THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 13, Number 2/2012, pp. 167–174

PROTOCOL VULNERABILITIES IN PRACTICE: CAUSES, MODELING AND AUTOMATIC DETECTION*

Bogdan GROZA, Marius MINEA, Marius CRISTEA, Pal-Stefan MURVAY, Mihai IACOB

"Politehnica" University of Timisoara, Faculty of Automatics and Computers
E-mail: bogdan.groza@aut.upt.ro

Starting from practical scenarios we underline that the most relevant security vulnerabilities in practice come from weak protocol design or implementation flaws rather than from weak or flawed cryptography. In particular, we outline security vulnerabilities in several kinds of scenarios starting from well explored fields such as computer networks to less explored ones from the automotive industry and control systems. Some of the security flaws that we discuss are already known while others are new and have been subject of our previous research. Finally, we emphasize that to assure good security, focus should be on assuring correct implementations and proper tools for automatic verification of services.

Key words: protocol, vulnerability, automatic verification.

1. MOTIVATION

Security relies intimately on cryptography. But cryptography alone cannot assure security in practice since cryptography relies on software or hardware for implementation, on networking for data transport, on usability insights (since solutions have to be exploited by humans educated or not), etc. This point of view is not new and is confirmed frequently by practical incidents, some of them even of international level, that show vulnerabilities despite the use of strong cryptography (e.g., in the case of Stuxnet valid digital certificates were used to sign corrupted drivers). Recently, this has been also underlined by top cryptographers Koblitz and Menezes along with strong criticism on some flamboyant cryptographic assumptions from the last decade [12].

Cryptography is built having adversaries in mind, but this does not generally hold for hardware or software and may be one of the reasons why cryptography has reached maturity faster than its software or hardware counterparts. For example, hardware implementations (for now classical cryptosystems) were ready as early as the late 80's but side channel attacks, such as timing attacks or differential power analysis, were considered only in '96 by Kocher et al. [13, 14] showing clearly that nobody devoted attention to the security of the implementation itself for more than a decade. The lack of preoccupation for security is pervasive in many areas. For example, industrial systems security came into attention only in the last decade and there are still many uncovered issues, as Stuxnet showed. The situation is apparently even worse in automotive security where cryptography is almost absent; several spectacular attacks are discussed in [15, 7].

In Fig. 1 we outline a view over the development flow of security solutions from theory to practice. We also point out some moments that we consider relevant for the development of each stage. For modern cryptography we consider the late 70's, although indeed its history can be traced earlier, i.e., the birth of public-key cryptography [8, 19], mainly because this moment marks somehow the move of cryptography from the secret (military) domain to public (academia) domain. Cryptographic algorithms were quickly adopted in protocols, e.g., the famous key distribution protocol by Needham and Schroeder [18], but only

* This paper was presented in part at the 1st Conference *Romanian Cryptology Days* – RCD2011, October 11–12 2011, Bucharest, Romania.

much later did these protocols reach practical implementations and were subject to formal analysis and penetration testing. For practical implementations we point at Netscape's SSL since we consider this to be the first major security suite present in practice. As important points in formal verification of security protocols we highlight Meadows [17] and Lowe [16] for the first use and development of dedicated tools. An even more recent trend shows interest in the automatic generation of proofs for cryptosystems, the CryptoVerif tool developed by Blanchet and Pointcheval [5] is probably the first of this kind. But using formal methods to verify protocols (or even prove cryptosystems) and the existence of such tools does not necessarily make things easier since in order to use formal methods one has to build formal models that can express properties in an abstract way. This is not a trivial task, and sometimes it is hard to define certain notions. Consider for example trust or freshness for which there are several formal definitions that do not necessarily match.

It is easy to see the age difference between cryptographic theory and the actual deployment in practice of these solutions as well as their verification through model checking, penetration testing, etc. Thus if one wants to find a vulnerability in practice, it may be more effective to look at the less mature steps, e.g., implementation, rather than at the cryptographic design. We outline this idea in what follows using a practical case study.

