NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-08-13
December 15, 2008

Exception Triggered DoS Attacks on Wireless Networks

Yao Zhao, Sagar Vemuri, Jiazhen Chen,
Yan Chen, Hai Zhou and Zhi (Judy) Fu

Abstract

Security protocols are not as secure as we assumed. In this paper, we identified a practical way to launch DoS attacks on security protocols by triggering exceptions. Through experiments, we show that even the latest strongly authenticated protocols such as PEAP, EAP-TLS and EAPTTLS are vulnerable to these attacks. Real attacks have been implemented and tested against TLS-based EAP protocols, the major family of security protocols for Wireless LAN, as well as the Return Routability of Mobile IPv6, an emerging lightweight security protocol in new IPv6 infrastructure. DoS attacks on PEAP, one popular TLS-based EAP protocol were performed and tested on a major university's wireless network, and the attacks were highly successful. We further tested the scalability of our attack through a series of ns-2 simulations. Countermeasures for detection of such attacks and improvements of the protocols to overcome these types of DoS attacks are also proposed and verified experimentally.

Keywords: Wireless network, DoS attack, exception

Exception Triggered DoS Attacks on Wireless Networks

Yao Zhao, Sagar Vemuri, Jiazhen Chen, Yan Chen, Hai Zhou and Zhi (Judy) Fu†
Northwestern University, Evanston IL, USA
†Motorola Labs, Schaumburg IL, USA

Abstract

Security protocols are not as secure as we assumed. In this paper, we identified a practical way to launch DoS attacks on security protocols by triggering exceptions. Through experiments, we show that even the latest strongly authenticated protocols such as PEAP, EAP-TLS and EAP-TTLS are vulnerable to these attacks. Real attacks have been implemented and tested against TLS-based EAP protocols, the major family of security protocols for Wireless LAN, as well as the Return Routability of Mobile IPv6, an emerging lightweight security protocol in new IPv6 infrastructure. DoS attacks on PEAP, one popular TLS-based EAP protocol were performed and tested on a major university's wireless network, and the attacks were highly successful. We further tested the scalability of our attack through a series of ns-2 simulations. Countermeasures for detection of such attacks and improvements of the protocols to overcome these types of DoS attacks are also proposed and verified experimentally.

1 Introduction

As the foundation of networks, protocols play vital roles in authentication and communication, and thus become one popular target of various malicious attacks. To be robust for unexpected events, a well designed protocol should consider and handle the exceptions properly. Unfortunately, the way of dealing with exceptions in most current protocols is simply to restart the protocol processing from the beginning, which causes a waste of resources and potentially can be used to launch serious denial-of-service (DoS) attacks. Note in this paper, we also consider the performance degradation attack as a weak DoS attack.

So far, people have paid little attention to such vulnerabilities caused by the (faked) exceptions including the TCP RST attacks [25], probably because of the lack of practical and severe attacks. In wired networks, it is very hard to sniff packets unless the network operators attempt to do so or a router gets compromised. Similarly, in wireless networks, TCP sessions are usually protected by lower layer encryptions. Therefore, it is hard to conduct TCP RST attack in practice, and that is why such vulnerabilities are largely ignored by the community.

In this paper, we reveal a serious vulnerability of exception handling in most wireless security and communication protocols by showing an exception triggered attack. The basic idea of the attack is to sniff the protocol communication and then inject *fake error messages* or *misleading messages* to trigger exception handling to bring down the whole protocol session. Note we assume the cryptographic algorithms are perfect and no encrypted data is readable by the attackers.

This attack has the following properties.

- **Easy to launch.** Any normal user can launch such an attack with cheap commodity hardware. Actually one important contribution of the paper is to demonstrate the easiness of such attacks in the real life using real-world experiments.
- **Efficient and scalable.** This attack only needs to send small error or unexpected messages. The traffic is much less than that of the normal connection setup. Thus, even a single node can attack a large number of clients simultaneously.
- **Stealthy.** Unlike jamming or rogue AS attacks, it cannot be detected by any of the existing intrusion detection systems.
- **Generally applicable to a variety of protocols.** Such attack can be applied to any security or communication protocol which involves exception handling in its connection setup. For example, it can even be applied to *strong* authentication protocols such as TLS in our case studies.

In this paper, we show two case studies. The first one is on TLS-based Extensible Authentication Protocols (EAP) such as PEAP [9], EAP-TLS [30] and EAP-TTLS [17]. EAP is widely used as a strong authentication protocol for wireless LANs and cellular networks. The second case study is on the Return Routability protocol of Mobile IPv6.

Through both real-world experiments and ns-2 simulations, we demonstrate the ease, efficiency and scalability of the exception triggered attacks. For example, with a commodity laptop, we can easily launch DoS attacks on PEAP authentications to stop clients from joining the the campus wireless network of a major university.

In addition, we propose countermeasures to such attacks. We analyze the symptoms of the exception triggered attacks and propose detection approaches. Also, we propose a game-theory based design principle to improve existing protocols against such attacks. We implemented such strategy in WPA supplicant [8] (an implementation of PEAP [9]) and show that the modified PEAP is immune to the fake error message based attacks in real experiments. Last, we propose the guideline for the design of future authentication protocols to be invulnerable to the exception triggered attacks.

The rest of the paper is organized as follows. In section 2, we describe the requirements and techniques to launch DoS attacks by triggering exceptions. In sections 3 and 4, two case studies for the exception triggered attacks are presented, and then they are evaluated in section 5. Next, in section 6, we discuss a few countermeasures to secure protocols against the DoS attacks. Finally, we present the related work in section 7 and conclude in section 8.

2 Attack Framework

In this section, we describe the framework of our exception triggered DoS attack on security protocols. Our attack principle is to explore the vulnerability of the protocol exception handling mechanism. There are a few requirements for launching the attack, but they are general and easy to satisfy in the real world, as shown in this section and the case studies in sections 3 and 4.

2.1 Attack Requirements

The following basic requirements are necessary to our attack.

- *Media Requirements:* The attacker can sniff and spoof packets.
- *Protocol Requirements:* Existence of fatal error condition to stop the protocol before the protocol is protected by strong cryptographic algorithms.
- *Timing Requirements:* Considerable time window exists between normal protocol communication, allowing attacks to inject packets at right place.

2.1.1 Media Requirements

Sniffing and spoofing are necessary for our exception triggered attack as well as many other attacks. Sniffing helps to know the status of the protocol and to determine when to spoof the attack packets.

