illusoryTLS: Nobody But Us Impersonate, Tamper, and Exploit

Alfonso De Gregorio, 25-26 November 15, ZeroNights, Moscow

Web PKI is Fragile: The sorrow state of the infrastructure we daily entrust our business upon

Здравствуйте, уважаемые коллеги, дорогие друзья! Очень приятно вас видеть! Good evening. Thank you for the opportunity to speak with you today, which is also my first time at ZeroNights.

And first times can be unforgettable; they should be unforgettable: The first love; The first concert; The first kiss; And the first Man-in-the-Middle attack. I remember the supposedly first time when I was Man-in-the-Middle attacked. Do you remember yours?

I had configured a client to pull logs from a web server via a SSL connection over the Tor network. After some time, as new circuits were established, a new pull request failed to connect due to a certificate error. I dumped the server certificate I was getting from the network endpoint and I found it to differ from the one I was expecting to see. And in that moment I realized, "Oh my gosh! I'm getting man-in-the-middle attacked!"

And the next thing I say to me is “Wow! This is so cool! This is so cool! How many security philosophers have the opportunity to study their attacked network flow from the inside out?” [JBT]

First times are unforgettable --- if only we could notice them. I hold dear the memory of a true positive, but this event is possibly outnumbered by the false negatives.

That is to say that today's Web PKI is fragile. We are, as a security community, witnessing with increasing frequency incidents, vulnerabilities, dramas, and failures of the infrastructure we daily entrust our business upon.

- China Internet Network Information Center (CNNIC), 2015
- Lenovo, 2015
- National Informatics Centre of India, 2014
- ANSSI, 2013
- Trustwave, 2012
- Türktrust, 2011-2013
- DigiNotar, 2011
- Comodo, 2011
- Verisign, 2010

In the aggregate, these and other missteps, malicious or otherwise, allowed the impersonation of a number of high profile websites and granted to the impersonating entities the ability to spy on a sizable number of unsuspecting site users. No fond memories for those of us who remain oblivious to their first times.

This is already unfortunate. If "No silent failure" is the pinnacle goal of security engineering [DG2], undetected impersonation is no measure of success in our engineering effort. As if that were not enough cause for concern, now we are discussing about building cryptographic backdoors. Go figure.

I work at the intersection of software security and security software, exploring, and trying to contain, the space of unanticipated state. [LS] I want crypto to be open and robust. To me it seems that secure backdoor is no more than an oxymoron, like almost safe. Imagine your special somebody asks you: “Darling, are you safe?” Answering "almost" you would not provide a satisfactory answer. But it is about cryptographic backdoors that I would like to speak with you today.

This is a timely topic often debated as matter for a government to legislate on. For today, I propose you to approach it as a technical space that some entities might have practically explored regardless of the policy framework. [BULLRUN] If somebody in the belief, however questionable, that a "technically secure backdoor with a golden key" is possible had built one, would we be able to notice if our communications were being exploited?

Before delving into the technical aspects, I would like to poll your opinion on the nature of backdoors. If you feel inclined, please, raise your hand and I'll recognize your view. How many of you think that backdoors can be asymmetric? How many of you think that backdoors can be embedded in data?

The common view is that backdoors are symmetric in nature and require the presence of malicious logic in the target system code base. That is to say, everyone with knowledge about the internals of the backdoor can exploit it and, given enough skills and effort, code review can spot their presence.

The scarce research published in this field challenge this view. Backdoors can be asymmetric. Their complete code does not enable anyone, except those with access to the associated key-recovery system, to exploit the backdoor. And backdoors can be planted in data. Or, to paraphrase a popular quote on homoiconicity of some programming languages: backdoor is data, data is backdoor.

I ask you to consider the current Web PKI. The entire X.509 PKI security architecture falls apart, if a single CA certificate with a secretly embedded backdoor enters the certificate store of relying parties. Have we sufficient assurance that this did not happen already?