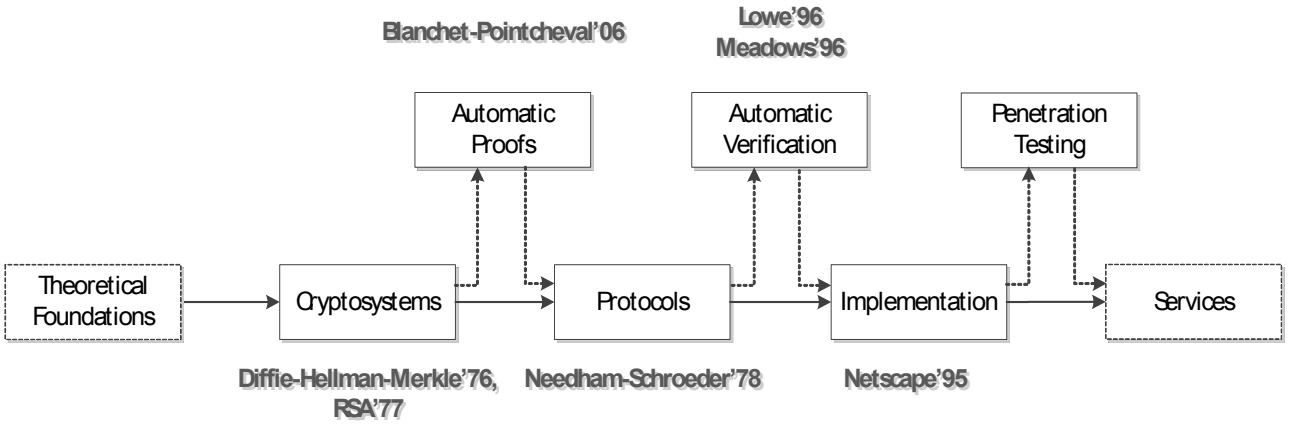

Fig. 1 – Development flow: from theoretical foundations to practice.

We continue with a motivating example. Siemens PLCs (Programmable Logic Controllers) came to widespread attention in the security community after the Stuxnet attack. To communicate over wireless networks, which are present in industry where cables are not an alternative, e.g., moving objects, SCALANCE routers are connected to the PLCs. These routers are the result of careful engineering work. In particular they use state-of-the-art cryptography that comes with SSL/TLS and WPA2. This includes modern algorithms such as: AES, RSA, DSA, ECDH, ECDSA, etc.

Consider a simple scenario in which one wants to cut down the wireless communication. A first impression is that, to achieve this, a wireless signal jammer will be needed. But after careful inspection of the 802.11 standard, it turns out that even if wireless security is assured via WPA2 (the strongest security level of the WiFi suite), de-authentication packets can be sent and they will cause the client to abort the connection. This can be done from any wireless device that allows cloning the MAC of the access point and there are lots of tools that can be used for this purpose.

Things get even worse on careful analysis of the configuration protocol as it turns out that there is a bug in the authentication protocol which allows the reuse of a previous response in the login stage. More, HTTP still works even if HTTPS is disabled after an attack caused by the injection of a wrong SSL/TLS packet (this can actually tempt users to place their password on HTTP which will allow an adversary to capture it). And, less relevant, the authentication protocol is a weak password based protocol while there is no obfuscation in the JavaScript code of the web interface, making it straightforward to determine how authentication is done. Relevant to note, password based protocols resilient to guessing attacks are known for almost two decades but they are still somewhat absent from practical implementations.

Table 1

Security features and vulnerabilities in an industrial WiFi AP

Security Features	Security Vulnerabilities
Embeds state-of-the-art crypto (RSA, ECC, ECDH, AES, etc.)	Accepts (forged) de-authentication packets
WPA2 WiFi Security	Bug in the authentication protocol
HTTPS Web Security (SSL/TLS)	HTTP still active after HTTPS locks
	No obfuscation in the Javascript code

2. PROTOCOL VULNERARBILITIES

To support the thesis, we outline several flaws in scenarios that cover a larger area from practice. We start by outlining some flaws that still persist in computer networking, despite the fact that most of them are commonly known. The we proceed to automotive and control systems, two areas which are somewhat newer for security incidents but for which practical incidents in the last five years showed that they can not be neglected anymore.

