OVERVIEW OF SIDE CHANNEL AND TIMING ATTACKS

MARTIN RUBLIK

Attacking a system is a tradeoff between the attacker's possibilities and gains and be-tween the difficulty of an attack. The article will focus on special techniques that attacker can use to gain valuable information only by observing the system. Though these types of attack are not easy to execute, the attacker can gain a lot. These attacks can be used to subvert a system that is secure under a commonly accepted threat model. A system can be secure in theory, but the at-tacker can subvert the system's defense by exploiting the vulnerabilities that were either not included in the threat model or emerged during the implementation phase.

Introduction

Based on what the attacker knows about the target we can divide the attacks into several categories:

- attacks where the attacker does not know anything about the target,
- attacks where the design of the target system is known,
- attacks where implementation of the target system is known or
- attacks where details about the implementation and environment specific details are known (such as software and hardware equipment, services topology, etc.).

Some systems rely on obscurity as a defense mechanism and this is where the first class of attacks belongs. In general this is not a good tactic, especially in cryptography. Modern cryptographic systems are designed with a premise that the internals of the system are known to the attacker. The reason is that it is really hard to keep secret about the design of a system, as attacker can usually reverse engineer the system easily. Several examples support this design principle. Among the others the most known failure was design of DVD CSS algorithm used for DVD protection. This algorithm was broken by Jon Lech Johansen who was only 16 year old at the time.

Attacking the system design is mostly a theoretic task, but breaking it, has severe consequences to the system and its practical use. In cryptography these types of attack are mostly algorithmic attacks and are of course implementation independent. Therefore when a *practical* algorithmic attack on cryptographic system is found the system needs to be replaced where applicable. An example of such an attack would be design flaws in WEP [1] that lead to WPA/WPA2 rollout or flaws found in MD5 hash algorithm [2] that lead to global hash algorithm change in X.509 certificates.

The third class of attacks is implementation specific and in general can be "easily" patched. However the patching possibilities may vary depending on the system's characteristics. For example it is easier to patch a network connected server operating system vulnerability, but it is much harder to patch an offline embedded system.

An example of implementation specific flaw would be the infamous Debian cryptographic random number generator flaw that allowed the attacker to brute-force a private cryptographic key generated on a Debian server [3] easily. The issue was that the key-space used for generating the private keys was reduced less than 300K keys due the bug in the random number generator. PRNG was incorrectly seeded by process IDs ranging from 0 to 32767 providing not enough seed entropy (only 15 bit) for generating the private keys.

Finally attacks, where special hardware or software equipment is known to the attacker are applicable only in special situations. On the other hand attacker has most knowledge about the targeted system, thus he can exploit system in several ways. An example of implementation and environment dependent attack on a system is a side channel attack.

Side channel attacks

Side channel attack is an attack where attacker infers valuable information by observing the system's behavior. In general side channel attack analyses leaked information with regards to hardware, software, implementation and design specifics of the targeted system.

Side channel attacks can be either passive or active. In a passive side channel attack the targeted system leaks information within the ordinary communication and the attacker deduces the results only by observing the system. In an active side channel attack the attacker can actively manipulate the

system and its hardware in either an ordinary or invasive / malicious way so it leaks useful information.

Active side channel attacks can exploit faults of a system (fault attacks). In these cases the attacker brings system to erroneous states in order to observe errors. An example of fault attack is presented in [4] and involves stressing a smart-card with high voltage or changing the system clock signal. These actions can lead to revealing information about stored secret keys protected by the smart-card in some cases.

A system can leak information by various means. Most known side channels include:

power analysis electromagnetic emissions (EMM) acoustic analysis	Attack requires close access to the target in order to gather information.
traffic analysis timing attacks	System observation *can* be performed remotely. It is possible to mount attacks against remote targets.

