Postprint Springer-Verlag. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2013 ;Volum 114. s. 283-301

Detection of Masqueraded Wireless Access Using 802.11 MAC Layer Fingerprints

Christer Idland, Thomas Jelle, and Stig F. Mjølsnes

Department of Telematics
Norwegian University of Science and Technology,
Trondheim
{christer.idland,thomas.jelle,sfm}@item.ntnu.no

Abstract. Many wireless Internet access operators prefer open local area network (WLAN) access because this reduces the need for user assistance for a variety of smaller devices. A 802.11 MAC spoofer masquerades as an authorized user and gains access by using an already whitelisted MAC address. We consider the scenario where the spoofer waits until the authorized user has finished the session, and then uses the still whitelisted MAC address for the network access. We propose and experiment with "implementation fingerprints" that can be used to detect MAC layer spoofing in this setting. We include eight different tests in the detection algorithm, resulting in 2.8 in average Hamming distance of the tests. Eleven different STA devices are tested with promising detection results. No precomputed database of fingerprints is needed.

Keywords: WLAN, 802.11, wireless, media access layer, masquerading, intrusion detection, network forensics, communication fingerprints.

1 Introduction

1.1 The Problem

Many wireless Internet access operators choose to provide a cryptographically unprotected wireless local area network (WLAN) link because this simplifies the user configurations for a variety of smaller devices, makes the wireless association faster, and reduces the cost of the user help-desk. Still, connecting to a wireless local area network for the first time may not be hassle-free for the user, because the setup also depends on user input for authentication, service selection, and payment. The operator can use a so-called *captive portal* for the purpose of user access control. A captive portal responds to any Hypertext Transfer Protocol (HTTP) client request (normally a web browser) with a special user authentication web page. All other uplink packets from the client will be blocked by the portal. The response page will give information about the internet access service, the operator, the access policy and accepted payment services. Once the client submits the proper credentials, then the Medium Access Control (MAC) address of the user's WLAN network interface card (NIC) is whitelisted in the portal and subsequent packets are routed normally.

IEEE distributes and manages the allocation of the MAC addresses on a global basis. The manufacturers of the 802.11 NICs manage the assignment of a unique MAC to each device produced. This MAC identifier is stored in the hardware or firmware of the NIC, but can in many instances be modified by an attacker for the purpose of masquerading as an authorized user, for instance in the simple access control based on checking the MAC address performed by a captive portal. This is often called *MAC spoofing attack*.

Theoretically, this threat of masqueraded attacks does not come as a surprise, because only the user is authenticated by the captive portal, whereas the communicating devices and all their subsequent data communication are left without any authentication at all. The IEEE 802.11 standard provides security mechanisms for establishing a common symmetric authentication key between the client station (STA) and the access point (AP), where each link data frame is protected by a message authentication code that enables verification by the receiver. If a MAC spoofer attacker does not have access to the secret authentication key, then it will become computationally impossible to generate the correct message authentication codes, and the data frames from the spoofer will be rejected by the AP. In practice, the setup of the cryptographic keys will require extra user input, which works against user convenience and operator preferences.

Several techniques have been proposed for detecting a spoofing attack *while* the victim of the spoofing attack is actively connected to a cryptographically unprotected WLAN. It is an open problem whether it is possible to detect a MAC spoofing attack when the victim is no longer connected to the AP. The problem addressed in this paper is how to detect MAC spoofing when only the masquerading NIC is actively accessing the AP. An automatic detection of this type of spoofing attacks requires new algorithms for distinguishing between the authorized user and the masquerading attacker. Our distinguisher algorithms presented here are based on observing the heterogeneity of different implementation characteristics of the 802.11 protocol. We investigate how distinctive features of the various 802.11 implementations create NIC fingerprints, and how these can be used in the detection of MAC spoofing.

1.2 Motivation

The Internet access network *Wireless Trondheim* is a city wide wireless access network, mainly based on the IEEE 802.11a/b/g technologies (Wi-Fi). Currently, the network consists of approximately 500 access points. The geographical coverage is in the Trondheim city center outdoor area of about 1.5 km^2. In addition, the indoor area of all the buildings of the Municipality of Trondheim plus the coverage area of other central buildings. Wireless Trondheim provides Internet access service to about 15.000 unique users each month. Typically, 50% of the client terminals will use the WPA2 Enterprise solutions, while the other half will use the captive portal solution with authentication but with no encryption.

The main goal of Wireless Trondheim is to provide easy wireless Internet access for its users on a wide range of wireless equipment, including simple low-end devices, such as music players and simple mobile phones. Moreover, the wireless network shall provide an arena for testing new services in a real environment (Living Lab). This implies that access control mechanisms must be as simple to operate and use as possible, keeping minimal requirements of the hardware and software of the wireless terminal.

On the other hand, cyber crime activities may be carried out by stealing user names and network addresses (IP, MAC) copied from other terminals. If a criminal act has been carried out and becomes investigated, then we want to make sure that the culprit is found and accused, and not some innocent third person. This implies the need for a strong authentication system, and works against the requirement of easy access for any user device.

