A Sophisticated Solution for Revealing Attacks on Wireless LAN

René Neumerkel and Stephan Groß

Technische Universität Dresden
Department of Computer Science
Institute for System Architecture
D-01062 Dresden, Germany
{rene.neumerkel, stephan.gross}@tu-dresden.de

Abstract. The development of the WPA and IEEE 802.11i standards have vastly improved the security of common wireless LAN setups. However, many installations still use the broken WEP protocol or even run with no security settings enabled. Furthermore, several threats are only partially addressed by the new security standards, e.g. rogue access points or denial of service. Specialised wireless intrusion detection systems are promising means to protect wireless networks against these threats. They can further improve the reliability and security of these networks. In our contribution we present such a wireless IDS to reveal common attacks on wireless LAN. We describe the development and evaluation of our prototype solution that seamlessly integrates with approaches for traditional wired networks.

1 Introduction

During the last decade we were witnessing the breakthrough of wireless communication techniques. Today, the vision of seamless Internet access everywhere at anytime has almost become true, e.g. by the growing number of wireless access points in public and private places. Unfortunately, security has not been a design goal in the first place of the underlying technical foundations. Thus, many of the deployed techniques suffer from severe security drawbacks. For example, the Wired Equivalent Privacy (WEP) protocol has once been the standard security mechanism in IEEE 802.11 wireless LAN. In 2001 Fluhrer et al. described a way to utilise a design flaw in WEP's usage of the key scheduling algorithm RC4 to break the encryption [1]. Since then, several free available software tools empowered even unskilled users to penetrate their neighbour's wireless LAN. At least with the publication of statistical cryptanalysis attacks in 2004 like the KoreK [2] or the chopchop attack [3], the WEP protocol must be considered definitely defeated. These attacks no longer depend on millions of captured packets to crack a WEP key but combine several techniques like traffic injection or predefined password dictionaries to derive a key within several minutes. Today, the usage of recent security protocols like WPA [4] or WPA2 aka IEEE 802.11i [5] are indispensable. But even with these measures enabled, a wireless LAN can still be plagued by security issues like Denial of Service attacks or misconfigured access points. To overcome these threats one should consider an intrusion detection solution to monitor the current status

S. Fischer-Hübner et al. (Eds.): TrustBus 2006, LNCS 4083, pp. 223–232, 2006.

of a wireless network. As common IDS are only considering ISO/OSI layer 3 upwards they are not able to detect specific wireless threats taking place on the MAC layer. Furthermore, unlike wired networks, the special characteristics of mobile networks pose a number of new challenges to the security design we cannot solve using tradional approaches [6]. Thus, we need specialised Intrusion Detection Systems for wireless LAN.

There are several commercial as well as open source solutions available for that purpose. Examples for commercial competitors are Airdefense[1], NetworkChemistry[2] or Internet Security Systems[3]. Prominent open source projects are Snort-Wireless[4], Kismet[5] and WIDZ[6]. However, in our opinion all available solutions suffer from one or more of the following limitations:

- They require special infrastructural means. Thus, they can only be applied in a fixed environment with a centralised network structure (e.g. Airdefense Guard).
- They unnecessarily depend on specialized hardware even if the implemented features can be solved in software (e.g. NetworkChemistry RFprotect).
- They are not integrated with common Intrusion Detection and network management solutions (e.g. WIDZ).
- They do not allow the addition of user-defined detection strategies as they solely rely on built-in attack signatures (e.g. Kismet)
- They require changes in source code when adding more complex detection strategies that involve more than one packet (e.g. SnortWireless)

In this paper we present a solution to overcome these drawbacks. The remainder of this paper is organised as follows: First, we summarize related work in section 2. Section 3 deals with the design and implementation of our prototype wireless intrusion detection system. We utilise the Bro IDS for this purpose which was originally designed for monitoring high-speed wired networks and which we enhanced with several specific means to observe wireless networks. Section 4 is dedicated to a detailed report of our prototype's validation. Finally, we come up with some final remarks on our results and on directions for further investigations.

