Timing Attacks on the RSA Cryptosystem

James L. Vaughn

April 28, 2011

Contents

1 Introduction - Why this Attack Matters

In a world where we store vast amounts of our most sensitive data electronically, protecting said data is a major priority amongst those who possess it. Not that keeping private information private is a new idea, it's been around practically forever, but the need to aggregate, store, transfer, and retrieve this data in electronic forms creates many new issues in protecting private data. For example, if someone wishes to purchase an item from an online retailer through an e-commerce web site, many security challenges are faced. First of all, how does the buyer know they are providing their personal information to the correct party and not a third party scammer? How can the buyer be sure that their personal information, assuming it is being sent to the intended recipient, is protected in transit over public network segments from being viewed by a third party? How can the seller ensure that the private information the customer provides them with is used only for the intended billing and is not accessible by unauthorized users or systems? The currently accepted answer to the first two of these questions is a combination of public key cryptography, to verify the identity of the actors and exchange a session encryption key, and symmetric key cryptography, using the exchanged key, to securely transmit the private data. The specific cryptographic protocol used to implement this functionality is Secure Sockets Layer (SSL), a means of encrypting data at the transport layer of the OSI model, which in this case, is application layer hyper-text transfer protocol (HTTP) packets, sent using transmission control protocol (TCP) at the transport layer are encrypted before being sent to the lower network, data link, and physical layers on the network. The third question, while relative to this example, is outside the scope of

this paper. The provided solution is typically implemented using the Rivest, Shamir, and Adleman (RSA) algorithm to cover the public key cryptographic portions. Note that the RSA algorithm is used for many applications and is not limited to its use in securing internet traffic, the provided example was chosen to illustrate one of the many common uses of the RSA algorithm in a way that would illustrate the gravity of a successful attack on it.

2 What is a Timing Attack?

Overview

Timing attacks are a special type of attack on a cryptographic system, which are categorized as side-channel attacks. In these types of attacks, the actual cryptosystem itself isn't attacked, but the implementation of the cryptosystem is manipulated or used in some un-intended way to gain information about the details of the specific instance of the cryptographic algorithm. This can be thought of installing high-security door locks on your house, reinforcing your door, and performing all industry standards to prevent forced entry, but overlooking that the window was left slightly open and, with some effort, could be used to bypass the door (still a completely secure door) completely. Timing attacks exploit the fact that the runtime of a specific algorithm on a computer system can vary a relatively large amount, based on the specific procedures that are performed, when provided with different inputs. While, in the purely mathematical sense, the RSA decryption equation (Equation 1) performs the exact same steps for every input, many common software implementations

of this perform some optimizations to improve performance of the system, as certain steps are not needed when certain inputs are given or certain intermediate results are reached. It is because of this that the capability to perform this exploit exists, since an implementation based purely on the mathematical definition would not provide the required performance variances, more on that later though. To execute this attack, the attacker carefully selects a set of inputs to provide the system and analyses the amount of time the system takes to generate a response. Given the ability to provide a system running the algorithm with an input, receive the output, and measure the computational time somewhat accurately, an attacker, with some knowledge of how the algorithm works, can determine the private values, such as the decryption exponent in the RSA algorithm, used in decryption. Once a value like the decryption exponent is obtained, the attacker can decrypt data sent to the intended recipient as if he or she were the intended recipient. The attacker hasn't actually compromised the cryptosystem itself, but has used other means to obtain private data used by the cryptosystem. This is a particular issue since there is no sure way to determine that the private values are now known to an unauthorized third party. This means that, after a successful attack, the attacker can collect encrypted data and decrypt it for the foreseeable future, or at least until the certificate the server is using expires in the case of SSL. The attacker can lurk in the shadows of the internet stockpiling large amounts of sensitive data to be used in possibly nefarious activities at a later point in time. Furthermore, in the RSA cryptosystem, knowledge of the decryption exponent d would allow an attacker to digitally sign documents that would, through accepted methods of verification, show that

the document was signed by the attack victim rather than the attacker.