2.1. Computer Networks

Computer networks depend on protocols that specify the messages that are exchanged at runtime, their format and structure. Protocols are linked with different protocol stacks, e.g., TCP/IP, or different models, e.g., OSI, and many protocols with underspecified security are still present in practice. We enumerate several such issues below.

The Point-to-Point protocol (PPP) (RFC 1661) is a data link layer protocol used for link establishment between two network nodes. This protocol is used by some ISPs when providing Internet services to clients. PPP can assure node authentication, encryption and data compression through the PAP and CHAP (RFC 1994) protocols.

PAP Authentication requires a simple handshake between two principals denoted in what follows as *A* and *B*: $A \rightarrow B$: $username_A$, $password_A$; $B \rightarrow A$: $username_B$, $password_B$. It is easy to see that attacks against PPP with PAP can be mounted in a straightforward manner because both the *username* and the *password* are sent in clear text.

CHAP Authentication is stronger and follows a challenge-response structure in three steps as shown below. We use standard notations: *A, B* are participant's names, *SN* is a sequence number and *Pwd* is the password. The *nonce* is used by CHAP to provide replay protection. To check for correctness, principal *B* will compute an MD5 on his side and if his result matches the one received from principal *A* then the authentication is correct. From time to time one of the nodes can request this process to be repeated. Even if CHAP provides a better authentication mechanism that PAP an attack can be also launched against CHAP as shown below. At the end of this attack *A* thinks that it has completed a connection with *B*, while *C* thinks that it is connected with *A*. To mount this attack, the same password must be present on all the routers from the routing domain - an occurrence that we emphasize is quite frequent in practice.

A→B: request
B→A: B,SN,Nonce
A→B: A,SN,MD5(SN,Pwd,Nonce)

A→B: request
Adv→C: request
B→Adv: B,SN,Nonce
C→Adv: C,SN',Nonce'
Adv→A: B,SN',Nonce'
A→Adv: A,SN',MD5(SN',Pwd,Nonce')

Fig. 2 – CHAP protocol (left) and an impersonation attack (right).

The Routing Information Protocol (RIP) (RFC 1058) is a network layer protocol used for path establishment by routers on the network. RIP gained popularity because it was deployed on BSD as the routed daemon. Since version 2, RIP offers authentication support (RFC 2453). The first authentication method, a plaintext authentication, requires again an insecure step: A→B: RP, Pwd. Here RP is the RIP packet. If the password that comes together with the routing update message is the same as the one configured on router B then the router will accept the routing update, otherwise the routing update is discarded. In case of plaintext authentication the password can be easily captured by the attacker using a sniffer. Once the attacker obtains the password he can construct false routing updates, by using these false routing updates, the routing tables of the whole routing domain can be corrupted, leading to a DoS attack. The second method is called cryptographic authentication (RFC 4822) and uses a keyed hash over the update packet: A→B: KID, SN, MD5(RP|Pwd). Here: KID is the id of the key that is used. If SN is bigger or equal to the last one that was received and if the MAC is correct then B accepts the routing update packet, otherwise it will be discarded. When using RIP, with MD5 authentication, the following attack can occur (presented in Fig. 2): the attacker captures a packet that was sent to Network 4, with sequence number SN4. It monitors the link between A and B to observe when the sequence number of the link becomes smaller than SN4. The sequence number is on 32 bits and it eventually cycles since one can force the router to increase the counter even by sending packets that will fail the authentication. Now, the attacker sends the captured packet to B. In this way B will think that Network 4 is directly connected to A and it will update its routing table to send all the packets destined to Network 4 through A, so data will no longer be able to reach Network 4. Another attack can be mounted against RIP with MD5 authentication when the split horizon with poison reverse rule is used. For example router C learns about Network 4 through router D. Because of the split horizon with poison reverse rule router C will send immediately a routing update to router D for Network 4 having the metric value 16 (infinite metric). The attacker can capture this packet, alter the source as coming from router D and send this packet back to C. Router C will think that Network 4 is no longer available and will no longer forward data to this network. This is also a form of DoS attack since router C can no longer reach Network 4.