Sniffing and spoofing are limited nowadays in wired networks. For example, current switches prevent sniffing in Ethernet and it is usually very hard to sniff on routers. ISPs and enterprises often use access control to block IP spoofing. However, in wireless networks, especially in 802.11 networks, sniffing and spoofing can be done with well designed software and off-the-shelf hardware. This is also the reason that we focus on attacks in wireless networks, although our attack principle should also work for protocols for wired networks.

2.1.2 Protocol Requirements

Typical security protocols usually exchange a few control messages to authenticate one or both of the communicating parties, before the session keys are established. All the messages before the establishment of the keys are unencrypted. Normally a protocol has the exception handling procedures, and an error message is usually triggered to inform the other partner whenever an exception occurs. In most protocols, the fatal or unexpected exceptions are handled by simply terminating the protocol, and the new authentication process will restart from scratch, if retried. This simple exception handling procedure could be used by an attacker to launch a DoS attack.

As an example, in TLS v1.1, there are a lot of unencrypted and unauthenticated error messages before the session key is established, and they could potentially be sent by either of the parties (client or the server) when an error condition occurs. In TLS v1.1, as long as there is a fatal error message, the authentication process will be terminated [15].

2.1.3 Timing Requirements

If any protocol satisfies the above requirements, it is possible to launch a DoS attack on the protocol by triggering exceptions. But for an attack to be practical, the spoofed packet should reach the target at the right time, not too early and not too late. So there is a so called *time window* for the attacker to spoof attack packets, and the attack is practical to launch if the time window is large enough.

Generally, the attacker first sniffs the messages on the channel to determine when to send the spoofed message. Then he sends a spoofed message to one of the parties to make it believe that some error happened so as to terminate the authentication process. Usually if the spoofed message reaches the target party before the legitimate expected message, the attack will take effect. Otherwise, a party will process the expected message and then move to a new state, in which the attack message may be obsolete and discarded silently. Therefore, the time window for an attack usually is from the moment of receiving the trigger message to the moment a legitimate expected message reaches the target.

2.2 Attack Methodology

The exception triggered attack can be launched in two ways. First, the attacker can directly spoof the error messages, which informs one or both parties involved in the communication the failure of the protocol. The second way is to send some misleading messages to trigger one party to send out an error message. For example, in a negotiation process the spoofed message can make the client choose a wrong cipher suite to talk with the authentication server.

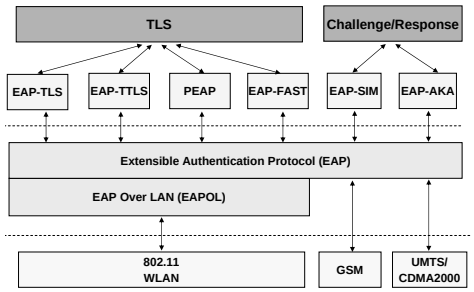

Figure 1: EAP protocols.

3 Case Study 1: TLS based EAP Protocols

In this section, we describe our first case study, exception triggered based attacks on TLS-based EAP protocols. We first give a brief background of the EAP protocol family and the TLS protocol, and then describe in details the attack procedure.

3.1 Background

3.1.1 Extensible Authentication Protocol (EAP)

Extensible Authentication Protocol (EAP) is a universal authentication framework that is frequently used in wireless networks, point-to-point connections and LANs.

EAP is not a specific authentication mechanism, but it provides a series of common functions and a negotiation process based on the desired authentication mechanism. Currently about 40 such mechanisms, called EAP methods, are supported. Those methods are defined in various IETF RFCs including EAP-MD5 [11], EAP-OTP [11], EAP-GTC [11], EAP-TLS [30], EAP-TTLS [17], PEAP [9], EAP-IKEv2 [31], EAP-SIM [18], EAP-AKA [12], and additionally a number of vendor specific methods. The most commonly used methods that are suitable for operating in wireless networks are: PEAP [9], EAP-TLS [30], EAP-TTLS [17],EAP-FAST [14], EAP-SIM [18] and EAP-AKA [12]. Figure 1 shows how these typical EAP protocols work with other layers. EAP-TLS, EAP-TTLS, EAP-FAST and PEAP are widely used in WLAN, EAP-SIM is designed for GSM systems and EAP-AKA is adopted in UMTS/CDMA2000. Among them, EAP-TLS, EAP-TTLS, EAP-FAST and PEAP all use TLS as their underlying authentication and cryptographic method.

3.1.2 Transport Layer Security (TLS)

Transport Layer Security (TLS) [15] and its predecessor, Secure Socket Layer (SSL) [6] are cryptographic protocols that provide secure communication on the Internet for secure web browsing, e-mail, Internet faxing, instant messaging and other data transfers. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. For example, HTTPS protocol layers on top of TLS protocol to protect sensitive network traffic.

TLS is a layered protocol, and consists of the Record protocol, the Alert protocol and the Handshake protocol. The Record protocol is designed to serve the Handshake protocol and Alert protocol, and offers symmetric encryption, data authenticity, and optionally compression [15]. The Alert protocol offers some signaling to the other protocols and between peers. An alert signal includes an alert level indication, and a FATAL ALERT always terminates the current connection. The Handshake protocol is responsible for the cipher suite negotiation, the initial key exchange, and the authentication of the two peers. In our attack we mainly attack the Handshake protocol by triggering the alert messages defined in the Alert protocol.

Figure 2 shows the flowchart of a successful TLS handshake process. We do not introduce the handshake protocol in detail due to space limitation, but the details can be found in [15]. Generally, the TLS server starts the procedure and the client responds with a hello message. The server then sends its certificate, selected cipher suit and so on. Note that TLS provides an option to authenticate the client as well by requesting the client certificate. The client returns selected cipher, its certificate (optional) and other cryptographic information. Note in Figure 2, we put some messages close as if bundled together because they may be embedded in a single EAP packet, depending on the message sizes.

3.2 Vulnerability in TLS based EAP Protocols

The vulnerability we find is in the TLS protocol, which is widely used in many security protocols such as HTTPS and TLS based EAP protocols. Therefore, all the TLS based EAP protocols such as PEAP, EAP-TLS, EAP-TTLS and EAP-FAST will be vulnerable to our attack. It is worth mentioning that we also implemented the same attack approach targeting HTTPS protocol which also relies on TLS, and the attack successfully stopped the GMail authentication using the HTTPS protocol. However, the hardness of sniffing and deciphering application layer data in wired and encrypted wireless networks makes such application level attack not very practical. Hence we mainly focus on the TLS based EAP protocols, which are MAC layer authentication protocols for wireless networks.