Wireless Trondheim's motivation for finding solutions to the identity theft problem is to avoid that innocent users are wrongly accused of serious crime. A possible scenario is if the police or other authority requests information about the identity and activity of users during the process of serious crime investigations, where Wireless Trondheim can only provide possibly incriminating information about customers that in reality are innocent. This can obviously happen if an attacker hijacks the session of a user to hide his identity while doing criminal activity. A successful impersonator could perform criminal activities under another innocents users identity and access. The innocent user will be prosecuted and could eventually be found guilty in serious crime. The prime motivation of Wireless Trondheim for deploying a spoofing detection service is therefore to be able to check whether it is likely that there has been an identify theft before handing over user information to the police or other authorities. The algorithms presented here can also be used in an intrusion detection system (IDS) may block the attack, but this blocking will require reliable digital evidence because the costs of refusing a legitimate user are potentially high, the commercial and reputational risks for the service provider, as well as the denial-of-service and possible false allegations against the subscribed user.

2 Background and Related Work

The Norwegian University of Science and Technology and the company Wireless Trondheim started a research collaboration in 2008 taking on the practical problem of detection of masqueraded wireless access in the 802.11 networks. This research activity has spurred several master projects and theses at Department of Telematics, NTNU. In previous work, techniques have been proposed and tested for detecting MAC spoofing while the victim is active. In particular, we based the detection techniques on MAC sequence numbers and other logical properties of the 802.11 MAC layer [1].

This paper considers the scenario where the spoofer waits until the authorized user has finished the session, and then take advantage of the still whitelisted MAC address for the network access. Many of the results reported in this paper are based on the master thesis work and supervision of Idland [2]. Here we publish the main results, and put them in the wider problem context of operational architecture and practice.

Franklin et al. [4] exploit the fact that the channel scanning algorithm searching for available APs is not explicitly defined in the 802.11 protocol. They develop a method based on statistical analysis of the interframe timing of transmitted probe requests in order to identify a specific driver, and the conclusion is that the majority of wireless drivers do have a distinct fingerprint.

There are important differences in the processing of the Null Data frames in various implementations and we make good use of this observation. Gu et al. [7] create seven rules to identify different behavior regarding the Null Data frames. They focus on the fact that this distinguishing implementation feature may allow an attacker to recognize and determine the location of a client station. Location is in this case limited to an AP, for instance at a coffee shop, at home, or at school. These rules form a basis for the algorithm we present here, but we will use the rules to detect, and not aid attacks.

There exists several commercially available wireless intrusion detection systems (IDSs) that claim to detect and even prevent MAC spoofing attacks. One product in particular, the HSMX from fdXtende [8], uses a captive portal functionality. The manufacturer claims that HSMX will detect MAC address spoofing and prevent hijacking. It turns out that the hijacking prevention is based on an active SSL window technique that might not run on low-end devices. Furthermore, the MAC spoofing detection algorithm is based on not accepting more than one single IP address for a given MAC address, thus it does not defend against attacks where the victim is no longer online.

3 The Threat Model

3.1 The Wireless Access Network

Wireless Trondheim manages an open 802.11 access network where 802.11a/b/g are supported. The Wi-Fi Multimedia feature is enabled. Figure 1 presents a simplified overview of the network structure, depicting the relevant components for our purposes here. The AP functionality is split into two separate devices, which are the Lightweight Access Point (LAP) and the Wireless LAN Controller (WLC). One WLC entity can control several LAPs entities. The captive portal functionality is integrated in an Internet Gateway entity (Nomadix). The IDS server can receive the fingerprinting parameter values possibly from the WLC, or make the acquisition itself by eavesdropping on the wireless link directly. A whitelisted MAC address in the captive portal will remain whitelisted for up to 60 minutes after traffic has ceased. Obviously, a shorter time to white list flush will reduce the user's convenience by having to repeat a log-in after a break, and reduce the time available for the attacker.