2 Related Work

We have already mentioned commercial and open source solutions in the previous chapter. In addition, there exist several contributions from the academic community from which we only mention some selected works. The need of reconsidering traditional network protection for mobile environments is addressed in [7]. [8] already presents a security architecture for mobile ad-hoc networks based on anomaly detection. However, this work tackles the problem from a more academic point of view and does not

[1] http://www.airdefense.net/
[2] http://www.networkchemistry.com/
[3] http://www.iss.net/
[4] http://snort-wireless.org/
[5] http://www.kismetwireless.net
[6] http://freshmeat.net/projects/widz/

address issues appearing in real-world scenarios. In [9] Lim et al. describe a prototype implementation of a wireless intrusion detection system. They modified an off the shelf wireless access point in order to detect common wireless attacks. In [10] they further enhance their prototype implementation with an active countermeasure capability and demonstrate the usefullness of their approach by a case study. Their approach is quite similiar to ours. However, they fully concentrate on the wireless scenario and do not consider the integration with general intrusion detection solutions as we do. The Distributed Wireless Security Auditor (DWSA) presented in [11] works toward finding unauthorized wireless access points in large-scale wireless environments. The system utilises a centralised architecture and is, thus, only applicable in a fixed environment. The most interesting feature of the DWSA might be its ability to track down a potential adversary based on three-dimensional trilateration.

3 Developing a Wireless Intrusion Detection System

The starting point for our research work has been an intensive literature and Internet research on commonly known threats to IEEE 802.11 wireless networks. Today, there exist a multitude of basic techniques and ready to use exploits for compromising a wireless LAN. However, these techiques can all be traced back to three basic principles: violation of confidentiality, integrity or availibility [12]. For the purpose of categorizing the acquired threats we used attack trees, a semi-formal method to analyse and document the security of a given system with respect to variying attacks [13]. Due to the lack of space we refer to [14] and [15] for a more or less complete description of commonly known threats to wireless LAN.

For developing our own wireless intrusion detection system we utilise an existing wired IDS called Bro[7] and added the necessary functionality for monitoring wireless networks. Thus, we are not only able to overcome the above mentioned missing integration of wireless solutions with conventional intrusion detection and network management systems but also improve the functional range of our system as we can fall back on the full range of mechanisms already implemented in the base system. In this section we first give a brief overview to the Bro IDS before we describe our modifications.

3.1 Introducing the Bro IDS

Bro is a Unix-based open source Network Intrusion Detection System (NIDS) designed for monitoring high-speed, high-volume wired networks. The system detects intrusions in real-time by passively monitoring network traffic and comparing it to a set of user-defined rules describing security-related events such as the occurrence of known attacks or unusual network activities [16].

Bro uses *libpcap* to capture network traffic. The received packet stream is processed by *Analyzers* operating on protocols at OSI layer 3 and above. Analyzers exist, for example, for ICMP, TCP and HTTP. Besides that, Bro also uses a *Signature Engine* which matches the incoming packet stream against patterns of known attacks, thus realising basic misuse-based intrusion detection. In both cases built-in events are generated

[7] `http://www.bro-ids.org`

whenever interesting network activity (e.g. a failed TCP connection attempt) occurs. Generated events are processed by an *Event Engine* which is responsible for calling the associated *Event Handlers* located in *Policy Scripts* outside of Bro's core. Event handlers can execute various actions like logging, real-time notification or generation of user-defined events. For this purpose, Bro provides a special scripting language which allows end users to define their own policy scripts. Besides that, Bro comes with a pre-defined set of policy scripts that can be modified to reflect a site's actual security policy.

By putting the event handlers outside of Bro's core, a clear separation of built-in detection mechanisms from site-specific interpretation of events is achieved. This becomes especially useful in mobile environments with their frequently changing network topologies as it simplifies the adaptation to new conditions.

3.2 Enhancing Bro for the Wireless World

As mentioned before, Bro has been specifically designed for wired intrusion detection and thus does not provide any means for monitoring wireless networks. Therefore, several modifications to Bro's architecture were necessary which we will describe in the following.