An attacker sniffs the communication between the wireless client and the access point, inspecting the authentication procedure through the Handshake protocol of TLS. Note before the handshake protocol of TLS establishes the keys to encrypt following packets, all the wireless packets are in clear-text and unencrypted. Triggered by some messages, the attacker spoofs corresponding messages to make the TLS authentication fail. In particular, the attacker have two ways to trigger the exception handling in TLS: 1) spoofing the FATAL ALERT messages to directly fool the client or the server to stop the authentication, 2) spoofing negotiation messages with different authentication parameters to confuse the two parties. Next we will discuss the detailed

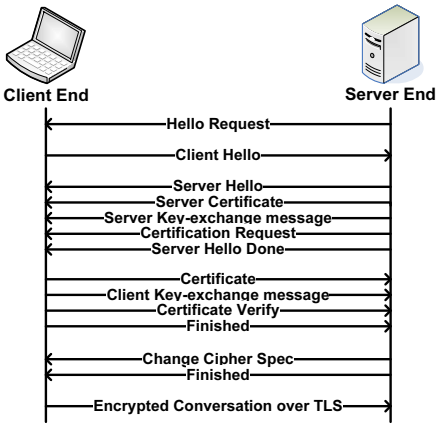

Figure 2: TLS Handshake.

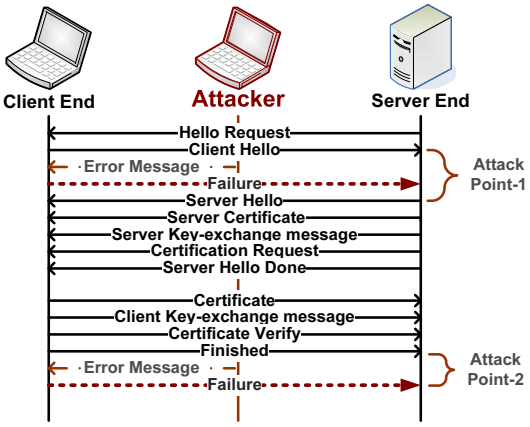

Figure 3: TLS Handshake failed - Attacker spoofi ng Server.

attack approaches.

3.2.1 Exception Triggered based Attack

Spoofing Server The attacker can spoof the messages as if they were coming from the TLS server [1]. There are at least two attack points in the TLS authentication framework: one is after the client sending the CLIENT HELLO and the other is the point after the bundle of client response messages (including CLIENT CERTIFICATE, CLIENT KEY EXCHANGE and other messages) (See Figure 3). If the bundles of messages are broken into multiple EAP packets, we have more attack points after each EAP packet.

The attacks to different points are similar and we use the first one as an example. As shown in figure 3, if the attacker sniffs a CLIENT HELLO message, he simply spoofs a FATAL ALERT message. The SERVER HELLO and further messages from the server are dropped. The client considers only the alert message, and sends a failure message to the server. Since the TLS transaction has failed, the encapsulating EAP protocol terminates with an EAP FAILURE message. An important problem in this attack is to satisfy the timing requirement. The attacker needs his spoofed message to reach the client earlier than the normal SERVER HELLO message from the server (See Figure 3). This time gap contains: 1) the message delivery time in wired network (both from the AP to the TLS server and from the TLS server to the AP), 2) the server's processing time and 3) the message delivery time from the AP to the client in the wireless network. This time gap may vary in different situations. For example, at the second attack point, the server usually needs to query the database and then verify the user's identity and password, and hence the processing time may be large. The background traffic affects the transmitting time in both wireless and wired networks. Meanwhile, the attacker's spoofing time contains two parts: 1) the time generating the spoofed message, which is negligible and 2) the transmitting time in the wireless network.

Spoofing Client Similarly, the attacker can spoof to be the client, and there are also at least two attack points according to the TLS authentication protocol. As shown in Figure 4, the first attack point is right after the client sniffing the HELLO REQUEST and the other is after receiving the SERVER HELLO DONE.

As shown in Figure 4, if the attacker's FATAL ALERT message reaches the server before the CLIENT HELLO message, then further messages from the client are dropped since the server considers only the alert message and sends a failure message to the client indicating the detection of an error. Then the EAP protocol terminates with an EAP FAILURE message. The second attack point is quite similar and hence we do not repeat its details.

3.2.2 Misleading Message based Attack

Besides directly spoofing and sending error messages to destroy the TLS communication, the attacker is also able to stealthily spoof misleading messages that intentionally trigger the exception handling mechanism of the TLS protocol. We found out that using a spoofed SERVER HELLO message with various parameter setting will cause the client side automatically respond a FATAL ALERT message to the server side.

The message exchange procedure is shown in the Figure 5. Regarding to the TLS RFC, the client side should provide a valid CLIENT HELLO message that contains a list of supportive cipher suites in the first place. After that the server responds to the client a SERVER HELLO message that indicates its choice of the cipher suite. Besides the SERVER HELLO message, client side needs to collect other required messages before moving forward to the next state, *e.g.*, SERVER KEY-EXCHANGE message. Therefore when the attack takes place, the client side will receive two SERVER HELLO messages. In that case, the client itself is

[1]To do so, we use AP's MAC address as the source MAC address of the spoofed message to spoof the AP. Note EAP protocol is directly on top of the MAC layer, and the IP and higher layers are not used at this stage. Therefore, attacks on IP layers and above such as TCP RST attack are not applicable in this scenario at all.

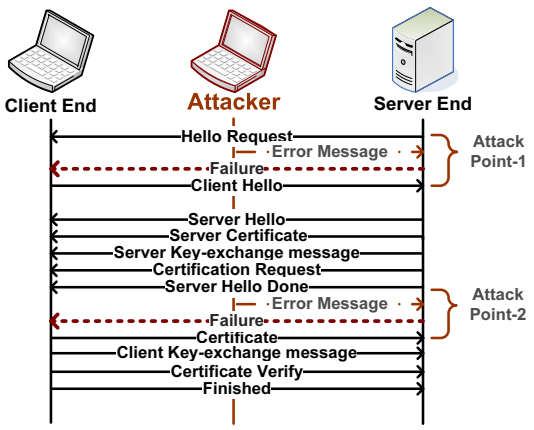

Figure 4: TLS Handshake failed - Attacker spoofi ng Client.