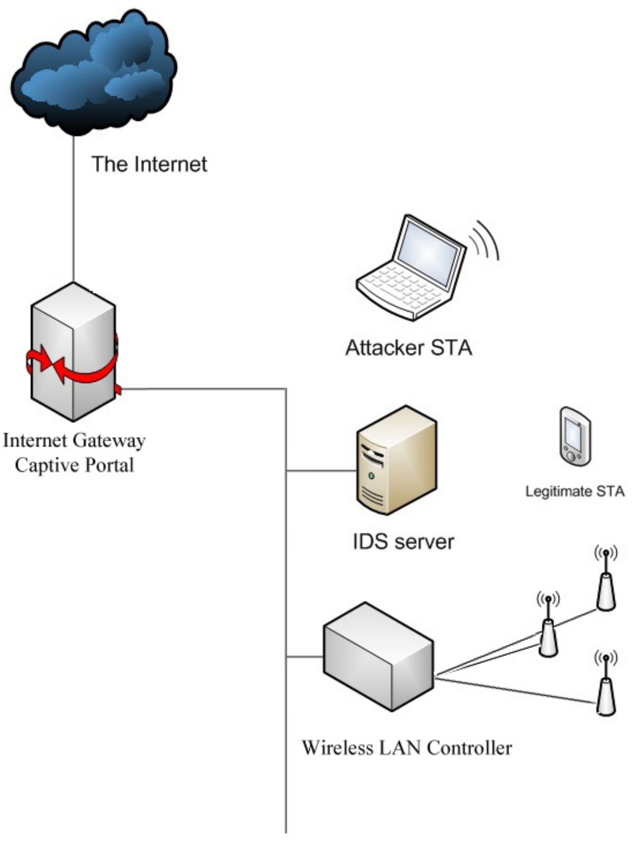


Fig. 1. The main network elements of the wireless access network. Several lightweight access points (LAP) are connected to a wireless LAN controller (WLC). The IDS server can receive fingerprinting properties from the WLC, or monitor the wireless links directly.

3.2 MAC Spoofing

An easy way to fool an access control system based on whitelisting of MAC addresses is by performing a MAC spoofing attack. The theory is that the attacker masquerades as a legitimate client, that is, a client that already has his MAC address whitelisted. Thereby the attacker gains access to the network. A suitable MAC address is easily obtained through eavesdropping of the victim's wireless communications.

Several methods for detecting attacks based on simultaneous access exists. The focus of this paper is on the MAC spoofing attack where the attacker and the victim do *not* need to be connected simultaneously. The main problem focus is on the wait-for-availability attack as it is the easiest to perform from the attacker's viewpoint. One can also argue that it is the most difficult attack to detect because the attacker does not force the legitimate client off the network. An attacker can force a victim off the network connection by sending a `deauthenticate` or `channel switch` frame to the client [3]. These MAC management frames can easily be observed by an IDS, ergo it can easily be detected. Our attack scenario is where the attacker waits until there has not been any frame coming from the victim's station (STA) for a significant period of time. Then, inferring that the victim has left, the attacker takes on the MAC address and tries to continue using the wireless access network.

4 Algorithms

4.1 Distinctive Features

The behavior of MAC layer STAs differs in many ways due to implementation differences of the 802.11 protocol. Most of these differences exist because the standard is not explicit, and therefore open to alternative interpretations. Some distinctive features are a result of different options and capabilities of the specific NIC in question. These distinctive features, called fingerprinting properties, are based on the rules for Null Data frame behavior found in Ref. [7]. We augment their list with several other distinctive features found during our research. Our list is presented in Table 1.

4.2 The Tests

The pseudocode for all algorithms described can be found in Appendix and Ref [2].

Test 1, PS-Poll Test 1 is based on the reported feature that some NICs use Null Data frames and others use PS-poll frames for power management. That is somewhat imprecise in relation to what we observed in the experiments. All STAs in the experiment used Null Data for changing power mode, but one STA used PS-Poll frames when the AP had buffered frames to send.

Therefore, the test is whether a STA use PS-Poll or not. If a STA use PS-Poll then the fingerprinting algorithm should observe a PS-Poll frame from the STA after the AP has announced that it has buffered frames for it. This PS-Poll frame should be observed within the `Listen interval` of that STA.

If the frame is a Beacon frame and the bit in the TIM corresponding to the STA in question is set then the `beacon_count` is incremented. If the `beacon_count` exceeds the `Listen interval` then the algorithm concludes that PS-Poll is not in use and returns suspect attack if it previously was in use. When a PS-Poll

Table 1. Fingerprinting properties and their possible values for 802.11 network interface implementations [2].

Fingerprinting Property	Possible Values
PS-Poll	True / False
Keep Alive	True / False
Null before Probe	True / False
Mode changing Null Data	True / False
Fixed Interval	True / False
Null Data Type	Regular/QoS including TID
Duration Calculation	Pairs of (rate, duration) for each rate
Association Request Duration	$\{0...32767\}$
Listen Interval	$\{0...256\}$
Supported Rates	Set of up to eight integers $\in \{2...127\}$
Extended Supported Rates	Set of up to 255 integers $\in \{2...127\}$
QoS Capability	Present / Not Present
Vendor Specific	Type of vendor specific element

frame is observed the `beacon_count` is reset and the algorithm naturally concludes that PS-Poll is in use, if that was not the case before the algorithm will return "suspect attack."

Test 2, Keep Alive This algorithm tests whether a STA sends a Null Data frame if it has been idle for 10 seconds in order to keep the session alive. The timestamps are in milliseconds and thus testing for exactly 10 seconds would yield very few hits, therefore the test is implemented with a buffer, currently set to 0.15 seconds.

First the `time_delta` is calculated from the timestamp of the previous frame and the current frame. Then the algorithm checks if the `time_delta` is within the range of 10 seconds $\pm$ the buffer of 0.15 second and in addition if the frame is a Null Data frame. If both of these conditions are true then the algorithm concludes that keep alive is in use and returns suspect attack if keep alive was not in use prior to this frame.