Capturing of 802.11 frames. In order to monitor wireless traffic, Bro needs to access raw 802.11 frames. As mentioned before, Bro relies on libpcap to capture network traffic. Since current versions of libpcap already support the capturing of raw 802.11 frames, only minor changes were necessary to make Bro recognize the data link types identifying wireless links. However, capturing wireless frames alone is not enough. When monitoring 802.11 traffic it is also important to know on what channel the capture device is listening on. Therefore, we added the necessary code to determine the current channel of each active wireless device at the start-up of Bro. It is also possible to determine the current channel on a per-packet basis. For this purpose, some drivers include an optional pre-header before the regular 802.11 headers.

At this point it seems important to mention that the modified Bro system is still capable of monitoring wired networks. Furthermore, it is also possible to monitor both wired as well as wireless networks simultaneously.

Processing of 802.11 frames. Once our prototype was able to capture raw 802.11 frames, further functionality was added to allow processing of captured frames. We currently focus on the analysis of 802.11 management frames since the majority of wireless network attacks is based on this frame type. The basic steps are shown in figure 1 and include extracting all available information from header fields and frame body. Our implementation currently supports only field types defined by the original IEEE standard and ignores any unknown field types including vendor-specific information elements. Based on the information extracted from frame header and body, built-in events are generated which yet had to be defined (see next section). In addition to analyzing management frames, preliminary processing of data frames and control frames is already being done and could be further implemented when necessary.

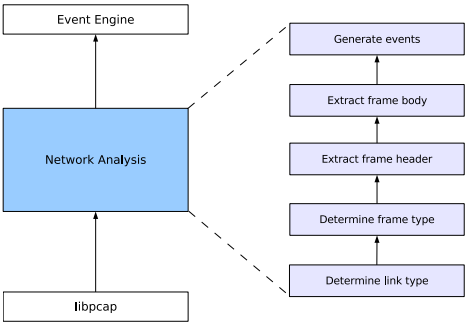

Fig. 1. Processing of 802.11 frames

Note that our current solution does not handle fragmented frames, as this was not a requirement for our prototype. However, we believe that the missing functionality could easily be added anytime if needed.

Definition of 802.11-Specific events. Once we added the functionality to process 802.11 frames, we needed to define the events to be generated based on the network traffic. We found that there is more than one possible approach when defining new events. We eventually decided to implement a straight-forward approach where we simply map each management frame sub-type to a corresponding event, for example *ProbeRequestReceived*. This way, traffic analysis can be done completely at script level, giving us the highest possible flexibility. Note that, for our prototype, performance was not an issue at this point. When it comes to monitoring busy wireless networks, we probably would have to think of a more efficient solution than the one currently implemented.

Extending Bro's script language. Having defined new built-in events for 802.11, we would now begin writing corresponding event handlers. However, we found that the scripting language used by Bro to define policy scripts is not sufficient when it comes to writing policies for 802.11. In particular, the language does not provide a suitable data type for working with addresses used by the 802.11 protocol. We modified Bro's policy script interpreter and added built-in support for the 802.11 address type. This turned out to be the most time-consuming task during developing our prototype.

Now we are ready to start writing 802.11-specific security policies which is covered in section 4.1.

4 Prototype Validation

For the validation of our prototype we looked at three common threats to wireless networks and developed appropriate detection strategies which we incorporated into a series of policy scripts used by our prototype. The policy scripts were expressed using our extended version of the Bro script language. During an experiment we simulated different attack scenarios and verified the proper functioning of our prototype. In the

following section we will describe the strategies which we used for our prototype and present the results of our lab test.

4.1 Policy Definition

The process of turning a detection strategy into a security policy that can be used by our prototype basically involves specifying the appropriate event handlers defining how certain events should be interpreted. The challenge, however, is to find a suitable detection strategy in the first place. We present some basic strategies for detecting three common wireless threats. They form the foundation of the policy scripts we used during our lab test.