$$m = c^d \, mod \, n \tag{1}$$

Attack Medium

This type of attack can be executed on the local computer, itself, in which the cryptographic algorithms are running or, assuming the computer is accessible via a network connection, from anywhere that can connect to the input point of the system to be attacked. The former can be executed in many ways, especially considering the current trends in shared server hosting and the move towards server virtualization. In the case of shared hosting on a single physical server, resources are isolated by the operating system and provided to users and their services as if the entire server belonged to them, even though they are only receiving a fraction of the available physical resources. This is particularly common in Linux based web hosting environments since dedicating an entire server with the processing power common in today's (even low-end) servers would be overkill for hosting many web sites and applications. While the operating systems that provide this type of environment are typically secured and locked down, multiple users and their services still exist on one physical server, separated only by software, not hardware, as is the case with individual servers. With that in mind, even with the user isolation, information about the other users of the system is intrinsically able to be determined (not to mention the introduction of the possibility of other non-timing based side-channel attacks, such as exploiting POSIX

interprocess communication functions or vulnerabilities in file system security to directly obtain the secret information). Users can see what share of the server's physical resources (processor time, memory, network bandwidth) they have and can, therefore, conclude that the remaining shares are not theirs and must belong to another user. Exploiting this type of weakness yields results that are much more accurate and precise than that of those obtained when a timing attack is performed over a network. When a timing attack is performed over a network there are many more factors that affect the amount of time that elapses between when a client first sends a request and when the client receives the result than just the amount of time the remote system takes to perform the algorithm. The attacker must realize and account for the multitude of variable factors that impact the amount of time it takes to transmit the data packets across a network, including network latencies introduced by switch and router hops, as well as the possibility that the individual data packets could each take completely different routes, each with different latencies, while in transit. Furthermore, since the target attack is presumably accessible by multiple users, results will vary more as the utilization of the target increases since it is more likely that the process or thread that processes the attacker's requests will be suspended during execution by the operating system's scheduler to facilitate other processes and threads, hence more accurate results can be obtained by the attack during periods of low utilization of the target system. All of this variance requires the attacker to perform each step of an attack multiple times and perform statistical analysis on the results obtained in order to gain a normalized result. Keep in mind that in some cases, network latency variances will prevent usable

results. Successful attacks are more likely to occur the closer (relative to network hops) the attacker is to the server.

3 Details of Exploit

In the implementation of the RSA decryption function as a computer program, certain calculations are skipped under certain conditions in order to make the function more efficient. The addition of an `if` statement in a high-level language compiles down to a simple `branch` command in assembly code. When one analyzes the amortized runtime of many calls to the compiled function, the runtime of the decryption algorithm is decreased approximately equal to half of the amount of time used to calculated the calculation that is controlled by the `if` statement. Since the inclusion of this `if` statement does not affect the actual outcome of the computation, it does not seems like it would make the system any more insecure. That's what makes side-channel vulnerabilities, such as timing attacks, difficult to detect. The cryptosystem used itself is not vulnerable, but the manner in which it is implemented allows for analysis of how it runs that can reveal secrets that were not intended to be made public. There are two common algorithms used to implement RSA decryption functions that are vulnerable to a form of timing attacks. Firstly, the "square and multiply" algorithm (an implementation is shown in C++ source) includes a perfect example of the exclusion of unnecessary computations in order to increase efficiency. In this algorithm, the binary representation of the decryption exponent, d, is analyzed bit by bit and, if a given bit of d is 1, the statement after the if executes, requiring a multiplication and a

modulus operation to be performed. The way this multiplication is performed is referred to as Montgomery multiplication. The vulnerability comes from the fact that the Montgomery algorithm calls for the reduction of the multiplication so it remains a valid number working in mod n. The attack is executed by injecting values of encrypted messages that double in such a way that the if statement is always true (since multiplying in 2 in base 2 is equivalent to a shift operation, the leading bit, at position i will remain 1) and the Montgomery algorithm will always be performed. The attacker can then determine the values of the bits of d based on the increased runtime incurred by performing the reduction step of Montgomery multiplication.

```
1  private long RSA_Decrypt_Square_and_Multiply(long c){
2          long x = c;
3          for(long i = 0; i < n; i++){
4                  x = pow(x, 2.0) % n;
5                  if(memcmp((&d+i),(void*)1,1){
6                          x = x*c mod n;
7                  }
8          }
9          return x;
10 }
```