If the first if-conditions fail then a new if-statement checks if the `time_delta` is larger than 10.15 seconds (10 + buffer). If true then this indicates that the STA is not using keep alive, and thus the algorithm concludes so and returns suspect attack.

Test 3, Null before Probe The rationale behind this test is that some STA sends a Null Data frame and enters PS mode before starting the channel scanning with Probe Request frames while other STAs do not do this.

First a set size and a minimum limit are defined. The set size is the number of probe request bursts that will be observed before any conclusion is made. The minimum limit is a number $\in [0, 1]$ representing the percentage of probe request bursts where the STA first sends a Null Data frame required in order for the algorithm to conclude that null before probe is in use.

If the current frame is a Probe Request frame then the algorithm continues. If the previous frame was a Null Data frame then null before probe is in use and the algorithm increments the `using_count` as well as the `total_count`. On the other hand, if the previous frame was not a Null Data frame or a Probe Request frame then null before probe is not in use and the algorithm only increments the `total_count`.

The last part of the algorithm checks if the percentage of times null before probe was in use is over the minimum threshold in order to conclude if it in fact is in use. Suspect attack is returned whenever the current conclusion differs from the previous conclusion.

Test 4, Mode changing Null Data Test 4 checks if the STA, when it has data frames to send, uses a Null Data frame to change mode or if it directly sends a regular data frame when changing power mode. Note that some STAs always use mode changing Null Data except when they have been in PS mode for a duration equal to their own `Listen interval`, this is therefore included in the test.

First a set size and a minimum limit are defined. The set size is the number of power mode changes that will be observed before any conclusion is made. The minimum limit is a number $\in [0, 1]$ representing the percentage of power mode changes that must be done by using Null Data frames in order for the algorithm to conclude that mode changing Null Data is in use.

The first if-statement checks whether the STA has been in PS mode longer than its `Listen interval`, and if so changes the recorded power mode of the STA to AM.

The next if-statement checks if the `chk_nxt_pkt` variable is set. The first time the algorithm is executed this is not the case and the algorithm continues to check if the STA is in PS mode. If the STA is in PS mode (before the current frame) the algorithm checks if the current frame is a Null Data frame that changes the power mode to AM. If this is the case, the `chk_nxt_pkt` variable is set. If it is not a Null Data frame with `pwr_mgt` bit $== 0$, but a regular data frame with `pwr_mgt` bit $== 0$, then the algorithm interprets this as the STA is not using null before probe and therefore increments the `total_count` without incrementing the `using_count`.

The reason for having the `chk_nxt_pkt` variable is that we are only interested in the power mode changes made when the next packet is a data packet. So, when the next packet is processed in the algorithm the `chk_nxt_pkt` variable is set and the algorithm checks if the current packet is a regular data packet. If this is the case the `using_count` is incremented as well as the `total_count`.

The last part of the algorithm works the same way as in Test 3, by returning suspect attack if the overall conclusion has changed since last time.

Test 5, Fixed Interval This test checks the duration between Null Data frames with different values in the `Power Management` bit. If the STA is using mode changing Null Data then this interval would translate into the time the STA was in PS mode. The idea behind this test is that some STA stays in PS mode for a *fixed* interval, regardless of which data to transmit.

First a new Null Data frame is detected. If the `pwr_mgt` bit of the previous Null Data frame was 1 (PS) and the value in the current frame is 0 (AM) then this pair of Null Data frames will be examined further. If the measured time interval value between these two frames is within a preset normality range, then the counter for successful detection of pairs (`pair_ok_count`) is incremented. The `pair_total_count` is incremented regardless of the result of the outcome of this comparison. In the experiments, the time interval average was computed from the first 10 time intervals of Null Data frames measured, and the normality range was heuristically set to $\pm 20\%$ of this average. The last part of the algorithm computes the fraction of `set_size` Null Data frame pairs where the time interval falls within the normality range. If this fraction is greater than a threshold parameter then the algorithm concludes that the STA does use a fixed time interval for the Null Data frames. The algorithm returns "suspect attack" if the result of the previous test run was the opposite of this run.

Test 6, Null Data Type Test 6 is based on observations regarding Null Data behavior. Recall that the 802.11 standard has two types of Null Data frames; the regular one and the QoS Null Data frame. It turns out that in a network where QoS is enabled (as it is in Wireless Trondheim) some STAs use the QoS Null Data while others do not use it. Amongst those who use the QoS enabled there is differences in which QoS priority (TID class) they utilize. These implementation differences make Test 6 viable.

The algorithm for Test 6 is relative simple compared to the other test algorithms. Nevertheless, it identifies a viable fingerprinting property. The algorithm basically identifies the type of Null Data frame in use and its priority class (TID) if it was a QoS frame. The algorithm then returns suspect attack if the identified frame differs from the previous identified Null Data frame.