While I am not in the position to know if this already happened, I decided to explore this scenario from both an experimental and speculative point of view.

illusoryTLS: Nobody But Us Impersonate, Tamper, and Exploit

In November 2014 I submitted an entry called illusoryTLS to the Underhanded Crypto Contest. [UCC] "The Underhanded Crypto Contest is a competition to write or modify crypto code that appears to be secure, but actually does something evil."

illusoryTLS is an instance of the Young and Yung elliptic curve asymmetric backdoor in RSA key generation. The security outcome is the worst possible outcome, because the backdoor completely perverts the security guarantees provided by the TLS protocol, allowing the attacker to impersonate the endpoints (i.e., authentication failure), tamper with their messages (i.e., integrity erosion), and actively eavesdrop their communications (i.e., confidentiality loss).

The design choices for this backdoor reflect the following threat model. The backdoor designer can:

1. "insert vulnerabilities into commercial encryption systems, IT systems, networks and endpoint communications devices used by targets".
2. "influence policies, standards and specifications for commercial public key technologies". [BULLRUN]
3. Interfere with the supply-chain. [EFF]
4. Disregard everything about policy. [RJA]
5. Or she is simply in the position to build the security module used by the Certification Authority for generating the key material.

illusoryTLS consists of three modules:

a A HTTPS client;
b A TLS server;
c A certificate store.

The code is part of network-simple-tls --- an Haskell library for simple network sockets usage patterns using TLS security --- and is used as-is, without any modification. [NST]

For the sake of simplicity, and without loss of generality, the server implements an Echo service over TLS. The client sends a basic HTTP command over a TLS channel an awaits the server response.

From a network security perspective, the architecture of illusoryTLS is similar to the architecture of standard Web technology, where browsers --- or other HTTP clients --- and servers rely on a database of public-key certificates (i.e., certificate stores) as trust-anchors in the validation of the certification paths.

If the client and server code is contributed by an open-source project and it is used as-is, where is the backdoor?

I have embedded one in the public-key certificate of the CA the communicating entities rely upon to mutually authenticate. And the backdoor is highly robust against reverse-engineering.

In particular, the upper order bits of the RSA modulus encode the asymmetric encryption of a seed generated at random. The same seed was used to generate one of the RSA primes of the CA public-key modulus. Hence, the RSA modulus is at the same time a RSA public key and an ciphertext that gives to the backdoor designer, and only to the designer, the ability to factor with ease the said modulus.

No backdoor was slipped into the cryptographic credentials issued to the communicating endpoints.

The backdoor hidden in illusoryTLS implements a security notion that is twenty years old. Adam Young and Moti Yung introduced the notion of secretly embedded trapdoor with universal protection (SETUP) at Crypto '96 [YY96]. illusoryTLS is an instance of the Young and Yung elliptic curve asymmetric backdoor in RSA key generation. [YY] The work by Adam and Moti expands on the research they published in the proceedings of Selected Area in Cryptography 2005. [YY05] The same authors present a working implementation of their attack at http://www.cryptovirology.com/

illusoryTLS has some noteworthy properties:

NOBUS (Nobody But Us): The exploitation requires access to resources not embedded in the backdoor itself. In this case the secret resource is an elliptic-curve private key. Hence, the vulnerability can be
exploited by the backdoor designer and by whoever gains access to the backdoor elliptic-curve private-key or to the associated key-recovery system. On a related note: How many of you believe that it is possible to forbid an enemy intelligence organization from gaining access to a private key? I expect that those of you who believe that this is possible are inclined to regard secure backdoors as also possible. And I expect that those of you who believe that any measure to prevent disclosure of the private-key will be circumvented are inclined to regard backdoors as inherently insecure.

Indistinguishability: As long as a computational hardness assumption called Elliptic-Curve Decision Diffie-Hellman (ECDDH) holds, the backdoored key pairs appear to all probabilistic polynomial time algorithms like genuine RSA key pairs. Therefore black-box access to the key-generator does not allow detection.