Another vulnerable decryption algorithm involves the use of the Chinese Remainder Theorem. This method of decrypting a message encrypted with the RSA algorithm eliminates the attack performed on the square and multiply algorithm, but introduces a new vulnerability. This vulnerability is based on the same principle that steps can be skipped if not needed without affecting the output. Values used to perform the attack are chosen to be smaller and bigger than the public n, the bits of the decryption exponent d can then be determined by repeated execution of the function with numbers of that pattern. The modulus command

that is executed based on the bits of the decryption exponent increases a timing variance that causes the vulnerability of this decryption algorithm. As before, many samples must be submitted and the results analyzed (including statistically to normalize and eliminate network latency and other variations) to gain meaningful data from the attack.

4 Defense Against Timing Attacks

There are many ways to eliminate the threat of timing attacks, the most straightforward approach being simply to not let the function that implements the vulnerable algorithm return a result until a time quantum has expired. While that approach is simple and completely masks any variance in computation time cause by the provided input (thus preventing the attacker from being able to perform the timing attack), it also greatly decreases the overall efficiency of the RSA decryption algorithm since the value of the time quantum must be equal to the worst case runtime of the algorithm in order to facilitate all possible scenarios. The two scenarios, the vulnerable implementation of decryption and the time-quantum based implementation, represent two extremes, favoring efficiency and security respectively. For this reason, other solutions that provide a reasonable balance of timing attack resistance and resistance have been developed. One such solution is referred to as "blinding". This method chooses a random number, r, and raises it to the power of the encryption exponent, multiplies it by the incoming message, c, and mods by n. This effectively just multiples the actual message by $r \, mod \, n$ and is divided out later in the algorithm. This means that the input multiplied by the random r is what impacts the timing of the algorithm, not just the

input, so the attacker loses the ability to inject specific values.

5 Modern Security Issues/Examples

If an attacker successfully executed a timing attack on, for example, an e-commerce website and found the private decryption exponent, the attacker could not only decrypt incoming messages, which would include customer personal information, such as credit card numbers, addresses, etc..., but the attacker would also be able to impersonate the e-commerce website (assuming they could modify the shopper's DNS records to point to their server), keeping the remote validation features of SSL encryption intact and the end-user being clueless they weren't actually talking to the intended e-commerce website. This could lead to the attacker requiring additional information in a fraudulent check-out scheme to gather more information about users, in other words, the attacker could implement a phishing website that was shown to be the actual e-commerce website by the SSL certificates. Another issue is the prevalence of botnets, large clusters of computers infected with viruses that allow for remote control of said computers. The number of infections is often very large and would provide the attacker with a means of performing a distributed timing attack which would make the attack quicker to perform and also, since the requests would be coming from many different hosts, bypass distributed attack detection and prevention systems commonly used to shield web-servers from attack. A commonly used cryptographic package, OpenSSL, had default settings in the 1990's that caused it to be susceptible to timing attacks. These settings have since been changed to include the blinding technique discussed previously as a default setting.

References

[1] David Brumley and Dan Boneh. Remote timing attacks are practical. *Comput. Netw.*, 48:701–716, August 2005.

[2] Burt Kaliski. Timing attacks on cryptosystems. *RSA Laboratories' Bulletin*, 1996.

[3] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, London, UK, UK, 1996. Springer-Verlag.

[4] Wade Trappe. *Introduction to cryptography : with coding theory.* Pearson Prentice Hall, Upper Saddle River, N.J, 2006.

[5] Wing H. Wong. Timing attacks on rsa: revealing your secrets through the fourth dimension. *Crossroads*, 11:5–5, May 2005.