Test 7, Duration Calculation Test 7 was motivated by the paper by Gopinath et al. [5]. The theory is that Null Data frames have the exact same size, and thus should have the same duration when the data rate is the same. A difference in the `Duration/ID` field would indicate different implementations in the calculation algorithm that again indicates two different STA.

It basically works by recording the duration taken from the `Duration/ID` field in the Null Data frames for each data rate used. Recall that the data rate is available in the radiotap header. If an inconsistency is found then the algorithm outputs suspect attack.

Test 8, Association Request Vendor specific extensions as a fingerprinting source is mentioned in the paper by Gopinath et al. [5]. This was the motivation to further investigate the Association Request frame looking for possible sources for fingerprinting. Several potential fields were identified and they are: Duration/ID, Listen interval, Supported Rates, Extended Supported Rates, QoS Capability and Vendor Specific.

When the input frame is an Association Request frame, the relevant fields mentioned above, called implicit identifiers, are recorded. If an Association Request has been recorded for this MAC address before a check on each individual field is done and suspect attack is returned in case of any inconsistencies.

4.3 Creating a Compound Fingerprint

In order to avoid a high false reject rate when using fingerprinting it is important to rely on several different properties and tests that can flag suspicious behavior. The fingerprints created in our experiments are a composition of properties determined by the eight tests explained above. The first six tests result in one fingerprinting property each, while to two latter results in several properties. Each of the properties from the two latter tests can be used to detect suspicious behavior in their own right. See Table 1 for an exhaustive list of fingerprinting properties and their possible values that make up the compound fingerprint. For a more in-depth explanation of the possible values and their usage consult the 802.11 standard [6].

Our MAC spoof detection algorithm comprises a combination of each of the eight tests described above, and some additional logic. The logic to determine whether we are dealing with an attack based on the output from the tests has not been described. We want to do fingerprinting on the fly and not necessarily generate a complete fingerprint. The question of how many properties that are needed, and a selection of optimal parameters in order to achieve acceptable low false positive and negative rates is open for further work. Some of the tests might prove to be sufficient by itself, while others require at least one other test in combination in order to conclude attack with a high probability. The experiments and the following results will shed some light on these questions.

5 Experiments and Results

5.1 The Terminal Equipment Test

We implemented the fingerprinting algorithm in Perl in order to test the different distinctive features of the STAs. Table 2 shows an overview of the STAs that were used in the experiments. The list includes laptops, smartphones and music players, in addition to a typical attacker setup (laptop with Backtrack 5), and are all included in the test.

5.2 Test Scenarios

Scenario 1. General Usage This scenario is constructed to test a browsing behavior. Two behavioral patterns were considered important to include when creating this scenario. First, the STA should continuously generate data traffic

Table 2. Overview of the STA devices used in the experiments.

No.	Name and Model	OS	NIC	Browser
S-1	Dell XPS	Windows 7	Intel Wi-Fi 1000bgn	IE 8
S-2	Lenovo S10-3S	Windows 7	Broadcom 802.11n	Opera 11
S-3	Acer Aspire 5745G	Windows 7	Broadcom 802.11n	IE 8
S-4	Dell Inspiron 9400	Windows 7	Intel P/W 3945abg	Chrome 11
S-5	iPhone 1. gen.	iOS 3.0.1	Not available	Safari
S-6	iPhone 4. gen.	iOS 4.3	Not available	Safari
S-7	Dell Latitude D610	Windows 7	Intel P/W 2915abg	IE 8
S-8	Acer Aspire 5670	Win. XP SP3	Intel P/W 3945abg	Firefox 3.6
S-9	iPod touch 1. gen	iOS 3.1.3	Not available	Opera Mini 5
S-10	HTC Hero	Android 2.2.1	Not available	"Internet"
S-11	Dell Inspiron 9400	Backtrack 5	Intel P/W 3945abg	Firefox 4

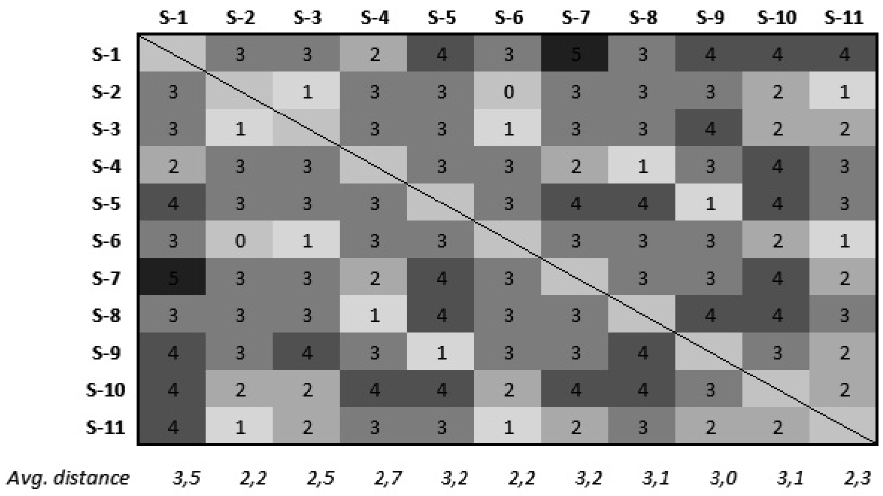
for a period of time, resulting in little or no idle time. Second, the STA should have longer periods where there is no traffic to send. The rationale is that this is a realistic usage pattern, such as browsing the web, start reading or watching something, then continue the web browsing. A scenario including these patterns should also be able to elicit different power management and Null Data behavior. All STAs of Table 2 were tested in this scenario.