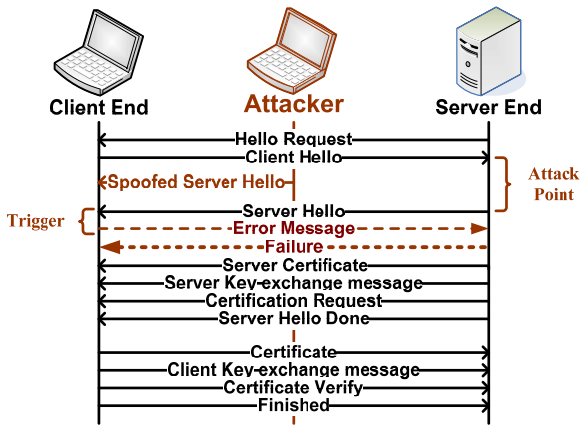

Figure 5: TLS Handshake failed - Attacker Trigger Exceptions.

not able to figure out which of the SERVER HELLO messages is the authenticate one that should be used in the following process. Thus the client side has to notify the server side that there is an error happened using the FATAL ALERT message and eventually terminate the whole procedure.

3.2.3 Discussion

Comparison between Error Message based Attack and Misleading Message based Attack Compared to the direct error message based attack, the misleading message based attack is more stealthy and harder to deal with. If the unprotected messages have multiple parameters to choose or set, the spoofed messages have lots of tricks to play. Even if the receiver notices the attack and get multiple "normal" messages with different content, the receiver may not be able to differentiate the legitimate message with the faked ones. This brings the difficulty in countermeasures and we will further discuss it in Section 6.

Increasing the Attack Success Rate Obviously our attacks on TLS based EAP protocols satisfy the medium requirement and the channel requirement (see Section 2.1). The only potential problem is the timing requirement, which requires the attack packet to reach the victim earlier than the legitimate packet. To make the attacker's packet sent out to air faster, we play a trick described below.

Wireless LAN uses the IEEE 802.11's DCF (Distributed Coordination Function) [10], which is based on CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). According to the collision avoidance mechanism of CSMA/CA, a station performs a back-off procedure before initiating the transmission of a frame. After detecting that the medium is idle for a DIFS (DCF inter-frame spacing) interval, the station selects a random backoff period from [0, CW-1], where CW is referred to as contention window. The station waits for the channel to be idle for a total time equal to this back-off period, after which it can transmit a data frame. The contention window CW has an initial value CWMin (31 in the standard), and is doubled when a collision occurs, up to the maximum value CWMax. When a frame is successfully transmitted, the contention window is set to its initial value CWMin.

To send packet faster, a simple way is to fix the CWMin and CWMax to the minimal number. Attacker can set its CWMin to be as small as 1, and in this case the back-off time of the attacker is much smaller than that of the normal wireless nodes. In this case, the chance of sending the attack packet earlier than the legitimate packet is close to 1. In our real experiments (See Section 5.1.1), we find that MADWifi driver [4] provides command line parameters to change the CWMin and CWMax easily.

3.3 Generalization to Other EAP protocols

The exception triggered vulnerability does not only lie in TLS based EAP protocols. Analyzing the authentication protocols in 2G and 3G cellular networks, EAP-SIM and EAP-AKA, we found they also potentially have a similar vulnerability. In EAP-SIM and EAP-AKA, the EAP NOTIFICATION message is used to indicate the result and exception cases, which is not protected through the authentication procedure. This protocol design offers the attacker an opportunity to maliciously use the EAP-Response/SIM/Client-Error message and EAP-Response/AKA/Client-Error message to interrupt the processing between clients and the authentication server and fail the authentication procedure. Note that the attack on EAP-SIM and EAP-AKA will work analytically, and it will be our future work to study its practicality.

4 Case Study 2: Mobile IPv6 Return Routability Procedure

In this section, we describe our second case study on Mobile IPv6 Return Routability (RR) procedure. We first introduce Mobile IPv6 RR procedure, and then describe in detail the attack procedure.

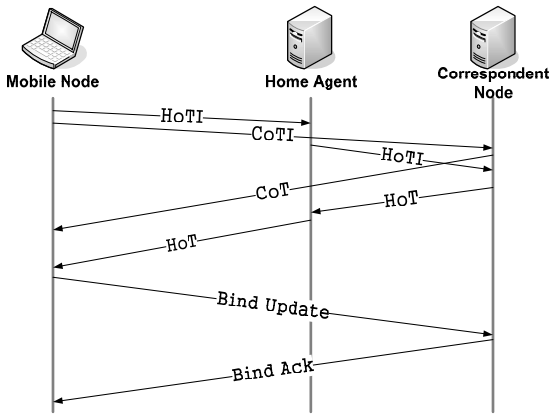

Figure 6: MIPv6 Return Routability procedure.

4.1 Background

Mobile IPv6 [20] is a protocol which allows nodes to remain reachable while moving around in the IPv6 Internet. Each mobile node (or MN in short) has a home address, regardless of its current location. If the mobile node roams to the remote network while communicating with other nodes (called Correspondent Nodes or CN), the packets from the Correspondent Nodes go to the home network first. These packets will be further forwarded to the MN's address in the remote network (called care-of address) by the Home Agent. Obviously this triangular routing is not optimized and hence the Return Routability procedure is proposed in Mobile IPv6 to allow the direct routing. Basically, the Mobile Node tells its care-of address to the Correspondent Nodes, and the following traffics from CNs are sent to the MN's care-of address directly. The application layer is not aware of the changes of the IP layer. To secure the Return Routability procedure, a simple weak authentication protocol is used, detailed in the following paragraphs.

Figure 6 shows the message exchange in the Return Routability procedure (RR procedure). The RR procedure begins when the MN sends HOME TEST INIT(HOTI) message to CN through HA and the CARE-OF TEST INIT(COTI) message directly to CN. CN responds with a HoT sent through HA and a CoT sent directly to MN. MN uses the information in HoT and CoT to generate a key, which it uses to sign the BINDING UPDATE message to CN. Upon the receipt of the BINDING UPDATE, CN adds an entry for the MN in its binding cache and optionally sends BINDING ACKNOWLEDGEMENT. Once this happens, MN and CN will be capable of communicating over a direct route. Thus they need not have to go through the HA any more and hence the route between MN and CN is optimized.

4.2 Vulnerabilities in Mobile IPv6 RR Procedure

Now we introduce two vulnerabilities in the MIPv6 RR procedure to the exception triggered attack: binding error vulnerability and binding acknowledgement vulnerability.