MS-CHAP and NTLM are application layer protocols developed by Microsoft that are used by Windows applications to access remote resources. The security of these protocols is based on the strength of the password which is chosen by the user. If the password has low entropy then the protocols can be attacked using dictionary attacks. Although password based protocols that are resilient to guessing are known, these weak protocols are still in use in Microsoft-based networks.

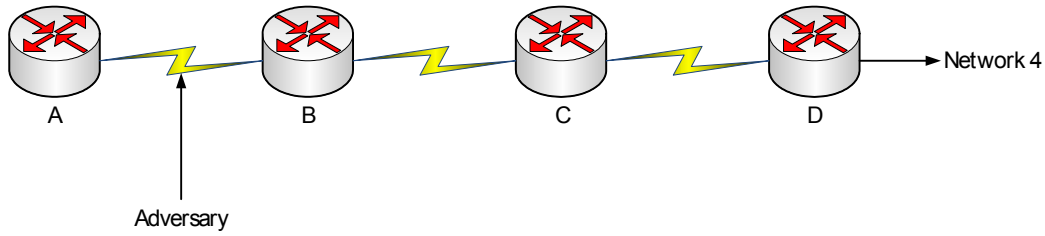

Fig. 3 – Network topology.

2.2. Automotive Industry

Embedded software employed in the automotive industry has evolved into complex applications due to the great variety of services offered by a modern car. Until recently, the security of such systems has been somewhat neglected and, as a result, many vulnerabilities are present in today's automobiles. Some standards that impose the usage of security mechanisms exist and are used by automotive software manufacturers. However, these standards do not cover the whole spectrum of applications and often leave some important aspects, e.g., the cryptographic functions that should be used in the authentication protocol, up to the manufacturers. This is an important issue since it is not clear whether there is sufficient expertise to develop security on the manufacturer side. Moreover, almost always security details are hidden. Note that in contrast most of the secure solutions from practice, e.g., SSL/TLS, IPSec, are the result of public scrutiny and many subsequent versions of the protocol.

A continuously increasing interest is found in breaking protocols used for software reprogramming to allow tuning or a cheaper way of upgrading different vehicle modules. Software reprogramming is usually done using the diagnostics interface and an Internet search will show that the online community provides cracked diagnostic software and counterfeited diagnostic connectors for many car makes and models. A commonly exploited vulnerability in the case of diagnostics done over controller area networks (CAN) stems from the standards [24, 25] which specify that an authentication mechanism called SecurityAccess should be implemented for restricted diagnostic services (such as software update). However, no rule is imposed for the actual algorithm that should be used for the computation of the security keys. The standard also states that the key can be either stored or computed upon each request. Koscher et al. [15] showed that the 16 bit stored key could be found in about seven and a half days in their experiments.

While some security mechanisms are provided for connecting to the interfaces available in vehicles, the protocols used for in-vehicle communication do not employ security. Koscher et al. [15] used a packet sniffing tool and reverse engineering to find a number of attacks on different vehicle modules. They also revealed deviations of the software implementation from the protocol standards, which provided new vulnerabilities. Sniffing together with replay attacks were also used by Hoppe et al. [11] on a CAN bus to build attacks on an electric window lift system.