Forward Secrecy: If a reverse-engineer breaches the key-generator, then the previously stolen information remains confidential (secure against reverse-engineering).

Reusability: The backdoor can be used multiple times and against multiple targets.

Impact: Or, why one rotten apple spoils the whole barrel

It is interesting to look to the impact and the implications of this rather subtle backdoor. The designer can use the private-key recovered by factoring the CA modulus to break the TLS security guarantees
at will, exposing the relying parties to a variety of attacks. They range from impersonation (i.e., authentication failure), to message tampering (i.e., integrity erosion), to active eavesdropping of encrypted communications (i.e., confidentiality loss), whenever the attacker mounts a MitM attack. Not necessarily for the first time.

The attacker does not need to have access to any private key used by system actors, namely the Certification Authority and the communicating endpoints. The attack works without any need to tamper with the communicating endpoints. Yet, the backdoor designer needs to retain control over the key-generation of the target RSA modulus.

Is the malicious implementer a threat mitigated by IT product security certifications? Later I will review briefly the key-generation security requirements set forth in the most relevant protection profiles in the Common Criteria certification process and demanded by industry organizations and associations (i.e., CA/Browser Forum) and argue why this is not the case.

First I would like to draw your attention on the security impact that such backdoor would have on the Web X.509 PKI. The presence of a single CA certificate with a secretly embedded backdoor in the certificate store renders the entire TLS security fictional.

You may have already found out by experience that one rotten apple spoils the whole barrel. The rotting fruit manufactures and releases ethylene, a hormone transferred in gaseous phase that promotes fruit ripening. In the X.509 setting, universal implicit cross-certification is the ethylene of trust. Almost prophetically the structural formula of the ethylene molecule, C2H4, proves to be a schematic diagram
for cross-certification.

"Cross certification enables entities in one public key infrastructure (PKI) to trust entities in another PKI. This mutual trust relationship [should be] typically supported by a cross-certification agreement
between the certification authorities (CAs) in each PKI. The agreement establishes the responsibilities and liability of each party." [MS]

"[An explicit] mutual trust relationship between two CAs requires that each CA issue a certificate to the other to establish the relationship in both directions. The path of trust is not [hierarchical] (neither of the governing CAs is subordinate to the other) although the separate PKIs may be certificate hierarchies. After two CAs have established and specified the terms of trust and issued certificates to each other, entities within the separate PKIs can interact subject to the policies specified in the certificates." [MS] But this is just in theory.

In practice, "most current PKI software employs a form of implicit cross-certification in which all root CAs are equally trusted, which is equivalent to unbounded cross-certification among all CAs. This means that, for example, any certificate can be trivially replaced by a masquerader's certificate from another CA, since both CAs are equally trusted... With the implicit universal cross-certification that exists in this environment, the security of any certificate is reduced to that of the least trustworthy CA, who can issue a bogus certificate to usurp the legitimate one, at the same level of trust." [PG] The bogus certificates issued to Egypt-based MCS holdings and installed in a man-in-the-middle proxy [MCS] or the Superfish MitM adware [SF] are just the most recent examples of exploitation for this obvious vulnerability.

That is to say that universal implicit cross-certification, or the lack of constraints on the signer's authority, is to security, or the absence of unmitigatable surprises [DG2], what ethylene is to fruit rotting: it makes the whole PKI as weak as its weakest link. And a CA certificate with a secretly embedded backdoor with the promise of exclusive exploitation may attract multiple attackers.
With multiple attackers going after a technical opportunity, the Nobody But Us paradigm would quickly turn into an Everybody Else But Me concern for those left behind. The race to exploitation would self-reinforce, leading to a me-too effect. A scenario that would negate any meaningful security whatsoever. And "No Security" it is a pretty good album, by the way.

Therefore, when dealing with this class of attacks in the context of X.509 Web PKIs, it might be not sufficient to avoid outsourcing the key generation. It becomes essential also to have assurance about the security of each implementation of vulnerable key-generation algorithms employed by trusted credential issuers.