The power analysis is in general a non-invasive attack. Attacker measures the power consumption during the computations done by a system which requires physical access to the target. This type of attack is therefore executed mostly against tamper evident / resistant hardware such as smart-cards and hardware cryptographic modules in order to gather information about the protected cryptographic keys. For systems running on common hardware there are easier ways to subvert the system using physical access, such as installing malicious software or utilizing hardware key-loggers. The idea behind the power analysis is that the targeted system consumes different amounts of power depending on the bit structure of a key. For example it is possible to reveal the private key of *unprotected* RSA implementation that uses square-and-multiply algorithm for modular arithmetic. In a square-and-multiply algorithm the power consumption varies significantly [5] when the private exponent's current bit is 0 or 1. Therefore by analyzing the consumption during the computation with a private key, the attacker can deduce the private exponent of RSA key pair. Practical example of power analysis was recently presented in research done by Sergei Skorobogatov (*University of Cambridge*) and Christopher Woods (*Quo Vadis Labs*). Using power analysis these researchers found a backdoor in a commercial field programmable gate array used also in a military sector [6].

Similar to power analysis is an attack technique based on measuring the electromagnetic emissions. In order to measure the electromagnetic emissions it is necessary to have close access to the system, but the attacker does not need the physical control over the system. EMM attacks can be executed against cryptographic tamper evident / resistant systems in manner similar to power analysis attacks, but can be useful for compromising a general system as well, because the physical control over the system is not strictly required. Van Eck (a Dutch computer researcher) described a process that used the EMM analysis in order to eavesdrop on the contents of CRT monitor [7]. This type of attack was later extended to eavesdropping on LCD panels as well. Another famous EMM analysis was performed by Martin Vuagnoux and Sylvain Pasini (*Security and Cryptography Laboratory/ EPFL*). They took advantage of electromagnetic emissions in order to eavesdrop on wired and wireless keyboards [8].

Finally the third class of side channel, that requires close access to the targeted system in order to observe the leaked information, is acoustic analysis. During acoustic analysis the attacker monitors sounds emitted by components of the system. Intuitive examples of such components would be keyboards [9] or printers [10]. Less intuitive example is exploiting sounds emitted by CPU. Different operations performed by a processor have different sound signatures. Based on this information it is possible to identify what type of operation is performed by CPU. Attacker can misuse this information in a combination with different type of side channel attack, for example in a timing attack [11].

Side channel attacks where system can be observed remotely are more practical for breaking into common systems for an obvious reason: the attacker does not need physical access to the targeted system (which is in general hard to gain). One of the side channel attacks that could be mounted remotely is traffic analysis. Traffic analysis is not a new concept and was used in military circles long before computer security. In computer security traffic analysis can be used to gain information based on size, *frequency* and *timing* of network communications. Traffic analysis is therefore in a close relationship with timing attacks. A practical example of traffic analysis was recently provided by *IOActive Research Labs*. In their proof-of-concept [12] they showed that it is possible to gather information about the places visited through *Google Maps™* over SSL encrypted connection by analyzing the encrypted network communications.

Last class of side channel attacks analyses system's response time in order to gather valuable information. This type of attack is called timing attack and we will describe it further in the article.

Timing attacks

Side channel timing attacks are commonly used in order to gather information that is not very large, does not change during the attack (at least not in a significant way) and is used as computation parameter during the attack. An example of such information would be cryptographic keys or passwords.

Timing attack was first introduced in scientific literature by Paul C. Kocher [13] in 1996. This paper is cited in almost every scientific article concerning timing attacks and in the paper Paul Kocher describes an attack that can recover RSA private cryptographic keys. This attack is not computationally demanding and requires only a few thousand cipher-text samples, which made it implementable in practice at the time of publishing the results.

Since then the timing attacks got even more practical. First timing attack that was applicable against a widely implemented solution was timing analysis of keystrokes in SSH protocol [14]. This attack combined traffic analysis of SSH encrypted communications and timing analysis of inter-keystroke information of user's typing. The researchers performed a statistical study on users typing patterns that improved a brute-force password attack by factor of 50. Using a special algorithm for guessing the passwords the researchers reduced the password space to 1/50 in common case.