Wireless communication is being used more and more by modern automobiles. Passive keyless entry, the immobilizer, the multimedia system and even the tire pressure monitoring system are using wireless communication to perform specific tasks. Like in the case of wired buses, vulnerabilities exist especially when wireless technology is employed. Rouf et al. [20] investigated the case of a tire pressure monitoring system and proved that eavesdropping is possible at a distance of roughly 40 meters. The lack of authentication and basic input validation mechanisms enabled them to reverse engineer the communication protocol and to trigger the sensors remotely resulting in a false warning being issued to the driver. In some cases a secure wireless communication protocol proves to be insufficient for providing security. As an example, the strong authentication and encryption used by passive keyless entry and start systems can be bypassed using simple relay attacks as proved by Francillon et al. in [10]. They built an inexpensive construction that allowed them to relay messages between the car and the smart key. The relaying system enabled them to open the car and even start its engine from distances up to 50 meters (maximum tested distance) without the need of getting close to the key (the key could be excited from up to 8 meters).

One way of counteracting attackers in search of protocol vulnerabilities is to use Intrusion Detection Systems (IDS). Such systems would monitor the in-vehicle communication in search for specific patterns that would indicate the presence of an attacker. Hoppe et al. [11] propose three patterns that may be used by an IDS operating on a CAN bus: abnormal message frequency, obvious misuse of CAN IDs and electrical signal characteristics. For building an efficient IDS, a comprehensive spectrum of attack patterns has to be established as a result of the careful modelling of attacker capabilities and behaviour along with a corresponding threat analysis [23].

2.3. Control Systems

In today's automation world the requirements are high and complex. Each new system has to be better, more reliable, more flexible and more user friendly than previous ones. New software technologies have emerged due to the need of integrating machines, control and monitoring instruments, field equipments in easy-to-use visualization environments and web-based applications. These technologies include: PCS7, Step7, WinCC SCADA/Flexible (Siemens), CX-Supervisor/CX-Programmer (Omron), Genesis 32 (ICONIX), Labview (National Instruments). By merging these concepts and modules, a successful and fully functional SCADA, DCS or PLC based system can be obtained. But there is still little attention in assuring security with respect to cyber attacks as shown by practical incidents, e.g., Stuxnet.

To draw a succint image of a modern SCADA/DCS system we outline one brief design in Fig.4. The schematics illustrates an actual SCADA topology, with Siemens equipments and software for monitoring and control of thermal power plant processes. Functional requirements include a minimum key set of features such as: uninterrupted operator control and monitoring, remote and safe operation and monitoring from anywhere in the world, suggestive synoptic images, evolution charts with database value storage, archives and easy to interpret alarm system, report generation on key parameters of the plant, flexibility, scalability

and powerful modular structure. Some of these features respond to clear security objectives such as: authorization, availability, traceability, etc. But there are still many security objectives that are not covered and this leaves room for attacks.

From the hardware point of view the system presented in Fig. 4 is built on Siemens modules and contains: PLCs, distributed data acquisition slaves (wired or wireless), operator servers, a web-server, human machine interfaces (HMI), an engineering system (ES), switches and wireless routers. While the exact behavior of this system is not relevant to the subject of our paper, this topology inherits several security vulnerabilities which concern us. Flaws in Scalance wireless routers were already outlined in Table 1 and in previous research we showed that these can be exploited to place the controlled process in undesired states. The topology also includes standard PCs for clients and servers that all run under the Windows OS and can be target to common viruses, trojans and any kind of malware. This was exactly the entry point of Stuxnet. Finally, PLCs communicate on ProfiBus which has no cryptographic security whatsoever. Indeed, embeding security in ProfiBus was not an objective decades ago when these devices operated in secure perimeters. But this is certainly not the case today, so more efforts should be focused on assuring security in such environments.