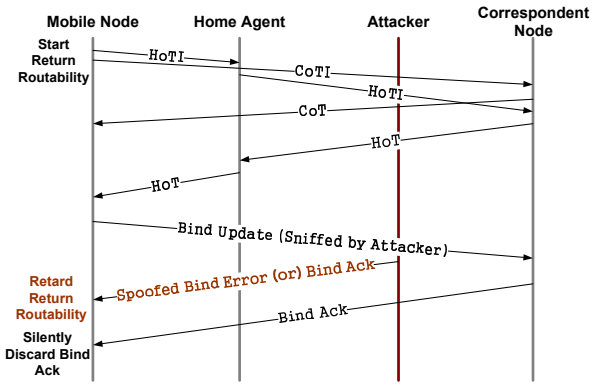

Figure 7: MIPv6 Bind Error and Bind Ack Vulnerability.

First, according to Mobile IPv6 [20], if a BINDING ERROR message is sent from the CN to the MN with the status field set to 2 (meaning unrecognized mobility header), the mobile node should cease the attempt to use route optimization. Since the BINDING ERROR message is not protected, it could easily be spoofed by an attacker to nullify the Return Routability procedure. As shown in Figure 7, when the attacker sniffs the BINDING UPDATE sent by the Mobile Node, it sends the spoofed BINDING ERROR to the Mobile Node. If the spoofed BINDING ERROR reaches the MN earlier than the valid BINDING ACKNOWLEDGEMENT from the CN, the MN will discard the valid BINDING ACKNOWLEDGEMENT, and thus the Return Routability procedure fails. Usually the CN is far away from the MN while the attacker is in the same wireless network as the MN, the attacking time window is quite loose for the attacker.

Second, similar to the binding error vulnerability, the binding acknowledgement vulnerability arises from the fact that BINDING ACKNOWLEDGEMENT message with status 136, 137 and 138 is used to indicate an error and is not protected at all [20]. Similarly, it could be easily spoofed by an external entity and accepted by the Mobile Node. The timing diagram in Figure 7 gives the details.

Disrupting On-Going Sessions We can disrupt the route optimization in on-going sessions as well, not limited to new sessions. Mobile IPv6 RFC states that the Return Routability procedure is repeated every few minutes so that the communication can be maintained and the binding keys can be updated. For example, in the MIPL [5] implementation, by default the binding keys should be updated every 3 minutes by performing the Return Routability procedure again. So the attacker only needs to waits for at most 3 minutes to disrupt an on-going session's route optimization, using the two vulnerabilities introduced above.

5 Evaluation

In this section, we describe our evaluations on our exception triggered based attack on both TLS-based EAP protocols and MIPv6 Return Routability procedure.

5.1 Evaluation of DoS Attacks on TLS-based EAP Protocols

First, we use the real world experiment to evaluate the attack feasibility and efficiency. Then we rely on simulations to show the scalability of the attack on multiple users simultaneously. In the evaluation we use PEAP [9] as the example, simply because PEAP is widely deployed, and we have at hand a large operative campus wireless network using PEAP to authenticate about 15,000 users. We believe the real experiments on an operative network lead to the most genuine evaluation results.

5.1.1 Real-world Experiment

Experiment Methodology The university's wireless network authenticates its users using PEAP (Protected EAP) [9], which adopts PEAPv0/EAP-MSCHAPv2 with TLSv1.1 [15] as the security method. The Access Point (AP) passes the EAP messages to the back-end authentication server for processing. Unless the authentication is successfully done, the client computer cannot obtain an IP address and therefore is unable to access the network resources. In our experiment, we use up to three laptops as the normal clients to connect to the university's wireless network, while another laptop acts as an attacker.

We first executed some controlled in-lab experiments. We used different wireless cards in various operating systems as the normal users, and tried to launch DoS attacks against them. The various wireless network management utilities we tested include: 1) the windows native client utility of windows XP and Vista, 2) the Dell utility, 3) the Proxim utility, and 4) the Linux NetworkManager utility of Ubuntu. We also let 1 to 3 clients connect to the network at the same time, so that we can test the attack against multiple simultaneous authentications. We also tested the attack in the university's cafeteria, where the clients are more diverse and out of our control. Note that in the cafe environment, there are two channels available and our attacker only works on one channel. If one channel does not work properly, the client software usually automatically switches to another channel, and hence we do not really DoS the network[2].

The attacker program was written in C++ on the Linux platform. It uses MADWifi drivers [4] and libpcap library [2] to sniff the channel and leverages on the lorcon [3] library[3] to send spoofed messages. We also use MADWifi drivers [4] to configure the wireless cards, *e.g.*, changing the CWMin parameter of IEEE 802.11.

Experimental Results

Feasibility. We found that different WLAN card management programs have quite different automatic retry functions. The windows native client utility of windows XP and Vista, and the Dell utility try to connect to the network only once. If the authentication fails, they do not automatically retry the connection, but wait for some user initiated action, *e.g.*, entering a new password. On the other hand the Proxim utility and the Linux NetworkManager utility of Ubuntu store the supplied credentials and automatically retry to connect to the network even if it fails multiple times. Especially, the Linux NetworkManager tries 6 times on average with the supplied credentials, and then it pops up a dialog box asking for a new credential. Although the automatic retry function may make the attack to be harder, we do not find any difference in our real experiments on the attack success rate, shown in the following paragraphs.

We also studied the time taken by the attacker to respond with a spoofed message once it has encountered an appropriate message. We found that on average it takes 0.29 milliseconds to compute the spoofed message and put it in the air. The numbers are obtained by measuring at the attacker the time difference from the time an EAP/TLS response is received to the time at which a spoofed error message is put on the channel. The average time for getting a reply from the server to the client was around 10ms. Thus there is a sufficient timing window to launch the DoS attacks.

Success Rates. In our lab experiments, we tested our DoS attack against $1 \sim 3$ clients which attempted to connect to the major university's wireless network simultaneously. We tested both the error message based attack (including spoofing both the client and the server) and the misleading message based attack. Our attacker always achieved a 100% success rate, as no clients could get authenticated and enter the wireless network under the attack. We also performed the experiment in the university's cafeteria as mentioned above. Our attacker was 100% successful in these experiments on the channel which it was running on, as we did not detect even a single EAP SUCCESS message. This experiment ran for 35 minutes, and all 7 different hosts were observed with failed authentications.