Rogue Access Points. An important step towards detecting wireless attacks is locating so called *rogue access points*. Traditionally, the term *rogue access point* refers to wireless access points that have been attached to a wired network without explicit permission of the administrator. Such access points represent a direct threat to the respective network as they may circumvent existing security measures. However, it is important to note that also access points which are not connected to the wired network can be a security problem when they associate with authorized clients. Imagine an employee's laptop associating with an unknown access point across the street while being connected to the company's wired network. That is why we have to extend the notion of rogue access points to include any unknown access point within range of our sensors–whether or not it is attached to the wired network.

To detect rogue access points, we implement the strategy described in [9] where every discovered access point is matched against a list of trusted access points. In our case, this list includes the MAC address (Basic Service Set Identification, BSSID), the network name (Service Set Identifier, SSID) and the operating channel for each trusted access point. In order to discover new access points, the monitor listens to *Beacon Frames* and *Probe Response Frames*. Note, that an access point must send out at least one of those two frame types to be recognized by a client. When comparing the BSSID, SSID and channel information contained in the received frames with those stored in the list of trusted access points, we distinguish between the following situations:

1. Both BSSID and SSID are completely unknown to the system. In this case, the system would give a notice that an unknown access point has been detected and send an alert as soon as an authorized client associates with the rogue access point. Herefore, one would of course have to address the problem of how to distinguish between foreign and authorized clients.
2. BSSID, SSID and operating channel match an entry in the list of trusted access points. In this case, the system assumes that it has detected a trusted access point. Note however, that it is still possible that an attacker replaced the original access point by his own.
3. BSSID or SSID are known to the system, however, it cannot find a matching entry in the list of trusted access points. In this case, the system would assume that someone attempts to spoof a valid access point and immediately sends an alert.

This strategy may not be suitable for large wireless networks with hundreds of access points. However, in cases where it is possible to manage a list of trusted access points, detection of rogue access points can be implemented in a very straight-forward manner.

Denial-of-Service. There exist different techniques for launching DoS attacks against wireless networks. Most popular attacks use some form of management frame flooding. For example, an attacker could flood the network with spoofed *De-Authenticate Frames* which seem to originate from the legitimate access point. This way, it becomes impossible for any client station to stay connected with this access point. On the other hand, an attacker could flood an access point with spoofed *Authentication Frames*, simulating hundreds of individual client stations attempting to authenticate with the access point. Eventually, the access point will be unable to respond to legitimate client requests [17].

When we look at different forms of management frame flooding, we find that they usually have one or more of the following characteristics:

- exceptional high frequency of certain management frames
- exceptional large number of different source addresses
- destination address set to broadcast address when it should not
- use of invalid source addresses
- unrealistic number of unique network names (SSID) on a single channel

By applying basic sanity-checks on the received management traffic most known flooding attacks should be detected. One could, for example, define threshold values for the number of unique source addresses received during a certain period of time (*Authentication Flooding*), the number of unique network names per channel (*Beacon Flooding*) or just the plain frequency of certain management frame types. Additionally, one could check for invalid source addresses or the destination field set to multicast addresses where it is not appropriate. The actual difficulty is to find suitable threshold values. Setting them too low would cause too many false alarms while setting them too high could mean that we miss less aggressive attacks.

Man-In-The-Middle. Another very popular intrusion technique is the so called man-in-the-middle attack where the attacker positions himself between an authorized client station and an access point. This attack involves setting up a rogue access point, usually combined with spoofing the network name and sometimes even the BSSID of a legitimate access point. Once an authorized client associates with the fake access point, the attacker configures his wireless interface to use the MAC address of the client station and connects to the unsuspecting access point on behalf of the legitimate client. Optionally the attacker could send spoofed *De-Authenticate Frames* prior to the attack in order to force clients to disconnect from the access point.