Later on a *remote* timing attack was presented by David Brumley and Dan Boneh (both from *Stanford University*) [15]. They attacked *Apache mod_SSL/OpenSSL* and were able to extract private keys over a local area network that included several network segments divided by routers and switches. This was a major breakthrough as until then, many people believed, that remote timing attacks were not practical. Until then most people also thought that timing attacks were applicable only to special hardware such as security tokens or smart-cards because decryption times would be masked by many concurrent processes running on a common operating system (such as web server). The attack took advantage of key negotiation in SSL protocol. In the SSL key negotiation phase the server decrypts data chosen randomly by the client in order to prepare a shared secret that is used for protecting the communication later. The attacking client sent malformed data, that made the server break communication with

an alert response. The time elapsed between sending the data and receiving the alert was used as observation sample by the attacker. The attack took about two hours and required around a million queries in order to extract 1024 bit RSA key.

Finally timing attacks were mounted in practice against commonly used consumer products. In 2007 a timing attack was used to crack *Xbox 360™* console [16]. The goal was to replace the Xbox operating system with an older one. The flaws in older Xbox kernel versions lead to possibility of running pirated software. The core of the attack resided in *memcmp* function that is used for comparing two memory blocks. This function was also used for comparing the HMAC values used to authenticate the boot-loader/kernel. The *memcmp* flaw resides in fact that the comparison is done byte-by-byte and will end immediately after the first difference is found. This flaw was exploited as follows. The attacker started the HMAC comparison with a random value and varied the first byte. He measured the response time for all comparisons (bytes 0, …, 255) and chose the value with the longest measured time. This way the attacker can guess the first byte of the HMAC based on the fact that the comparison with correctly guessed byte will take a longer time (as the other comparisons will terminate immediately). The time difference between correct and incorrect guess was about 2200 microseconds. By iterating the guess cycle the attacker finally he got all the necessary bytes of the correct HMAC.

Timing attacks can also be used in situations that do not strictly belong to side channel attacks. In these situations the attacker uses the timing information for creating a covert channel that transfers the information from a compromised system. An example of such covert channel could be illustrated by blind SQL injection.

In blind SQL injection the system is prone to SQL injection but the attacker is unable to download results of the exploit through the web page in a traditional way. However he can guess the results based on the response time of a server. Following listings illustrate the possible SQL statements that can be used in order to infer this type of information (Listing 1).

The first query makes use of *MySQL* BENCHMARK command. This command is used for multiple executions of an expression. By running high number of operations (such as generating a MD5 hash) we can create a time delay. If the user *root* is present in the table *users* then the query will run for a longer time, otherwise it will finish almost immediately. This way the attacker can determine whether *root* access is enabled on a *MySQL* server (Listing 2).

A similar technique is illustrated in listing 2, though the SQL code is targeted on MS SQL database engine. If the user that is accessing the database is a domain administrator the query execution lasts more than 10 seconds.

Finally the third listing illustrates an attack that makes use of a time demanding query. The command selects and counts the result over a large table created by a series of cross joins. A clean SQL server installation has about sixteen users in *sysusers* table. The number that should be the result of the query is therefore quite big ($16^8=2^{32}$) and the join / count takes a while. The query will run for a longer time only if the first (highlighted) part of the condition is true, otherwise the SQL optimizer will stop the query and return an empty set. This type of optimization is also known as early exit and represents the core idea of all timing attacks. The attacker starts with guessing the first character of the currently connected user and ends when he guesses the entire logon name. The attacker code could be optimized by performing a binary search on the current guessed character in the user name, instead of using simple substring comparison (Listing 3).

Though blind SQL injection attack is time consuming, it works every time the system is prone to SQL injection and it is easy to automate and execute. Attacker can even take advantage of several tools for automating SQL injection, including famous *sqlmap*.

The results of blind SQL injection attack or attacker introduced time related covert channel can be accurately distinguished, because the attacker controls the execution time. This is not true for classic side channel timing attacks. The classic timing attacks leak information with much smaller variance. In most cases it is necessary to collect many measurements in order to mount successful timing attacks.