Efficiency. Unlike the jamming based attack, the exception triggered attack needn't send attack messages frequently. Suppose the error message based attacker successfully disrupts the victim computer at the first attack point, it only costs the attacker one spoofed error message of 79 bytes, compared to 14 messages of a total length of 1480 bytes between the victim clients and the TLS server [4]. Thus the attack efficiency ratio for spoofing error messages (*i.e.* attack traffic compared to the authentication traffic) is 5.34% in terms of packet volume and the ratio in the number of messages is 7.14%. For the success of the second attack point, the ratio in volume is 6.84% and the ratio in the number of messages is 6.25%. If the attack with misleading message is used, the length of the SERVER HELLO

[2] A laptop can easily equip three or even more wireless cards via PCMCIA and USB interfaces and hence attacking multiple channels simultaneously is not a problem for out attack.

[3] The lorcon library was tweaked a little so that it does not change the content of spoofed packets.

[4] The 14 messages includes the EAP messages and the IEEE 802.11 control messages involved in the authentication, which are not shown in the figure of the EAP protocol communication.

message is 119 bytes and there is only one attack point. The efficiency ratio in terms of packet volume is 8.04%, and the ratio in the number of messages is 7.14%.

5.1.2 Ns-2 Simulation

Simulation Methodology We simulated the DoS attack of spoofing a server in ns-2 simulator [7] to study the performance and scalability of our attacker. Ns-2, by itself, does not have the TLS and EAP protocols, and hence we first implemented the TLS and EAP modules in it. In our simulation, we simulate the error message based attack of spoofing the server. Similar to the real attack, the attacker spoofs a FATAL ALERT message and sends it to the client whenever it sniffs a relevant message from the TLS client to the TLS server. The attacker was active at two attack points, the same as the real attacker (See Section 3.2.1). The attacker has the ability to change the CWMin parameter of his WLAN card driver so that the attacker can potentially get fast access to the channel. We vary the CWMin of the attacker node from 1 to 31, while keeping the CWMin as 31 for all other nodes. The automatic retry feature is also implemented into the TLS protocol. The TLS authentication process is restarted if the client fails in its authentication attempt. By default, the client tries for a maximum of 18 times before it gives up completely. The number of 18 comes from the imagined typical scenario: Linux NetworkManager retries 6 times on average before giving user a chance to enter new credentials, and the user tries for 3 times before giving up. An interval of 1 second, as observed from our real-world experiment, is set between each retry.

In our ns-2 simulation setup we create one TLS server, one TLS attacker, and vary the number of TLS clients from 1 to a maximum of 50. We believe it is extremely rare for more than 50 clients to authenticate simultaneously in a wireless LAN network.The inter-arrival time between clients is very small, randomly chosen between 0 and 0.5s. This small interval guarantees that the authentications of all the clients are overlapped. The access points broadcast BEACON messages every 50ms, as observed in the real experiments. The TLS server and access points are connected via wired network in practice. However, for convenience we put them on the same node and inject certain latency in the communication between the TLS server and the access point. This latency is to simulate the wired network delay and the TLS server's processing delay. The default latency is set to be 10ms, which is observed in our real experiment in the campus setting. We also vary this latency in the simulation to study its effect on the attack success rates. All the results are taken on an average of 20 runs.

Experimental Results Figure 8 shows the attack success rate with different CWMin values as a function of number of clients. Except for CWMin=31, the attack stops all the authentication attempts, no matter how many clients are trying simultaneously. With CWMin as 31, the attacker has the same ability as normal clients to access the channel, and hence with certain possibility the attack packets are later than the legitimate responses, letting some clients get authenticated. Interestingly, we see the authentication success rate goes up first and then drop to 0. By inspecting the logs carefully, we find that when the number of clients is small (*e.g.*, less than 10), the TLS server delay plays an important role. The attack packet can usually get into channel during the 10ms delay. But as the number of clients increase, the attacker may have to wait more than 10ms to get the channel when other nodes, by chance, occupy the channel to send packets. So the 10ms advantage of the attacker over the AP is mitigated. However, as the number of clients keeps increasing, the AP has more and more packets in its queue. Thus the TLS server response may be delayed for quite a long time at AP. On the other hand, the attacker packets usually are not queued, because attacker has much less packets to send than the AP.

Figure 9 shows the effect of the TLS server delay on the TLS Attack. This delay aids the attacker by giving it a much larger time window and hence the attacker has higher success rate. So we can see that the smaller the server delay, the larger the authentication success rate. Consider the extreme case when the server delay is 0 ms and CWMin is set to be 31. It is clear that as the number of clients increase, the packets queued at the AP play an important role, and hence decreases the authentication success rate when there are more and more simultaneous authentications. Still, when CWMin=1, no authentication can finish at all.

5.2 Evaluation on MIPv6 Route Optimization Protocol

Being the emerging new infrastructure, MIPv6 as well as IPv6 is not widely deployed currently. So we conducted a testbed experiment to demonstrate the practicality of this attack. Due to the space limit, we briefly describe our testbed experiment and results in this paper, leaving the details in our technique report [33].

We built the Mobile IPv6 network using the Mobile IPv6 Implementation for Linux (MIPL v2.0.2) [5], the most popular open source MIPv6 implementation. The underlying network is all IPv4, and all logical interfaces were GRE-based (Generic Routing Encapsulation developed by Cisco), to tunnel IPv6-over-IPv4.

In the experiment, the Mobile Node was first in the Home Network, and it kept pinging the Correspondent Node continuously. Then we moved it to the Foreign Network. After configuring the new care-of address and registering it with the Home Agent, the Mobile Node started return routability. We conducted the experiment for 100 runs. Every time we observed that, with the attacker in action, the return routability procedure was disrupted and the Mobile Node repeatedly tried to complete the procedure. While the retry attempts were being made, none of the ICMPv6 echo requests (Ping) reached the Correspondent Node. Hence, in the MIPL implementation [5], the effect of the attack in-

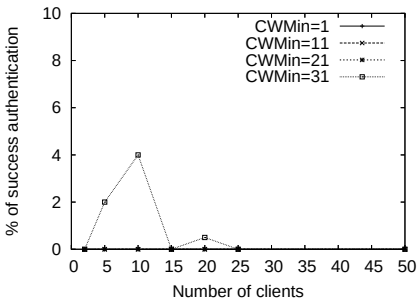

Figure 8: Authentication success rate with different CWMin.

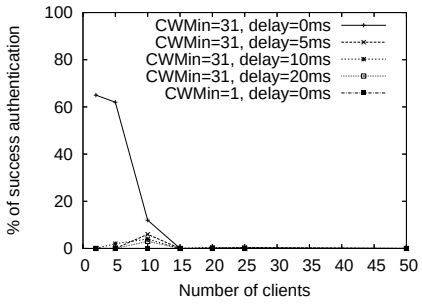

Figure 9: Authentication success rate with various server delays.

