20 years of Bleichenbacher's attack

Authors
Gage Boyle, MSc (Royal Holloway, 2018)
Kenny Paterson, ISG, Royal Holloway

Abstract

The HTTPS protocol is a vital tool in safeguarding the security of our personal and business lives. This protocol ensures that important data such as financial information, intellectual property and login credentials are encrypted and integrity protected as they travel across the Internet. Ultimately this means that a malicious observer is unable to obtain, view, use or sell this important data. To both technical and non-technical users, the presence of "HTTPS" at the start of the website URL will provide enough confidence to consider entering sensitive information such as bank or credit card details. However, in this article we will explain how even websites owned by the most reputable organisations may be exposed to a 20-year-old attack if HTTPS is not properly implemented.

Around 33% of internet servers were found to be vulnerable to this attack in 2016, and both Facebook and PayPal remained vulnerable in 2017. Furthermore, related weaknesses in HTTPS implementations are still regularly being discovered. As a result, a secure TLS implementation is imperative for all organisations looking to maintain their business reputations and sensitive intellectual property. [a]

[a] This article is published online by Computer Weekly as part of the 2019 Royal Holloway information security thesis series `https://www.computerweekly.com/ehandbook/The-exploitation-of-flaws-in-the-HTTPS-protocol`. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full thesis is published on the ISG's website at `https://www.royalholloway.ac.uk/research-and-teaching/departments-and-schools/information-security/research/explore-our-research/isg-technical-reports/`.

Introduction

Secure Sockets Layer/Transport Layer Security (SSL/TLS) is a cryptographic protocol that establishes and maintains a secure communication channel between two parties (typically a client and a server). When used in conjunction with HTTP, the result is the HTTPS protocol mentioned above. For many years, the use of RSA encryption to transport session keys has been a popular option within TLS. However, since 1998, implementations of TLS using RSA-based encryption have been and continue to be vulnerable to an adaptive chosen ciphertext attack - originally known as the "Million Message Attack" due to Daniel Bleichenbacher. At a high level, the attack can capture a target RSA ciphertext from the network. It then carefully chooses variations of the ciphertext and takes advantage of small amounts of information leakage when those variants are decrypted to reduce the number of possible plaintext options. Once the attack is complete, the number of plaintext options has been reduced to one, with that one containing the target session key.

In this article, we will focus specifically on how the attack is mounted in the context of TLS, but the issues we highlight go far wider.

How does Bleichenbacher's attack work?

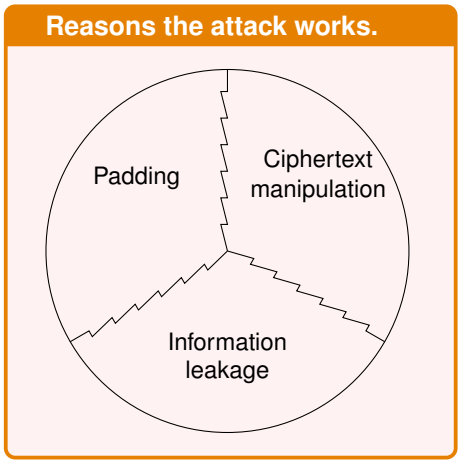
There are three main reasons the attack is possible.

The first reason is a direct result of a standardised and popular padding scheme (PKCS#1 v1.5) that is used in conjunction with RSA in TLS. In order to encrypt using RSA, extra (random) data must be

added to a message to ensure it is long enough to be encrypted using the public key parameters. The PKCS#1 v1.5 padding scheme requires this data to follow a strict format. This format effectively creates upper and lower numerical bounds on the plaintext to be encrypted when it is interpreted as a number. The attack seeks to sharpen these bounds to the point where only one plaintext candidate remains.

The second reason the attack is possible arises from an attacker's ability to manipulate RSA plaintexts in a certain controlled way by modifying only the ciphertext. This is sometimes called the homomorphic property of RSA. Although an attacker would not know what value an unknown plaintext has been manipulated to, the fact that the attacker can choose how the plaintext is manipulated provides the second puzzle piece.

The third reason is due to commonly observed information leakage from RSA decryption code. Specifically, an attacker can often tell whether an unknown encrypted plaintext is padded correctly or not, just by submitting a ciphertext for decryption and observing the reaction of the decrypting code (for example, as implemented on a TLS server). To be concrete, that code may send an error message if the padding is invalid, or the time taken to decrypt may vary depending on whether the padding is correct or not. The way in which a network connection is terminated may even be different depending on the validity of the padding. In more advanced versions of the attack, the inner workings of decryption (and padding checks) may be revealed to an attacker who can indirectly observe the decryption process's cache access pattern. We refer to this information leakage in general as being a *padding oracle*.