Nate Lawson and Taylor Nelson gave a talk at Blackhat 2010 [17] that described methods and procedures for mounting a timing attacks in practice. In general the steps of executing a timing attack can be summarized as follows:

- Set up the client to query the target and measure typical response time
- Vary the parameters, take many samples per guess and prepare a statistical base

Listing 1. *MySQL query for checking if* ***root*** *user exists in user database*

```
      IF EXISTS (SELECT * FROM users WHERE username = 'root') BENCHMARK(100000000,
MD5('expression'))
```

Listing 2. *MSSQL query for checking whether current connected user is a domain administrator*

```
      IF (SELECT SYSTEM_USER)='DOMAIN\Administrator' WAITFOR DELAY '0:0:10'
```

Listing 3. *MSSQL (time demanding) query for checking whether current connected user name starts with DO*

```
      SELECT 'FOO'

      WHERE

      SUBSTRING(SYSTEM_USER,1,2)='DO' AND

      (SELECT count(*) FROM sysusers AS sys1, sysusers as sys2, sysusers as sys3,
sysusers as sys4, sysusers as sys5, sysusers as sys6, sysusers as sys7, sysusers
as sys8)>1
```

- Perform a statistical analysis, filter the base
- Guess the partial secret
- Repeat the steps until the entire secret is revealed

The most important steps are the statistical base preparation and the analysis. It is necessary to collect and filter the results so that the positive and negative guess can be distinguished. An example of such basis is illustrated by following figure.

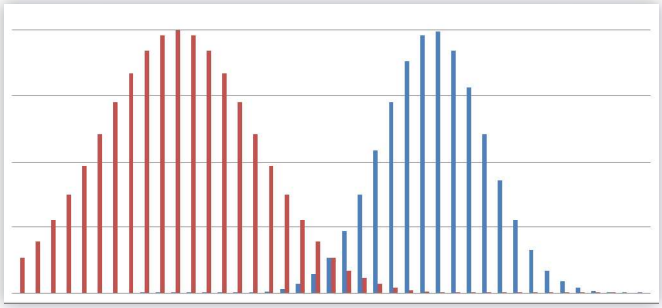

Figure 1. *Sample histogram of measured events*

The sample histogram illustrates two easily distinguishable sets of events (time measured when the test was run with first set of parameters is illustrated by red color the other one is blue). This way the attacker can decide with a reasonable amount of certainty which guess was correct. If the histogram is not that perfect the attacker could try to enlarge the base, use statistical filtering methods, vary the parameters or try to change the environmental circumstances so that he can take more accurate measurements. If the attacker is not able to distinguish the sets the system is not vulnerable to timing attacks.

Defense techniques against side channel timing attacks

The obvious defense against timing attack is to eliminate all the leaks, so that system finishes computation in constant time regardless of the input. This is especially important for security sensitive operations such as operations with cryptographic or security characteristics, but can be time intensive and could degrade the performance for other non-sensitive operations.

Another possibility is to introduce random delays / noise to the affected processes. This could look effective at first glance, but the attacker can bypass this protection by increasing the number of measurements [13]. In general the number of samples increases quadratically with the timing noise. This means that if the standard deviation between the samples was *10ms* and the added noise was normally distributed with 1 second standard deviation, then if the original attack required 1000 operations for successful outcome, the attacker will need to perform $(1000ms/10ms)^2*1000$ operations.

Another possible defense relies on detecting an attack. This may be possible depending on circumstances. For example it is suspicious if the same client sends invalid HMACs or malformed SSL HELLO requests to a web server. After several attempts the attacker should be blocked, thus rendering the attack unsuccessful. On the other hand there may be situations where the attack is not detectable. These attacks typically make use of common legitimate traffic (such as the SSH attack described earlier in the article) and blocking the traffic is therefore not an option.

Of course other defense techniques that depend on the nature of the attacked operations exist. One can use a replacement operation that is not vulnerable to timing attack or modify the operation in a way that makes the attack impractical. An example of such modification is RSA timing attack protection [13]. The protection resides in *blinding* (multiplying) the cipher-text by a random number before the exponentiation. This process prevents the attacker from knowing what bits are processed and makes bit-by-bit analysis impossible.