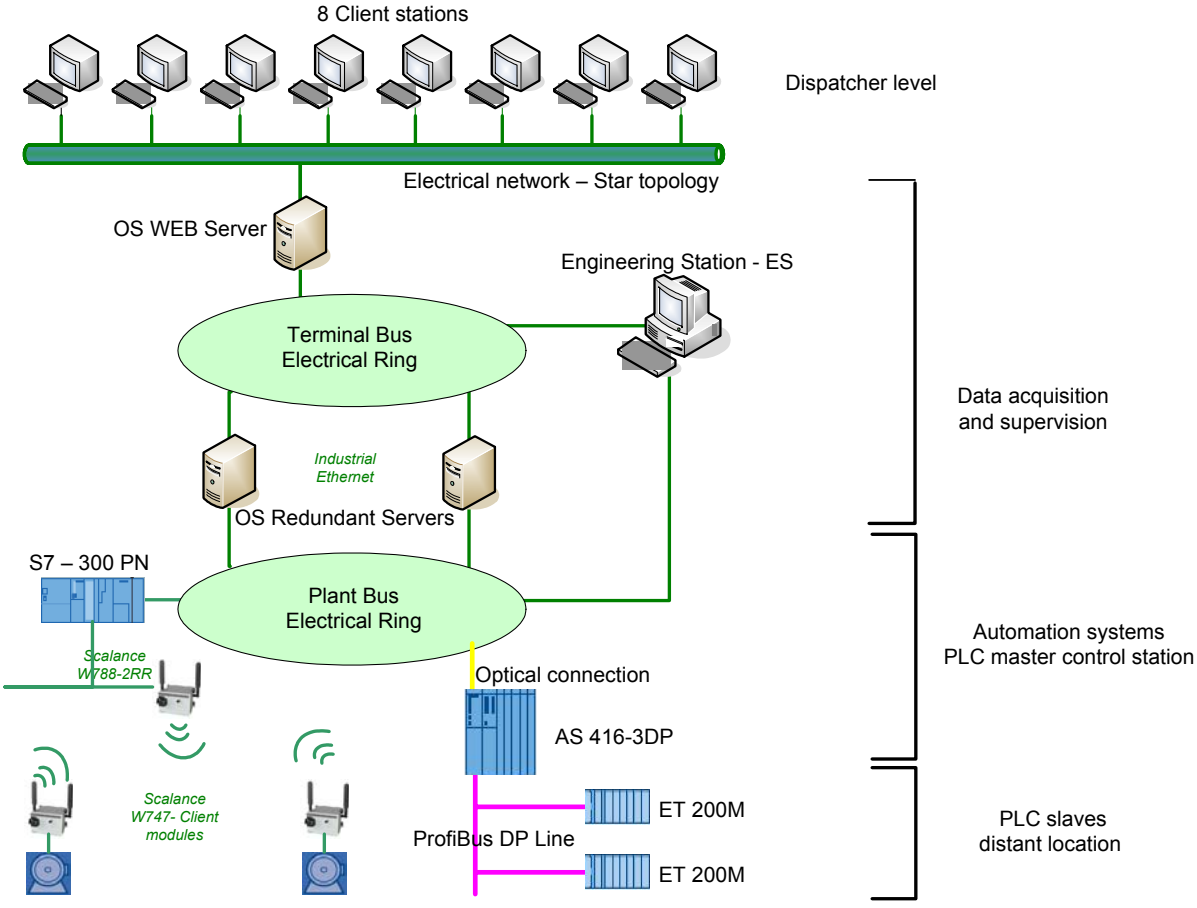

Fig. 4 – A possible SCADA schematics for process control and monitoring.

3. MODELLING AND AUTOMATIC DETECTION

Cryptography offers secure blocks for most operations needed in practice: hashing, encryption, signature, etc. There are still some areas in which very efficient solutions are not yet known, for example deniable encryption or fully homomorphic encryption, but a large majority of security objectives are efficiently covered by cryptographic primitives and protocols. Notable, cryptographic theory is well ahead of practice and solutions exist even for the time when quantum computers will appear.

Significant progress has been also made on the formal verification of security protocols. There are many tools capable of verifying protocols: Meadows' NRL [17], Lowe's FDR [9], Blanchet's ProVerif [4], the AVANTSSAR tool-set which includes three such tools CL-Atse [22], OFMC [3] and SATMC [2]. However it is notable that these model checkers cannot handle all types of attacks. Examples include resource exhaustion and guessing attacks which we have addressed in previous work for the AVANTSSAR tool-set.