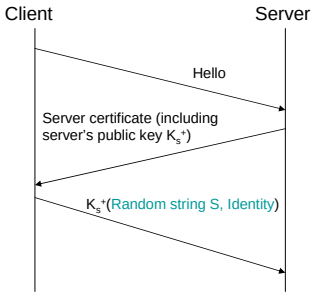

Figure 10: The design of robust security protocol.

creased and made it a DoS attack instead of a performance degradation attack.

6 Countermeasures to Exception Triggered DoS Attacks

In this section, we discuss countermeasures to the exception triggered attacks. First, we consider how to detect such attacks, which is not supported generally by intrusion detection systems, to the best of our knowledge. Since it is usually more desirable to secure a system against such attacks, we propose a general approach that can be easily adopted by protocols to gain robustness against the exception triggered attacks. Third, we propose the design guideline for future authentication protocols to be invulnerable to such attacks.

6.1 Attack Detection

The detection of the exception triggered attacks cannot be done by the general signature based IDS such as snort [29] or network IDS such as bro [28]. The detection must understand the protocol and identify the symptoms of the attack. We find there are two symptoms of such attacks: 1) There are duplicated response messages with different contents in the same stage of the protocol's state machine. 2) The protocol finally ends abnormally. If such two symptoms happen many times in a certain time period, it is an indication that the exception triggered attack is in action. Note the first symptom can rule out most 'normal' exceptions, such as wrong password.

6.2 Protocol Improvement against Exception Triggered Attacks

To secure a protocol against such attacks, we propose a game-theory based design principle. For a protocol entity, the processing of a received message will be delayed if its consequence is costly, and the amount of delay time will be proportional to the cost of the consequence. When multiple conflicting messages are received, the one with the least cost of consequence should be taken. This preference is particular helpful to deal with attacks directly using error messages. In case that multiple messages are of the same consequence but differ on contents such as authentication parameters (*e.g.* the misleading message based attack), there are two solutions:

- The receiver randomly accepts one message because the receiver cannot find out the legitimate packet. Therefore, the authentication has some chance to be success if the receiver by chance picks the correct one. With multiple times of authentication retries, the authentication is expected to get through anyway.
- The receiver accepts all the messages and starts different branches to deal with them. With this solution, the authentication will succeed without retries, which is the advantage of the solution. However, the implementation is much complex, especially when the authentication protocol takes several steps of unprotected communication before encryption.

Take our case study on TLS-base EAP protocol as an example. The fatal error messages and their triggering messages cause the worst consequence, *i.e.*, a termination of the authentication. So these messages will take longest delay in processing (*e.g.*, 1 sec in our countermeasure implementation described in Section 6.2), and such introduced latency has only limited effect on the protocol performance. For normal messages, the delay time will be very small (*e.g.*, hundreds of milliseconds) in order to maintain the performance. If the entity receives both a fatal error message and a normal message that advances the authentication stage, the entity will select the latter one, which obviously has a consequence of least cost.

There may be a concern that smart attackers will utilize the countermeasures for new attacks. For example, an attacker may spoof "correct" messages to advance an authentication even if the authentication has real errors and should terminate. However, the attack cannot proceed with the fake authentication for a long time without paying a proportional cost, and can never finish it because of the lack of the secrecy. So potentially the attack can delay the restart of the authentication shortly, but this consequence is not severe.

Implementation and Experiments of Countermeasures We implemented the above proposals into the TLS protocol

to test its effectiveness. Since we only have the control of clients, we modified the client software, WPA Supplicant v0.5.10 [8] software to incorporate the above approaches in the client side.

First, WPA Supplicant was modified so that normal messages are preferred over the error messages. Whenever the client receives a TLS ALERT message it waits for a maximum of 1 second before processing the alert message. If the client gets a legitimate message that fits the current state of the authentication process, then it discards the alert message and processes the other messages. Second, the misleading message attack can only send multiple SERVER HELLO and Server Key Exchange messages, but the server certificate cannot be spoofed. The modified WPA Supplicant will generate multiple keys and finally only the correct one can work in the following communications. The tests were performed in the same experimental setup as described in 5.1.1. The experiments were conducted 10 times and the attacker was never successful in any of the runs. It is because that the client always receives the real server response message before the one second threshold. Also, since the client will process the real positive messages immediately, there is no delay on the authentication time. Our experiments show that the average delay of authentication remains the same as that of the clients without being attacked.

In short, the experiments and results show that our countermeasures are easy to be adopted by existing protocols, and are effective against the exception triggered attacks in practice.

6.3 Design of Robust Security Protocols

In this section, we present the guideline for new protocol design to be robust to the exception triggered attack. The high-level idea is: Get packets encrypted or authenticated as early as possible. And do not have complex states and parameters before packets are encrypted or authenticated.

Figure 10 illustrates an example of the design philosophy. First, the client sends a HELLO message to request for authentication. The HELLO message only notifies the server for authentication and does not include any particular information such as the identity of the client. Second, the server sends back its certificate, signed by the well-known certificate authorities. In the certificate, the client obtains the verified public key K_s^+ of the server. Third, the client uses the public key K_s^+ to encrypt a random session identity I and a random string S and sends back the encrypted packet. In the following communications, the server and the client take I as the session ID and use S to authenticate packets. Because S is hidden from the sniffer, the attack has no way to interfere the legitimate authentication procedure once the session ID I and shared key S are established. Note we can also use Deffie-Hellman key exchange algorithm [16] to take place the server certificate to exchange S against eavesdroppers, if server certificate is not available.

Let's consider if attacker can attack during the communication of the first three messages. The first HELLO message has no useful information for the attacker, and the server even does not need to allocate a new session. The second message is server's certificate, which cannot be spoofed. After getting the server certificate, the attacker may send back an error message, *e.g.*, saying the certificate is wrong, or send back a misleading message, *e.g.*, with guessed identity I' and a random string S'. Neither the attack will work actually. The server simply ignores the error message and only waits for encrypted messages with the server's public key. The misleading message will be accepted by the server, but since the attacker's I' and S' are different to the client's, the attacker cannot interfere with the client's authentication.