Scenario 2. Wait-for-availability Attack Here the user is operating for approximately eight minutes, with some periods of high traffic, and other periods with little or no traffic, similar to the behavior of Scenario 1. Then the user logs off and the attacker spoofs the user's MAC address and performs the wait-for-availability attack. One pair of STA devices were randomly chosen, S-5 and S-7, and tested in this scenario.

Scenario 3. Concurrent Usage This scenario is similar to Scenario 2. The main difference is that now there are five additional STAs connected to the LAP, all generating concurrent traffic. In other words, a total of seven STAs are connected and generating traffic. The scenario is also shortened down to a total of eight minutes (four minutes before the attack and four minutes after). This fact certainly makes it harder for the algorithm to determine the fingerprints as it will be less communication data available for analysis. The question is whether four minutes of monitoring in an active network is enough, or if longer monitoring period, ssuch as the ones in Scenario 1 and 2, are required.

5.3 Results

Each of the 7 first distinctive features corresponds to one test, the next 4 features are gathered in a single test because they all depend on the association request frame. That leaves us with 8 different test/ distinctive features. Figure 2 shows the hamming distance between any two of the STAs used in the experiments, note that the maximal distance of 8 could be obtained if two STAs differed on every single test.



	S-1	S-2	S-3	S-4	S-5	S-6	S-7	S-8	S-9	S-10	S-11
S-1		3	3	2	4	3	5	3	4	4	4
S-2	3		1	3	3	0	3	3	3	2	1
S-3	3	1		3	3	1	3	3	4	2	2
S-4	2	3	3		3	3	2	1	3	4	3
S-5	4	3	3	3		3	4	4	1	4	3
S-6	3	0	1	3	3		3	3	3	2	1
S-7	5	3	3	2	4	3		3	3	4	2
S-8	3	3	3	1	4	3	3		4	4	3
S-9	4	3	4	3	1	3	3	4		3	2
S-10	4	2	2	4	4	2	4	4	3		2
S-11	4	1	2	3	3	1	2	3	2	2	
Avg. distance	*3,5*	*2,2*	*2,5*	*2,7*	*3,2*	*2,2*	*3,2*	*3,1*	*3,0*	*3,1*	*2,3*

Fig. 2. Hamming Distance for the fingerprints in Scenario 1 [2].

Scenario 1 The results from each of the eight tests performed are presented in Table 3 and Table 4. The first column presents the different fingerprinting properties, each of the other columns represents one STA. The first five fingerprinting features are either present (T) or not (F). Algorithms 1-5 determine these values.

Scenario 2 The eight distinctive features or fingerprints determined for each STA were identical to the ones determined in Scenario 1, with the exception of "Fixed Interval" for S-7. The fingerprints from this scenario are shown in Table 5.

Scenario 3 In both Scenario 2 and 3 the attacker's Authentication Request frame was observed, making it easy to identify exactly when the attack occurred. The fingerprints from this scenario can be seen in Table 6.

6 Discussion and Conclusion

We have identified and experimented with some communication fingerprints of the IEEE 802.11 MAC layer that may serve to distinguish user stations. Our

Table 3. Results from Scenario 1 on tests 1-8 for S-1 to S-6. Undetermined entries are shown as "—".

Fingerprinting Property	S-1	S-2	S-3	S-4	S-5	S-6
PS-Poll	F	F	F	F	F	F
Keep Alive	**T**	F	F	—	F	F
Null before Probe	**T**	—	**T**	**T**	**T**	—
Mode changing Null	F	F	F	**T**	F	F
Fixed Interval	F	F	F	F	F	F
Null Data Type	QoS0	Reg.	Reg.	QoS0	QoS7	Regular
Duration Calculation	314	44	44	314	258	44
Ass. Req. Duration	60	314	213	314	314	314
Listen Interval	10	10	10	10	10	10
Supported Rates	s1	s2	s3	s3	s3	s2
Ext. Sup. Rates	e1	e2	e1	e1	e1	e2
QoS Capability	F	F	F	F	F	F
Vendor Specific	v1-v4	v1,v5	v1-v4	v1	v1	v1,v5

Table 4. Results from Scenario 1 on tests 1-8 for S-7 to S-11. Undetermined entries are shown as "—".