Bringing these three elements together, suppose a server is using the PKCS#1 v1.5 padding scheme for RSA encryption. Then an attacker can manipulate the original ciphertext (and therefore the unknown plaintext), and then send this to the padding oracle that is provided by the server. This oracle will then inform the attacker as to whether the new plaintext is padded correctly (or not). If it is padded correctly, the new unknown plaintext can be further manipulated and sent to the oracle once more. This process continues, and with the most efficient versions of Bleichenbacher's attack, in general, each modified plaintext that is found to be correctly padded allows the attacker to halve the size of the interval containing the real plaintext. As a result, the set of possible plaintexts gets smaller and smaller, until only one possible value remains. In the context of TLS, this last plaintext immediately yields the session's master key. If this session went on to transmit sensitive information such as bank or credit card details, the information would be easily accessible to an attacker.

Can it get any worse?

The short answer is yes. The efficiency of Bleichenbacher's original attack has been improved greatly over the last 20 years, and newly found weaknesses and vulnerabilities in implementations of TLS and other protocols have provided a steady stream of practical padding oracles.

Such improvements to the efficiency of the attack have also led to the practical possibility of forging RSA digital signatures. This potentially more devastating attack requires a more time-consuming initialisation step for an attacker. However, it does enable them to impersonate the vulnerable server without ever needing to know the private signature key. A practical implementation of such an attack was carried out against Facebook in 2017 as a means of demonstrating that weaknesses in TLS implementations represent real threats to ordinary users. Fortunately, Facebook have now fixed their vulnerable TLS implementation and removed the information leakage that provided a padding oracle.

However, the padding oracle demonstrated in Facebook's TLS implementation is just the tip of the iceberg; over the last couple of years many padding oracles have been discovered in cryptographic code.

Vulnerable products/vendors include F5, Citrix, Radware, Cisco ACE, Cisco ASA, Erlang, Bouncy Castle, WolfSSL, Java Secure Socket Extension, MatrixSSL, Palo Alto Networks, IBM GSKit, Unisys ClearPath MCP, Symantec IntelligenceCenter, Symantec SSL Visibility, Cavium Nitro, Fortiguard SSL Deep Inspection and Haskell-TLS.

Fixes have been made available for all the above vulnerabilities after appropriate disclosure from the researchers concerned, with one exception: support for the Cisco ACE devices was discontinued several years ago and so no update has been made available. Moreover RSA encryption is the only method available for establishing TLS sessions in those Cisco ACE devices. The upshot is that there is now no way to use those devices securely.

In addition to these practical applications of the attack, there are modified versions that have exploited code bugs and allowed for extremely efficient attacks against unpatched OpenSSL implementations. Cross-protocol and cross-version attacks are also possible. Here the targeted server does not even need to be vulnerable. Instead it simply needs to share its RSA public key with another server (or another service on the same server) that is vulnerable. In modern IT architectures, it is quite common for multiple servers and/or services run by the same business entity to share public keys, and therefore such a situation is not unrealistic.

So why does Bleichenbacher's attack still work?

The main reason the attack works at all is due to the difficulty of securely implementing RSA in combination with PKCS#1 v1.5 padding whilst avoiding all possible attack vectors. The community's state of knowledge on how to do this has evolved over the last 20 years, and the TLS standard has been updated several times to reflect this changing state. At the same time, there is a lag between what the standard says and what implementations actually do. Even relatively new implementations such as Facebook's can reintroduce known security issues. That being said, the TLS standard could have taken a different approach. Instead of advising on how to patch implementations to prevent attacks, TLS could have switched to using a different cryptographic construction entirely, one known to provably resist Bleichenbacher-style attacks such as RSA-OAEP.

Nonetheless, even if the standards are not quite where they should be, there are several other reasons as to why Bleichenbacher-style attacks still work:

- *Security systems are not patched quickly enough when vulnerabilities are discovered.* In 2016 there were some devastating vulnerabilities discovered in the OpenSSL code that allowed for the completion of a Bleichenbacher-style attack in under 1 minute on a single workstation. The attack was disclosed and fixed quickly. However, vulnerability scanning that took place many months after the security patch became available showed that 26% of HTTPS servers were still vulnerable.
- *The same cryptographic key is being used for multiple purposes across multiple servers.* For example, the same key may be used for RSA encryption and digital signing using RSA. This means that if the TLS implementation of encryption is vulnerable then so is the digital signing implementation. Additionally, if one server is vulnerable then so are all servers that it shares a key with. In modern business architectures, this means that one vulnerable server can put the whole corporate network at risk.
- *Legacy systems are poorly maintained.* We need legacy systems to support older services. However, they can pose a security risk to more modern systems that share information or services with them. As these older systems get replaced, they may be left active for backwards compatibility reasons but may soon be forgotten about and not subject to proper patch management. Such a situation can mean that these older architectures quickly become outdated and vulnerable to newly discovered bugs. Furthermore, as mentioned above with Cisco ACE devices, it may be that patches are simply not available and therefore older devices are no longer secure to use.

