer", *1999 IEEE Computer Society Symposium on Security and Privacy*, pp 216-34

[25] Catherine Meadows, "A Formal Framework and Evaluation Method for Network Denial of Service", *12th IEEE Computer Security Foundations Workshop*, Jun 28-30, 1999, Mordano, Italy

[26] Roger M. Needham, Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", *Communications of the ACM*, December 1978 vol. 21 #12, pp. 993-999

[27] R. Oppliger. Security issues related to mobile code and agent-based systems. pp. 1165-1170. *Computer Communications*, Vol. 22, No. 12 (July 1999):

[28] Lawrence C. Paulson, "Proving Security Protocols Correct"', in *IEEE Symposium on Logic in Computer Science*, Trento, Italy (1999), pp 370-81

[29] A. W. Roscoe, "The Theory and Practice of Concurrency", Prentice Hall, 1997

[30] Dawn Xiaodong Song, "Athena: A New Efficient Automatic Checker for Security Protocol Analysis", *12th IEEE Computer Security Foundations Workshop*, Jun 28-30, 99, Mordano, Italy

[31] T. Sander, C. Tschudin, "Protecting Mobile Agents against Malicious Hosts", *Lecture Notes in Computer Science, Special Issue on Mobile Agents*, Edited by G. Vigna, 1998

[32] Paul Syverson, ``A Taxonomy of Replay Attacks," *Proceedings of the Computer Security Foundations Workshop VII*, Franconia NH, 1994 IEEE CS Press (Los Alamitos, 1994)

[33] F. Thayer, J.C. Herzog, and J.D. Guttman, "Strand Spaces: Why is a Security Protocol Correct?" In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, 1998

[34] Brett Tjaden, "A Method for Examining Cryptographic Protocols" University of Virginia Doctoral Dissertation, January 1997

[35] Vigna and Kemmerer, "NetSTAT: A Network-based Intrusion Detection System", *Journal of Computer Security*, Volume 7, Issue 1, 1999

[36] "Attacks on Encryption Code Raise Questions About Computer Vulnerability", Wayner, Peter, New York Times (01/05/00) P. C2

[37] Alec Yasinsac, "Evaluating Cryptographic Protocols:, Ph.D. Dissertation, University of Virginia, Jan 1996

[38] Yasinsac, Alec; Wulf, William A, "Evaluating Cryptographic Protocols", University of Virginia Technical Report, CS-93-66, December 22, 1993

[39] Alec Yasinsac and Wm. A. Wulf, "A Framework for A Cryptographic Protocol Evaluation Workbench", *Proceedings of the Fourth IEEE International High Assurance Systems Engineering Symposium* (HASE99), Washington D.C., Nov. 1999

[40] Alec Yasinsac, "Detecting Intrusions in Security Protocols", accepted to the *Third ACM Workshop on Intrusion Detection Systems*, Athens, Greece, Nov 1-4, 2000

[41] Susan Pancho, "Protocols", Proceedings of the New Security Paradigms Workshop", Sept. 1999

[42] Mihir Bellare and Phillip Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", Proceedings of the First ACM Conference on Communications and Computer Security, ACM, November, 1995

[43] Paul E. Proctor, The Practical Intrusion Detection Handbook, Prentice Hall, Inc. 2001, ISBN 013-025960-8, pp 46-47