At this time, Mac OS X Yosemite has 211 CA certificates installed. [OSX] A similar number of certificates is present in the Firefox, Google Chrome, and Microsoft Windows certificate stores. Have we sufficient assurance about the hundreds CA certificates we daily entrust our business upon?

The threats posed by a malicious implementer are not convincingly mitigated by current IT product security certification. The CA/Browser Forum requires publicly trusted certificates to be issued in compliance with the European Standard EN 319 411-3. According to the standard, the CA key generation shall be carried out within a device that meets the requirements identified by some approved protection profiles. The CEN Workshop Agreement 14167 Part 2, 3, and 4 are three of those protection profiles. The assurance level for these protection profiles is EAL4 augmented. The augmentation results from product adherence to three additional requirements ADV_IMP.2, AVA_CCA.1, and AVA_VLA.4. These are focused on assessing the presence of vulnerabilities in the Target of Evaluation (TOE) and on guaranteeing that the implementation representation is an accurate and complete instantiation of the TOE security functional (TSF) requirements. Special emphasis is placed on identifying covert channels and on estimating their capacity --- which is relevant, because SETUP attacks makes use of the key-generation as a covert channel for itself. However, the certification process requires the developer the performance of the vulnerability assessment and documentation tasks. Of course, this conflicts with our threat model. The evaluator is left with the documentation and the implementation representation provided in fulfillment of the certification. Arguably, at the required Evaluation Assurance Level the assessment may fail to rule out the presence of backdoors in the key generation. Formal methods are in fact required only at the two highest levels (EAL6 and EAL7). And the level of detail of the implementation representation may render backdoor detection unlikely (e.g., HDL at design time, netlist at fabrication time as candidate representations).

So, a key takeaway I would like to leave you with today is that, as long as the implementations of RSA --- or, more generally, algorithms vulnerable to this class of attacks --- used by trusted entities (e.g., Certification Authorities) cannot be audited by relying parties (e.g., X.509 end-entities), any trust-anchor for the same trusted entities (e.g., root certificate) is to be regarded as a potential backdoor.

That is to say that as long as the implementation of algorithms adopted by CAs and vulnerable to this class of backdoors cannot be audited by relying parties, the assurance provided by illusoryTLS (i.e., none whatsoever) is not any different from the assurance provided by systems relying upon TLS and RSA certificates for origin authentication, confidentiality, and message integrity guarantees.

A number of mitigation exist, from key pinning [HPKP] to proper explicit cross-certification. But none of them is widely deployed yet.

A Backdoor Embedding Algorithm: Elligator turned to evil

The subtleness of a backdoor planted in a cryptographic credential resides in the absence of malicious logic in the systems whose security it erodes. This is the reason why you will not find anything amiss in the illusoryTLS code base --- and neither the contest judges did it. Admittedly funnier, though, is the backdoor embedding algorithm. So, today I would like to share a novel way to embed an elliptic-curve asymmetric backdoor into a RSA modulus.

Few weeks after my submission to the Underhanded Crypto Contest, Ryan Castellucci shared on GitHub [RC] his variant for such attack.

The idea is to:

1. Embed a Curve25519 public-key into the key-generator;
2. Generate an ephemeral Curve25519 key at random;
3. Compute a shared secret using Elliptic Curve Diffie-Hellmann;
4. Use the shared secret to seed a cryptographically secure pseudo-

random number generator (CSPRNG) based on AES run in CTR mode,;

5. Generate a normal RSA key using the seeded CSPRNG,;
6. Replace 32-bytes of the generated modulus with the ephemeral Curve25519 public-key;
7. Use the original prime factors to compute two new primes leading to a new modulus embedding the ephemeral public-key ;
8. Output the RSA key .