- *Despite the availability and standardisation of other mechanisms, RSA key transport is still being used to set up TLS sessions.* Clearly TLS implementations should be kept up to date. However, it is the way in which an attacker can manipulate an RSA encrypted plaintext that allows a Bleichenbacher-style attack to be successful. If RSA key transport is disabled, then such an attack is no longer possible.

What should we do about it?

Follow appropriate standards. It is always important to follow appropriate standards when it comes to building secure implementations of systems. The same can be said of the PKCS#1 v1.5 padding scheme standard. There are alternatives such as the RSA-OAEP padding scheme. However, unless very carefully implemented, it too is vulnerable to a chosen ciphertext attack which is comparatively more damaging. Therefore, instead of trying to avoid PKCS#1 v1.5 padding, we should look to ensure that all appropriate countermeasures have been taken to thwart Bleichenbacher-style attacks. By properly following the appropriate standards, such a task is achievable with moderate effort.

Patch systems quickly. It is common for a TLS implementation to be based on some open source software such as OpenSSL. However, irrespective of where a TLS implementation comes from, bugs will be discovered, and attackers will take advantage. As a result, a well-refined patch management system is a vital means of dealing with the modern threats on the Internet.

Effectively manage cryptographic keys. Good key management is extremely important, and that also means good key separation. As such, a cryptographic key should only be used for its intended purpose, and ideally only have one purpose. So, keys used for encryption should not be used for digital signatures, and keys used on one server should not be used on another. This can be an expensive requirement to meet, but it will prevent digital signature forgeries, as well as most of the more recently discovered cross-protocol/cross-version attacks, including those targeting TLS 1.3 (the newest standard from the TLS series).

Understand the entire system architecture. We must have a clear view of the systems in play and how they interact. As sections of an architecture are deprecated, they still need to be part of the patch management program, and once they are no longer needed or no longer possible to secure, they should be completely removed. Furthermore, on older architectures, weak encryption over no encryption is not necessarily the best option, and these insecure architectures can put more modern secure systems at risk. As a result, all aspects of an architecture must be well-maintained.

Consider alternative algorithms for TLS key establishment. For TLS key establishment, a more sensible approach would be to utilise (Elliptic Curve) Diffie-Hellman key agreement in combination with RSA signatures. Not only does this make the TLS protocol immune to all known Bleichenbacher-style attacks, it also has the added benefit of providing forward secrecy, meaning that should the server's RSA private key be exposed at some point, then any previously completed sessions remain secure. The use of RSA encryption for key transport has been deprecated in TLS 1.3, but this does not mean the protocol is no longer vulnerable to Bleichenbacher-style attacks, particularly if cryptographic keys are shared across protocol versions.

The future

Looking to the future, the use of RSA encryption is gradually becoming less common. This will continue to be the case as more servers move to TLS 1.3. However, these servers will still need to support earlier versions of the protocol for backwards compatibility reasons. Therefore, the need to maintain RSA encryption/decryption capabilities will remain for many years to come. It is also likely that alternative means of obtaining side-channel leakage from RSA implementations will continue to be discovered by determined researchers. This could lead to additional oracles, and ultimately other ways of enabling Bleichenbacher-style attacks.

To conclude, here are five key takeaways:

1. Follow standards carefully for new and existing implementations of TLS.
2. Apply patches quickly and thoroughly to all aspects of the security architecture.
3. Avoid export and weak ciphersuites within TLS (and disable any instances of SSL).
4. Account for and maintain all aspects of the security architecture and remove elements when they are no longer required or no longer secure.
5. Take public key management seriously and, where possible, do not share cryptographic keys across multiple TLS versions and servers.

Biographies

Gage Boyle graduated in 2018 from Royal Holloway, University of London with an MSc in Information Security - where he was awarded with the outstanding MSc student prize for the highest grades that academic year. Prior to this, he graduated from the University of East Anglia with a 1st class degree in Mathematics, also receiving the prize for distinguished performance.

Kenny Paterson is a Professor of Information Security in the Information Security Group at Royal Holloway, University of London. His research on the security of TLS has received significant media attention, and his influence has helped to spread the adoption of modern encryption schemes such as those supported by TLS 1.2 and 1.3. Kenny is also a co-founder of the Real World Cryptography series (strengthening the links between academia and industry), and he has been awarded several prizes for his high quality research - most recently winning best paper awards at both CHES and IMC in 2018. Among other things, Kenny is currently co-chair of the Crypto Forum Research Group (CFRG), a body providing cryptographic advice to the IETF, and editor-in-chief of the Journal of Cryptology.

Series editor: S.- L. Ng