7 Related Work

Wireless networks seem to be more vulnerable than the wired networks, simply because of the open media nature of the wireless networks. Since the thrive of the wireless LAN, cellular networks, ad hoc and sensor networks, there are a great number of fresh vulnerabilities and novel attacks discovered on different wireless networks [1, 13, 19, 24, 27, 32]. This paper mainly focuses on the DoS attacks, especially on protocols of wireless LAN.

Denial of Service attack is a notorious problem in wireless networks. Since the early time of radio networks, jamming has been a powerful method to disable wireless communications. Recently in [32], Xu *et al.* studied the feasibility of launching and detecting jamming attacks in wireless networks. They proposed four different jamming attack models that can be used by an attacker to disable the operation of a wireless network, and evaluated the effectiveness of each attack model. Noubir *et al.* investigated the resiliency to jamming on data protocols, such as IP, over WLAN [27]. They concluded that without good coding, the jamming attack can be very efficient by only injecting a single bit to disrupt a packet in current WLAN.

Other than jamming, people also found many other stealthy and practical attacks on wireless networks, especially on WLAN. In [13], Bellardo *et al.* described some vulnerabilities in the 802.11 management and media access services, and more importantly, implemented them to show the practice. They demonstrated with the deauthentication and virtual carrier sense attacks, which can easily disable the wireless connections with PDAs. Martinovic *et al.* proposed two different and novel attacks against web-based authentication in wireless environment [21]. The first attack is similar to the rogue AP attack in order to hijack wireless clients, while the other attack was based on the well-known ARP spoofing.

It is a popular direction to use model checkers to automatically find vulnerabilities in protocols [22–24, 26]. Narayana *et al.* used TLA+ to automatically find vulnerabilities in WiMax initial ranging and authentication process. Mitchell *et al.* used the finite-state verification tools, Murϕ, to analyze authentication protocols such as SSL,

EAP-GPSK and others [22–24]. Among them, EAP-GPSK analysis [24] is most related to our work. The Denial-of-Service attack in [24] also spoofs messages to confuse the authentication entities and stops the authentication process. It is similar in principle to our attacks on TLS based EAP protocols, but differs much in the real implementation. While Mitchell *et al.* mainly worked on the theoretical analysis and proof, our work focus more on the practical side and use real experiments to demonstrate the effectiveness of the attacks. Meanwhile, instead of studying protocols case by case with the same methodology, we tend to identify one potential vulnerable component in different protocols, *i.e.*, the exception handle with error messages.

8 Conclusion

In this paper, we propose the exception triggered denial-of-service attacks on wireless security protocols. The attacks explore the vulnerabilities in the exception handling process in security protocols, *i.e.*, blindly accepting error messages and terminating the communication imprudently. We demonstrated the effect of these attacks on two case studies: TLS-based EAP protocols and the Return Routability procedure of Mobile IPv6 protocol. Using real-world experiments and testbed experiments we prove the success and the practicality of the attacks, with off-the-shelf hardwares and well-written softwares available online. We also propose the detection scheme and protocol improvement principle against the exception triggered based attacks. Real implementation and experiments demonstrate the effect of the proposed countermeasures.

References

[1] Aircrack-ng. `http://aircrack-ng.org`.

[2] Libpcap library. `http://www.tcpdump.org/`.

[3] LORCON library - LOss of Radio Connectivity. `http://802.11ninja.net/lorcon`.

[4] Madwifi OpenSource Driver. `http://madwifi.org/`.

[5] Mipl: Mobile ipv6 for linux. `http://www.mipl.mediapoli.com/`.

[6] Secure Socket Layer 3.0. `http://wp.netscape.com/eng/ssl3/`.

[7] The Network Simulator - ns-2. `http://www.isi.edu/nsnam/ns/`.

[8] WPA Supplicant. `http://hostap.epitest.fi/wpa_supplicant/`.

[9] Microsoft PEAP version 0, 2002. `http://www.watersprings.org/pub/id/draft-kamath-pppext-peapv0-00.txt`.

[10] IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifi cations, Aug. 1999.

[11] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and E. H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, Jun. 2004.

[12] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). RFC 4187, Jan. 2006.

[13] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *USENIX Security Symposium*, 2003.

[14] N. Cam-Winget, D. McGrew, J. Salowey, and H. Zhou. The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST). RFC 4851, May. 2007.

[15] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Apr. 2006.

[16] W. Diffi e and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[17] P. Funk and S. Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). RFC 5281, Aug. 2008.

[18] E. H. Haverinen and E. J. Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). RFC 4186, Jan. 2006.

[19] Y. Hu, A. Perrig, and D. J. P. leashes. A defense against wormhole attacks in wireless network. In *IEEE Infocom*, 2003.

[20] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775, Jun. 2004.

[21] I. Martinovic, F. A. Zdarsky, A. Bacliorek, C. Jung, and J. B. Schmitt. Phishing in the wireless: Implementation and analysis. In *IFIP International Information Security Conference*, 2007.

[22] J. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *USENIX Security Symposium*, 1998.

[23] J. C. Mitchell, M. Mitchell, and U.Stern. Automated analysis of cryptographic protocols using murphi. In *IEEE Symp. Security and Privacy*, 1997.

[24] J. C. Mitchell, A. Roy, P. Rowe, and A. Scedrov. Analysis of EAP-GPSK authentication protocol. In *ACNS*, 2008.

[25] A. Mondal and A. Kuzmanovic. A poisoning-resilient tcp stack. In *IEEE ICNP 2007, Beijing, China*, 2007.

[26] P. Narayana, R. Chen, Y. Zhao, Y. Chen, Z. Fu, and H. Zhou. Automatic vulnerability checking of IEEE 802.16 WiMAX protocols through TLA+. In *Workshop on Secure Network Protocols (NPSec)*, 2006.

[27] G. Noubir and G. Lin. Low power DoS attacks in data wireless LANs and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3), 2003.

[28] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31, 1999.

[29] M. Roesch. Snort: The lightweight network intrusion detection system, 2001. `http://www.snort.org/`.

[30] D. Simon, B. Aboba, and R. Hurst. The EAP-TLS Authentication Protocol. RFC 5216, Mar. 2008.

[31] H. Tschofenig and et. al. The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method. RFC 5106, Jan. 2008.

[32] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *MobiHoc*, 2005.

[33] Y. Zhao, S. Vemuri, J. Chen, Y. Chen, H. Zhou, and Z. J. Fu. Exception trigged dos attacks on wireless networks. Technical Report NWU-EECS-08-13, Univ. of Northwestern, Dec. 2008. http://cs.northwestern.edu/∼yzhao/.