To recover the target private key the attacker:

1. Extracts the ephemeral Curve25519 public-key from the target modulus;
2. Computes the shared secret via Elliptic Curve Diffie-Hellmann and using the private-key associated to the public-key embedded in the key-generator;
3. Uses the shared secret to seed the cryptographically secure pseudo-random number generator (CSPRNG) based on AES run in CTR mode;
4. Generates a normal RSA key using the seeded CSPRNG,;
5. Replaces 32-bytes of the generated modulus with the ephemeral Curve25519 public-key;
6. Uses the original prime factors to compute two new primes leading to the target modulus embedding the ephemeral public-key;
7. Output the recovered RSA key.

Although the idea is nice, the key pairs generated using this algorithm fall short in terms of indistinguishability. In fact it is easy to tell backdoored certificates apart from genuine RSA certificate using only black-box access. Does anybody see why this is the case?

The attack embeds a public-key into an RSA modulus. Elliptic curve public-keys are points on the curve. And elliptic-curve points are easily distinguished from uniform random strings. Hence, a security evaluator could check if the coordinates encoded using the candidate 32-byte substrings of the modulus satisfy the elliptic curve equation.
Can we repair the backdoor? How?

If we could make the elliptic curve points indistinguishable from random strings, then the backdoor indistinguishability would be retained. Designed by Daniel J. Bernstein et al. with the goal to make anti-censorship protocols undetectable, Elligator provides an encoding for points on a single curve as strings indistinguishable from uniform random strings.

All cyber security technology is inherently dual use. This long-held and well-sustained belief [DG] is corroborated by Elligator. Undetectability of curve points, just like any and all cyber security tools, can be used for good or ill; for censorship-circumvention or for surveillance.

I believe we can positively contribute to the discussion and practice of information security by walking the fine line between offense and defense. In this spirit, I have taken the time to write the Python bindings [PyE] for an implementation of Elligator that supports Curve25519 [YA] and rewrote the elliptic-curve asymmetric backdoor in RSA key generation to make sure the backdoored certificates are indistinguishable from genuine ones. [REB]

The key-generation proceeds as follows:

1. Embed a Curve25519 public-key into the key-generator;
2. Generate an ephemeral Curve25519 key at random and the associated uniform representative string;
3. Compute a shared secret using Elliptic Curve Diffie-Hellmann;
4. Use the shared secret to seed a cryptographically secure pseudo-random number generator (CSPRNG) based on AES run in CTR mode;
5. Generate a normal RSA key using the seeded CSPRNG;
6. Replace 32-bytes of the generated modulus with the representative string associated to the ephemeral Curve25519 public-key;
7. Use the original prime factors to compute two new primes leading to a new modulus embedding the uniform representative string,;
8. Output the RSA key.

To recover the target private key the attacker:

1. Extracts the representative string from the target modulus;
2. Maps the representative string to the candidate ephemeral Curve25519 public-key;
3. Computes the shared secret via Elliptic Curve Diffie-Hellmann and using the private-key associated to the public-key embedded in the key-generator;
4. Uses the shared secret to seed the cryptographically secure pseudo-random number generator (CSPRNG) based on AES run in CTR mode,;
5. Generates a normal RSA key using the seeded CSPRNG;
6. Replaces 32-bytes of the generated modulus with the representative string found in the target modulus,;
7. Uses the original prime factors to compute two new primes leading to the target modulus embedding the uniform representative string;
8. Output the recovered RSA key.

Maybe we have time for a demo.

Before wrapping up, I have one last remark to make.

Today we are in the progressive Amsterdam. Therefore I am reminded of Vincent van Gogh, who once said:

"Though I am often in the depths of misery, there is still calmness, pure harmony and music inside me."

If you, like me, believe that insecurity, or the presence of unmitigatable surprises, is the misery of our times. And, if at the same time, you, like me, are also sanguine about our ability to attain better security tradeoffs and give our children the chance to reach confidence in the processes to which they will entrust their business, then we can paraphrase Vincent van Gogh and say that:

"Though we are often in the depths of insecurity, there is still calmness, pure harmony and music inside us."