Fingerprinting Property	S-7	S-8	S-9	S-10	S-11
PS-Poll	F	F	F	**T**	F
Keep Alive	F	F	F	F	F
Null before Probe	**T**	**T**	F	—	F
Mode changing Null	**T**	**T**	F	F	—
Fixed Interval	**T**	F	—	F	—
Null Data Type	Reg.	QoS0	QoS7	Reg.	Regular
Duration Calculation	44,314	44,314	258	44,213,223,258	44
Ass. Req. Duration	314	213	314	258	314
Listen Interval	10	10	10	3	5
Supported Rates	s3	s1	s3	s4	s1
Ext. Sup. Rates	e1	e1	e1	e3	e1
QoS Capability	F	F	F	**T**	F
Vendor Specific	v1	v1-v4	v1	v1	v1

algorithms are able to detect spoofing attacks where the victim is not connected simultaneously with the attacker, something commercial IDS cannot do today. We have shown the feasibility of passively measuring the MAC layer fingerprints without specialized equipment, and that this can be done efficiently under realistic network access conditions. No precomputed database of fingerprints is necessary. The test data were acquired under realistic access scenario setups of 8-10 minutes, using 11 different devices. The communication fingerprints exhibited an average Hamming distance of 2.82. Even in the case of severely reduced communication data available (scenario 3), the tests show that the proposed al-

Table 5. Results from Scenario 2 on tests 1-8 for S-5 and S-7.

Fingerprinting Property	S-5	S-7
PS-Poll	F	F
Keep Alive	F	F
Null before Probe	**T**	**T**
Mode changing Null	F	**T**
Fixed Interval	F	F
Null Data Type	QoS7	Regular
Duration Calculation	258	44,223,258,314
Ass. Req. Duration	314	314
Listen Interval	10	10
Supported Rates	s3	s3
Ext. Sup. Rates	e1	e1
QoS Capability	F	F
Vendor Specific	v1	v1

Table 6. Results from Scenario 3 on tests 1-8 for S-5 and S-7.

Fingerprinting Property	S-5	S-7
PS-Poll	F	F
Keep Alive	F	F
Null before Probe	—	**T**
Mode changing Null	—	—
Fixed Interval	—	—
Null Data Type	QoS7	Regular
Duration Calculation	258	314
Ass. Req. Duration	314	314
Listen Interval	10	10
Supported Rates	s3	s3
Ext. Sup. Rates	e1	e1
QoS Capability	F	F
Vendor Specific	v1	v1

gorithms are still able to distinguish between devices. The level of attacker skills required to avoid detection and evidence of session hijacking attacks in 802.11 can be raised considerably by using techniques presented here.

It remains to find out how to fine tune the selection of parameter values in the algorithms to gain optimal detection efficiency. Also, some of the fingerprints used are not easily altered as they are hard-coded in the firmware and drivers of the devices, while other fingerprints might be easier to alter or conceal, so assessing the difficulty of modifying or concealing a fingerprint are future work.

References

1. Eirik Holgernes. Detecting Identity Thefts in Open 802.11e Enabled Wireless Networks. Masters thesis, Department of Telematics, NTNU, June 2010. 109 pages. `http://daim.idi.ntnu.no/masteroppgave?id=5476`
2. Christer Idland. Detecting MAC Spoofing Attacks in 802.11 Networks through Fingerprinting on the MAC Layer. Masters thesis, Department of Telematics, NTNU, June 2011. 96 pages. `http://daim.idi.ntnu.no/masteroppgave?id=6260`
3. Martin Eian and Stig F. Mjølsnes. The modeling and comparison of wireless network denial of service attacks. In *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*, 2011, ACM.
4. Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *Proceedings of the 15th conference on USENIX Security Symposium.* Volume 15, 2006.
5. K. N. Gopinath, Pravin Bhagwat, and K. Gopinath. An empirical analysis of heterogeneity in IEEE 802.11 MAC protocol implementations and its implications. In *Proceedings of the 1st international workshop on wireless network testbeds, experimental evaluation & characterization*, 2006.
6. IEEE. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems— Local and Metropolitan Area Networks—Specific Requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report, IEEE, 2007.
7. Wenjun Gu, Zhimin Yang, Can Que, Dong Xuan, and Weijia Jia. On Security Vulnerabilities of Null Data Frames in IEEE 802.11 based WLANs. In *Proceedings of The 28th International Conference on Distributed Computing Systems*, pp. 28–35, 2008, IEEE.
8. fdXtended. HSMX - Internet Access Platform, Datasheet. Retrieved June 9, 2011 from `http://www.fdxtended.com/datasheets/HSMX-datasheet.pdf`

Appendix: The Algorithms

```
Input: $frame$

if $frame ==$ Beacon frame and bit in $TIM$ is set then
   $beacon\_count{+}{+}$
   if $beacon\_count > listen\_interval$ then
      $use\_PSPoll =$ false
      if $use\_PSPoll$ was true then
         return suspect attack
      end if
   end if
end if

if $frame ==$ PS-Poll frame then
   $beacon\_count = 0$
   $use\_PSPoll =$ true
   if $use\_PSPoll$ was false then
      return suspect attack
   end if
end if
```