In order to develop a strategy for detecting man-in-the-middle attacks, we look at them from the perspective of the monitor. From this point of view, a typical attack would look similar to this: When started, the monitor identifies a trusted access point on channel 3. At some point during runtime, it discovers a new access point on channel 11 which it identifies as rogue access point. Depending on the attackers strategy, the monitor could also witness some kind of denial-of-service attack on channel 3. Either

way, the monitor would, at some point, record a successful client association to the rogue access point on channel 11. Shortly after that it would witness the "same" client connecting to the legitimate access point on channel 3.

Based on these observations, a possible strategy could be as follows: first, the monitor has to detect an existing rogue access point. Furthermore, the monitor keeps a record of established connections between access points and clients. It sends an alert as soon as it detects simultaneous connections from one client to different access points, one of those being an identified rogue. Detecting the ongoing MAC spoofing would further improve the monitor's level of confidence. This, however, requires some sophisticated detection techniques as described in [18] and is beyond the scope of this paper.

The strategies described here were incorporated into several policy scripts using Bro's internal scripting language.

4.2 Laboratory Test

The setup used in our lab test is shown in table 1. Our prototype uses off-the-shelf hardware: a notebook with a 1 GHz processor and two wireless PCMCIA cards. Monitor, Attacker and Victim where situated in the same room. The access point was placed in a different room to give the attacking station the necessary gain in signal strength.

During our lab test we conducted the attacks described in section 4.1 using tools available from the Internet. The results of our test are shown in table 2. Except for the Association and Authentication Flooding, all of the attacks were successful. We

Table 1. Test setup

Attacker	Debian GNU/Linux 3.1 (Kernel 2.4.29) Netgear WG311 (Atheros chipset, madwifi driver 15.05.2005) Linksys WPC11 (Prism chipset, hostap driver 0.4.1, airjack driver 0.6.6alpha)
Monitor	Debian GNU/Linux 3.1 (Kernel 2.6.12) Netgear WAG511 (Atheros chipset, madwifi driver 15.05.2005) Netgear MA521 (Realtek chipset, driver by Andrea Merello version 0.21)
Victim	Windows XP Professional (Service Pack 2) Avaya Wireless Silver World Card (Hermes chipset, Windows driver)
Access Point	D-Link DI-624+

Table 2. Test results

Attack	**Tools**	**Successful**	**Detected**
Rogue Access Point	hostapd + airsnarf	Y	Y
Association-Flooding	void11	N	Y
Authentication-Flooding	void11	N	Y
Deauthentication-Flooding	wlan_jack	Y	Y
Beacon-Flooding	fakeAP	Y	Y
Man-In-The-Middle	monkey_jack	Y	Y
Client MAC-Spoofing	no special tool	Y	N

believe that the reason for this is because our access point already had some built-in protection mechanism against this type of attack. More importantly however, except for the client MAC spoofing, all intrusion attempts have been detected by our prototype. As mentioned before, detecting MAC spoofing involves more sophisticated techniques than those described here. However, we believe that it should be possible to incorporate adequate strategies into our prototype.

5 Conclusion and Perspectives

In this paper we presented our prototype of a wireless intrusion detection system. We introduced an existing wired IDS and described our modifications in order to use it for monitoring wireless networks. Also, we analysed common wireless threats and came up with a strategy for detecting them. Finally, we incorporated those strategies into a series of policy scritps which we used to validate our prototype. We demonstrated that our prototype in fact realises basic wireless intrusion detection. Our approach does not require any special infrastructural means or specialised hardware. Thus, it can be easily adopted for any mobile environment. By utilising an existing intrusion detection solution for wired networks our system can easily be integrated into a comprehensive security solution.

However, a few issues remain unsolved. Being merely a proof-of-concept, the detection strategies currently used by our prototype are rather rudimental and have to be improved by further research. The same applies to performance issues. A promising approach for better detection strategies is the aggregation of several IDS into cooperating compounds. Besides that, there also remains a more fundamental problem when implementing wireless intrusion detection systems: in order for them to be effective, they must monitor all channels simultaneously. Since we only want to use standard hardware, this turns out to be a rather difficult task. Possible solutions for this include frequent channel hopping and the implementation of virtual network interfaces. Last but not least, we have concentrated on threat detection and yet neglected the need for (physical) response. We are also aware that our approach is only able to detect previously known attack scenarios. However, the handling of so called unknown attack vectors is beyond the scope of this paper and must be postponed for future reseach work.