Conclusion

Easier ways than side channel attacks exists for subverting the system's security. A targeted attack on the system with specially crafted malicious PDF file or a phishing scam is much easier to prepare and quite likely to succeed. However the history shows us that side channel and timing attacks can be practical and security designers should include them in their threat modeling, especially when designing security sensitive applications and embedded systems that are going to be used for a longer period of time.

MARTIN RUBLÍK, PH.D., CISSP

Martin Rublík is a senior security consultant at BSP Consulting and a lecturer at University of Economics in Bratislava. He received his master's degree at Faculty of Mathematics, Physics and Informatics (Comenius University in Bratislava) in 2005 and his Ph.D. degree at Faculty of Economic Informatics (University of Economics in Bratislava) in 2010. His main area of expertise includes identity management, PKI, smart cards and network security.

References

- [1] Andrea Bittau, Mark Handley, and Joshua Lackey, „The Final Nail in WEP's Coffin – IEEE Symposium on Security and Privacy" Oakland, 2006.
- [2] Marc Stevens and Alexander Sotirov and Jacob Appelbaum and Arjen Lenstra and David Molnar and Dag Arne Osvik and Benne de Weger. (2009) Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. [Online]. *http://eprint.iacr.org/2009/111*
- [3] Luciano Bello. (2008, May) DSA-1571-1 openssl – predictable random number generator. [Online]. *http://www.debian.org/security/2008/dsa-1571*
- [4] Hamid Choukri, David Naccache, Michael Tunstall and Claire Whelan Hagai Bar-El. (2004, April) The Sorcerer's Apprentice Guide to Fault Attacks. [Online]. *http://eprint.iacr.org/2004/100*
- [5] Pankaj Rohatgi. (2010, May) Protecting FPGAs from power analysis. [Online]. *http://www.eetimes.com/design/programmable-logic/4199399/Protecting-FPGAs-from-power-analysis*
- [6] Wim Van Eck. (1985) Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk? [Online]. *http://cryptome.org/emr.pdf*
- [7] Rakesh Agrawal Dmitri Asonov. (2004) Keyboard Acoustic Emanations. [Online]. *http://rakesh.agrawal-family.com/papers/ssp04kba.pdf*
- [8] Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, Caroline Sporleder. Michael Backes. (2010, August) Acoustic Side-Channel Attacks on Printers. [Online]. *http://static.usenix.org/event/sec10/tech/full_papers/Backes.pdf*
- [9] A. Shamir E. Tromer. (2004) Acoustic cryptanalysis. [Online]. *http://www.cs.tau.ac.il/~tromer/acoustic/*
- [10] Sylvain Pasini Martin Vuagnoux. (2009) Compromising Electromagnetic Emanations of Wired andWireless Keyboards. [Online]. *http://www.usenix.org/events/sec09/tech/full_papers/sec09_attacks.pdf*
- [11] Christopher Woods Sergei Skorobogatov. (2012, March) Breakthrough silicon scanning discovers backdoor in military chip (DRAFT of 05 March 2012). [Online]. *http://www.cl.cam.ac.uk/~sps32/Silicon_scan_draft.pdf*
- [12] IOActive Research Labs. (2012, February) I can still see your actions on Google Maps over SSL. [Online]. *http://blog.ioactive.com/2012/02/ssl-traffic-analysis-on-google-maps.html*
- [13] Paul C. Kocher, „Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in Advances in Cryptology—CRYPTO'96, Santa Barbara, California, USA, 1996.
- [14] Dan Boneh David Brumley. (2003, August) Remote Timing Attacks are Practical. [Online]. *http://www.usenix.org/events/sec03/tech/brumley/brumley.pdf*
- [15] David Wagner, Xuqing Tian Dawn Xiaodong Song. (2001, August) Timing Analysis of Keystrokes and Timing Attacks on SSH. [Online]. *http://static.usenix.org/events/sec01/full_papers/song/song.pdf*
- [16] (2007, December) Xbox 360 Timing Attack. [Online]. *http://beta.ivc.no/wiki/index.php/Xbox_360_Timing_Attack*
- [17] Taylor Nelson Nate Lawson. (2010, September) Exploiting Timing Attacks in Widespread Systems. [Online]. *http://www.youtube.com/watch?v=i-djDiBtu93Y*