An uncovered issue is the gap between the formal and computational views over cryptographic primitives. The formal view treats cryptographic operations in a symbolic manner while the computational one uses mathematical instruments such as complexity theory or probability theory. The first approach is more versatile and can be handled even by automatic tools but it can miss subtle aspects of certain cryptographic primitives. Bridging this gap has evolved into a very active area since the seminal paper of Abadi and Rogaway [1].

But there is an even bigger gap between formal methods and tools for penetration testing. Verifying implementations is not easy, and in fact can be impossible, since solving such an issue can be equivalent to solving the Turing completeness problem. But there is hope. In order to achieve the desired level of abstraction, one approach is to model the API of an implementation rather than the code. An interesting result is presented in [6] where attacks generated by a model checker are used to test security tokens, succeeding in breaking all of them. Two complementary approaches can be used to verify actual protocol implementations. Automatic extraction of models that can subsequently be verified formally has been done for C, F# and Java. The opposite approach is to construct and verify models, and provide a correct-by-construction code generation technique, which has the advantage that more discipline can be enforced for both model and code.

For implementations that are not limited to protocols, a different direction concerns static analysis techniques to detect or disprove certain classes of attacks. Type-based techniques and taint analysis are used for analysis of buffer overflow, code injection and format string and attacks, and other general information flow properties are checked using static analysis.

Formal methods have also been used in lightweight and mixed approaches to verify implementations. Security-specific mutation operators can be applied to formal models, with the resulting attack traces used for mutation testing. White box fuzz testing uses advances in symbolic execution to direct exploration and achieve systematic coverage. Efficient constraint solvers for strings can be used to detect vulnerabilities such as buffer overflows and generate actual concrete inputs that exhibit the attacks.

Many of these techniques have been combined into state-of-the-art implementations, such at the BitBlaze binary analysis platform [21] or the commercial Coverity suite that can analyze realistic applications, making automatic detection of several classes of security vulnerabilities a practical reality.

4. CONCLUSION

Nowadays, if one wants to mount an attack against some principal then implementation flaws should offer a bigger success rate than shortcomings in the cryptographic design. It is hard to state a more concise conclusion on the security of a system than Koblitz and Menezes did in [12]: "such an evaluation needs to incorporate many other disciplines and involve people with hands-on experience and not just theoretical knowledge". Thus security is a mixture between efficient cryptosystems with tight security proofs, well investigated mathematical backgrounds, provable (or formally verified) protocols, correct (possibly verified or provable) implementations and of course educated users. To this one can add plenty of less understood black magic which is the reason why many ground-breaking papers will always appear on this subject.

ACKNOWLEDGEMENTS

This work was partially supported by CNCSIS-UEFISCDI project number PNII-IDEI 940/2008 and by the strategic grant POSDRU/21/1.5/G/13798, POSDRU 107/1.5/S/77265 and POSDRU/88/1.5/S/50783, Project ID50783 (2009), inside POSDRU Romania 2007-2013, co-financed by the European Social Fund Investing in People.