Thank You.

References:

[BULLRUN] COMPUTER NETWORK OPERATIONS. SIGINT ENABLING.
https://www.eff.org/files/2014/04/09/
20130905-guard-sigint_enabling.pdf

[DG] Geer Jr. D. E., (2014); Cybersecurity as Realpolitik.
http://geer.tinho.net/geer.blackhat.6viii14.txt

[DG2] Geer Jr. D. E., (2014); Security of Things, Cambridge
http://geer.tinho.net/geer.secot.7v14.txt

[EFF] 20150117-Spiegel-Supply-chain Interdiction.
Stealthy Techniques Can Crack Some of SIGINT's Hardest Targets
https://www.eff.org/document/
20150117-spiegel-supply-chain-interdiction-stealthy-techniques-can-crack-some-sigints

[HPKP] Evans, C. et al (2015); Public Key Pinning Extension for HTTP,
RFC 7469, IETF

[JBT] Taylor, J. B. (2008); My stroke of insight, TED Talk,
http://www.ted.com/talks/jill_bolte_taylor_s_powerful_stroke_of_insight

[LS] LangSec: A Workshop on Language Theoretic Security --
Call for Papers, http://spw15.langsec.org/cfp.pdf

[MCS] Langley A. (2015); Maintaining digital certificate security
http://googleonlinesecurity.blogspot.it/2015/03/
maintaining-digital-certificate-security.html

[MS] Microsoft, Cross Certification,
https://msdn.microsoft.com/en-us/library/windows/desktop/
bb540800(v=vs.85).aspx

[NST] Renzo Carbonara, network-simple-tls, Haskell library for simple
network sockets usage patterns using TLS security,
https://github.com/k0001/network-simple-tls

[OSX] List of available trusted root certificates in OS X Yosemite
https://support.apple.com/en-us/HT202858

[PG] Gutmann, P. (2002); PKI: It's Not Dead, Just Resting.
Computer (35,8), IEEE, 0018-9162

https://www.cs.auckland.ac.nz/~pgut001/pubs/notdead.pdf

[PyE] De Gregorio, A. (2015); PyElligator
https://github.com/secYOUre/pyelligator

[RC] Castellucci, R. (2015); rsabd.py
https://gist.github.com/ryancdotorg/18235723e926be0afbdd

[REB] De Gregorio, A. (2015). RSA Elligator Backoor.
https://github.com/secYOUre/rsaelligatorbd

[RJA] Anderson, R. J. (2015); Meeting Snowden in Princeton.
https://www.lightbluetouchpaper.org/2015/05/02/meeting-snowden-in-princeton/

[SF] Bonneau J. and Eckersley P. and Hoffman-Andrews J. (2015); Lenovo Is Breaking HTTPS Security on its Recent Laptops
https://www.eff.org/deeplinks/2015/02/further-evidence-lenovo-breaking-https-security-its-laptops

[UCC] Underhanded Crypto Contest. https://underhandedcrypto.com/

[YA] Yawning Angel. libelligator.
https://github.com/Yawning/libelligator/

[YY] Adam L. Young and Moti M. Yung, 'An Elliptic Curve Asymmetric Backdoor in OpenSSL RSA Key Generation', Advances in Cryptovirology, to apper, http://www.cryptovirology.com/

[YY05] Adam L. Young and Moti M. Yung, 'A Space Efficient Backdoor in RSA and its Applications'. In Bart Preneel and Stafford E. Tavares, editors, Selected Areas in Cryptography -- '05, pp 128-143, Spring, 2005, Lecture Notes in Computer Science No. 3897.

[YY96] Adam L. Young and Moti M. Yung, 'The Dark Side of Black-Box Cryptography, or: Should we trust Capstone?'. In Advances in Cryptology---Crypto '96, N. Koblitz (Ed.), LNCS 1109, pp. 89-103, 1996.