In the near future we intend to include our wireless IDS solution as a basic building block in the collaborative architecture of a self-protecting mobile device [19]. Our main concern in doing so is the establishment of trustful communication pathes in the absence of any central instances like trusted third parties.

References

1. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography (SAC 2001). Volume 2259 of LNCS., Springer (2001) 1–24
2. KoreK: The KoreK attack – What FMS conveniently forgot to say. netstumbler.org forum (2004) `http://www.netstumbler.org/showthread.php?t=11869` last visited: February 9, 2006.

3. KoreK: chopchop – Experimental WEP attacks. netstumbler.org forum (2004) http://www.netstumbler.org/showthread.php?t=12489 last visited: February 9, 2006.
4. Anonymous: Wi-Fi Protected Access: Strong, standards-based, interoperable security for today's Wi-Fi networks. Technical report, Wi-Fi Alliance (2003) http://www.wifialliance.com/OpenSection/pdf/Whitepaper_Wi-Fi_Security4-29-03.pdflast visited: June, 28 2005.
5. Anonymous: 802.11i – Amendment 6: Medium Access Control (MAC) Security Enhancements. Technical report, Institute of Electrical and Electronics Engineers, Inc. (2004) http://standards.ieee.org/getieee802/download/802.11i-2004.pdf last visited: February, 24 2006.
6. Yang, H., Luo, H., Ye, F., Lu, S., Zhang, L.: Security in Mobile Ad Hoc Networks: Challenges and Solutions. IEEE Wireless Communications **11** (2004) 38–47
7. Buttyán, L., Hubaux, J.P.: Report on a Working Session on Security in Wireless Ad Hoc Networks. ACM SIGMOBILE Mobile Computing and Communications Review **7**(1) (2003) 74–94
8. Zhang, Y., Lee, W., Huang, Y.A.: Intrusion Detection Techniques for Mobile Wireless Networks. Wireless Networks **9** (2003) 545–556
9. Lim, Y.X., Schmoyer, T., Levine, J., Owen, H.L.: Wireless Intrusion Detection and Response. In: Proceedings of the 2003 IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY, USA (2003)
10. Schmoyer, T.R., Lim, Y.X., Owen, H.L.: Wireless Intrusion Detection and Response. A case study using the classic man-in-the-middle attack. In: Proceedings of the IEEE Wireless Communications and Networks Conference, Atlanta, Georgia, USA (2004)
11. Branch, J.W., Jr., N.L.P., van Doorn, L., Safford, D.: Autonomic 802.11 Wireless LAN Security Auditing. IEEE Security & Privacy (2004) 56–65
12. Welch, D.J., Lathrop, S.D.: A Survey of 802.11a Wireless Security Threats and Security Mechanisms. Technical Report IOTC-TR-2003-101, Information Technology and Operations Center, Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, New York 10996, USA (2003)
13. Schneier, B.: Modeling security threats. Dr. Dobb's Journal (1999)
14. René Neumerkel: Entwicklung eines Angriffssensors für Wireless LAN. Master's thesis, Technische Universität Dresden (2005)
15. Vladimirov, A., Gavrilenko, K.V., Mikhailovsky, A.A.: WI-FOO. The Secrets of Wireless Hacking. Addison-Wesley Professional (2004)
16. Paxson, V.: Bro: A System for Detecting Network Intruders in Real-time. Computer Networks **31**(23–24) (1999) 2435–2463
17. Bellardo, J., Savage, S.: 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In: Proceedings of the 12th USENIX Security Symposium, Washington D.C. (2003) 15–28
18. Wright, J.: Detecting Wireless LAN MAC Address Spoofing. http://home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf (2003) last visited: February, 28 2006.
19. Groß, S.: Selbstschützende mobile Systeme. In: Sicherheit 2006, Beiträge der 3. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI). (2006)