REFERENCES

1. M. ABADI and P. ROGAWAY., *Reconciling two views of cryptography*, Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, pp. 3–22, 2000.
2. A. ARMANDO and L. COMPAGNA, *SAT-based model-checking for security protocols analysis*, International Journal of Information Security, **7**, *1*, pp. 3–32, 2008.
3. D. A. BASIN, S. MODERSHEIM, and L. VIGANO, OFMC: *A symbolic model checker for security protocols*, International Journal of Information Security, **4**, *3*, pp. 181–208, 2005.
4. B.BLANCHET, *An eficient cryptographic protocol verifier based on prolog rules*, Computer Security Foundations Workshop. IEEE Computer Society, 2001.
5. B.BLANCHET and D. POINTCHEVAL, *Automated security proofs with sequences of games*, Advances in Cryptology CRYPTO pp. 537–554, 2006.
6. M. BORTOLOZZO, M. CENTENARO, R. FOCARDI, and G. STEEL, *Attacking and fixing PKCS# 11 security tokens*, Proceedings of the 17th ACM conference on *Computer and Communications Security*, pp. 260–269; ACM, 2010.
7. S. CHECKOWAY, D. MCCOY, B. KANTOR, D. ANDERSON, H. SHACHAM, S. SAVAGE, K. KOSCHER, A. CZESKIS, F. ROESNER, and T. KOHNO, *Comprehensive experimental analyses of automotive attack surfaces*, USENIX Security 2011.
8. W. DIFFIE and M. HELLMAN, *New directions in cryptography*, IEEE Transactions on, Information Theory, **22**, *6*, pp. 644–654, 1976.
9. B. DONOVAN, P. NORRIS, and G. LOWE, *Analyzing a library of security protocols using Casper and FDR*, Proceedings of the Workshop on Formal Methods and Security Protocols, Citeseer, 1999.
10. A. FRANCILLON, B. DANEV, and S. CAPKUN, *Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars*, Proceedings of *Network and Distributed System Security* Symposium, 2010.
11. T. HOPPE, S. KILTZ, and J. DITTMANN, *Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures*, Proceedings of the 27th International *Conference on Computer Safety, Reliability, and Security* (SAFECOMP'08), Springer-Verlag, 2008, pp. 235–248.
12. N. KOBLITZ and A. MENEZES, *The brave new world of bodacious assumptions in cryptography*, AMS Notices, **57**, pp. 357–365, 2010.
13. P. KOCHER, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology CRYPTO96, Springer, 1996, pp. 104–113.
14. P. KOCHER, J. JAFFE, and B. JUN, *Differential power analysis*, Advances in Cryptology (CRYPTO99), Springer, 1999, pp. 789–789.
15. K. KOSCHER, A. CZESKIS, F. ROESNER, S. PATEL, T. KOHNO, S. CHECKOWAY, D. MCCOY, B. KANTOR, D. ANDERSON, H. Shacham, and S. SAVAGE, *Experimental security analysis of a modern automobile*, Security and Privacy (SP), 2010 IEEE Symposium on, May 2010, pp. 447–462.
16. G. LOWE, *Breaking and fixing the Needham-Schroeder public-key protocol using FDR*, Tools and Algorithms for the Construction and Analysis of Systems, 1996, pp. 147-166.
17. C. MEADOWS, *The NRL protocol analyzer: An overview*, Journal of Logic Programming, **26**, *2*, pp. 113–131, 1996.
18. R. NEEDHAM and M. SCHROEDER, *Using encryption for authentication in large networks of computers*, Communications of the ACM, **21**, *12*, pp. 993–999, 1978.
19. R. RIVEST, A. SHAMIR, and L. ADLEMAN, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, **21**, *2*, pp. 120–126, 1978.
20. I. ROUF, R. MILLER, H. MUSTAFA, T. TAYLOR, S. OH, W. XU, M. GRUTESER, W. TRAPPE, and I. SESKAR, *Security and Privacy Vulnerabilities of I-Car Wireless Networks: A Tire Pressure Monitoring system Case Study*, Proceedings of USENIX Security Symposium, 2010.
21. D. SONG, D. BRUMLEY, H. YIN, J. CABALLERO, I. JAGER, M. KANG, Z. LIANG, J. NEWSOME, P. POOSANKAM, P. SAXENA, *BitBlaze: A new approach to computer security via binary analysis*, Information Systems Security, Springer, 2008, pp. 1–25.
22. M. TURUANI, *The CL-Atse protocol analyser*, The 17th International Conference on Term Rewriting and Applications, LNCS **4098**, pp. 277–286, Springer, 2006.
23. C. VESTLUND, *Threat Analysis on Vehicle Computer Systems*, Master Thesis, Linköping University, 2010.
24. *** *Road vehicles – Diagnostics on Controller Area Networks (CAN) – Implementation of unified diagnostic services (UDS on CAN)*, ISO 15765-3, 2004.
25. *** *Road vehicles – Unified diagnostic services (UDS) – Specification and requirements*, ISO 14229-1, 2006.

Received March 13, 2012