Algorithm 1: Test 1, PS-Poll

```
Input: $frame$

  $time\_delta$ = timestamp previous frame - timestamp current frame
  buffer is 0.15 sec
  $within\_buffer = 9.85 < time\_delta < 10.15$

  if $within\_buffer$ and $frame ==$ Null Data then
    $use\_keep\_alive$ = true
    if $use\_keep\_alive$ was false then
      return suspect attack
    end if

  else if $time\_delta > 10.15$ then
    $use\_keep\_alive$ = false
    if $use\_keep\_alive$ was true then
      return suspect attack
    end if
  end if
```

Algorithm 2: Test 2, Keep Alive

```
Input: $frame$

  $set\_size = 5$
  $min\_limit = 0.80$

  if $frame ==$ Probe Request then

    if previous frame was Null Data then
      $using\_count$++
      $total\_count$++
    else if previous frame was not Probe Request then
      $total\_count$++
    end if
  end if

  if $total\_count == set\_size$ then
    $use\_null\_before\_probe = using\_count > set\_size \times min\_limit$
    if $use\_null\_before\_probe$ changed value then
      return suspect attack
    end if
    $using\_count = 0$
    $total\_count = 0$
  end if
```

Algorithm 3: Test 3, Null before Probe

```
Input: $frame$

  $set\_size = 30$
  $min\_limit = 0.9$

  if $frame$ == Beacon frame then
    if number of Beacons since data > $listen\_interval$ then
      $pwr\_mode$ = AM
    end if
  end if

  if $chk\_nxt\_pkt$ == true then
    if frame type is data and $fame \neq$ Null Data then
      $using\_count{+}{+}$
      $total\_count{+}{+}$
    end if
    $chk\_nxt\_pkt$ = false
  else
    if $pwr\_mode$ == PS then
      if $frame$ == Null Data and pwr_mgt bit == 0 then
        $chk\_nxt\_pkt$ = true
      else if frame is DATA and pwr_mgt bit == 0 then
        $total\_count{+}{+}$
      end if
    end if
  end if

  pwr_mode = pwr_mgt bit (1 = PS, 0 = AM)

  if $total\_count == set\_size$ then
    $use\_mode\_chng\_null = using\_count > set\_size \times min\_limit$
    if $use\_mode\_chng\_null$ changed value then
      return suspect attack
    end if
    $using\_count = 0$
    $total\_count = 0$
  end if
```

Algorithm 4: Test 4, Mode changing Null Data

```
Input: $frame$

  $set\_size = 50$
  $threshold = 0.8$

  if $frame$ == Null Data then
    $time\_delta$ = time elapsed since previous Null Data frame

    if $pwr\_mgt\_previous == PS$ and $pwr\_mgt\_current == AM$ then
      if $average$ exists then
        if $time\_delta$ is within $range(average)$ then
          $pair\_ok\_count{+}{+}$
        end if
        $pair\_total\_count{+}{+}$
      else
        calculate $average$ from first 10 pairs
      end if
    end if

    if $pair\_total\_count == set\_size$ then
      $use\_fixed\_interval = (pair\_ok\_count/set\_size) \geq threshold$
      if $use\_fixed\_interval$ changed value then
        return suspect attack
      end if
      $pair\_ok\_count = 0$
      $pair\_total\_count = 0$
    end if
  end if
```

Algorithm 5: Test 5, Fixed Interval

```
Input: $frame$

  if $frame$ == Null Data then
    categorize frame as QoS or regular
    if $frame$ was QoS type then
      identify the priority class (TID)
    end if
    if $frame$ differs from the previous Null Data frame then
      return suspect attack
    end if
  end if
```

Algorithm 6: Test 6, Null Data Type

```
Input: $frame$

  if $frame$ == Null Data then
    get the duration value from the Duration/ID field
    get the data rate from the radiotap header

    compare duration for given data rate with previous value

    if differences in duration value for same data rate then
      return suspect attack
    end if
  end if
```

Algorithm 7: Test 7, Duration Calculation

```
Input: $frame$

  if $frame$ == Association Request then
    record implicit identifiers
    if Ass. Req. for same MAC address is recorded before then
      compare implicit identifiers from current and previous frame
      if inconsistencies in implicit identifiers then
        return suspect attack
      end if
    end if
  end if
```

Algorithm 8: Test 8, Association Request

```
Input: $frame$

  run Test 1, PS-Poll
  run Test 2, Keep Alive
  run Test 3, Null before Probe
  run Test 4, Mode changing Null Data
  run Test 5, Fixed Interval
  run Test 6, Null Data Type
  run Test 7, Duration Calculation
  run Test 8, Association Request

  evaluate outputs from tests 1-8

  return attack / no attack based on evaluation
```

Algorithm 9: The Fingerprinting Algorithm.