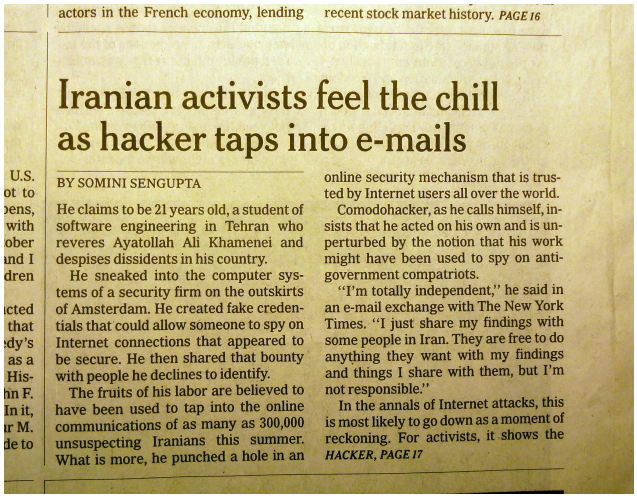
October 2011
Geoff Huston

Hacking away at the Internet's Security

The front page story of the September 13 2011 issue of the *International Herald Tribune* said it all: "Iranian activists feel the chill as hacker taps into e-mails." The news story relates how a hacker has "sneaked into the computer systems of a security firm on the outskirts of Amsterdam" and then "created credentials that could allow someone to spy on Internet connections that appeared to be secure." According to this news report this incident punched a hole in an online security mechanism that is trusted by hundreds of millions of Internet users all over the network.



actors in the French economy, lending recent stock market history. PAGE 16

Iranian activists feel the chill as hacker taps into e-mails

BY SOMINI SENGUPTA

He claims to be 21 years old, a student of software engineering in Tehran who reveres Ayatollah Ali Khamenei and despises dissidents in his country.

He sneaked into the computer systems of a security firm on the outskirts of Amsterdam. He created fake credentials that could allow someone to spy on Internet connections that appeared to be secure. He then shared that bounty with people he declines to identify.

The fruits of his labor are believed to have been used to tap into the online communications of as many as 300,000 unsuspecting Iranians this summer. What is more, he punched a hole in an online security mechanism that is trusted by Internet users all over the world.

Comodohacker, as he calls himself, insists that he acted on his own and is unperturbed by the notion that his work might have been used to spy on anti-government compatriots.

"I'm totally independent," he said in an e-mail exchange with The New York Times. "I just share my findings with some people in Iran. They are free to do anything they want with my findings and things I share with them, but I'm not responsible."

In the annals of Internet attacks, this is most likely to go down as a moment of reckoning. For activists, it shows the

HACKER, PAGE 17

International Herald Tribune – Page 1, September 13 2011

Other news stories took this hyperbole about digital crime and tapping into email conversations on the Internet to new heights, such as The Guardian's report on the 5th of September, that claimed that the "DigiNotar SSL certificate hack amounts to cyberwar, says expert." (http://www.guardian.co.uk/technology/2011/sep/05/diginotar-certificate-hack-cyberwar)

If the Internet's security is so vulnerable to attack, then this incident surely calls into question the basic mechanisms of trust and security upon which the entire global Internet has been constructed. By implication it also calls into question the trustworthiness of services operated by the major global Internet brands such as Google and Facebook, as much as it raises doubts about the levels of vulnerability for the use of online services such as banking and commercial transactions.

Just how serious is this?

Is this the end of civilization as we know it? Well, hardly!

Is digital cryptography now broken? Has someone finally managed to devise a computationally viable algorithm to perform prime factorization of massively large

numbers, which lies at the heart of much of the cryptography used in the Internet today? I really don't think so! [1]

Is this a systematic failure of security? Do we need to re-think the entire framework of cryptography and security in the Internet? Not this time!

As far as I can tell, there has been no dramatic failure in the integrity of the digital technology used for security in the Internet today. Yes, some folk were caught out by this, including the Netherlands government which uses certificates issued by a compromised certification authority, DigiNotar (http://www.diginotar.com), as part of its online service infrastructure. But the hacking incident was not based on a successful direct attack on the technology of cryptography per se, and there is no reason to suppose that the strength of today's encryption algorithms is any weaker today than they were yesterday.

But in observing that the basic technology tools of the Internet's security framework are still operating within acceptable bounds of integrity, and observing that this hacking attack did not drive a gaping hole in our amour of digital cryptography, what cannot be claimed is that the use of these cryptographic tools in today's Internet service environment is similarly trustworthy. The hacking attempt apparently was successful in so far as it provided the capability for third parties to impersonate trusted services and thereby capture users' private data, and evidently some folk did indeed do precisely that, and that's not good at all.

Accordingly, it's worth looking a little more closely at this hacking episode and look at the way in which security is applied to the world of web browsing and the manner in which the vulnerabilities in this security framework were evidently exploited in this hacking episode.

Securing a Connection

When I point my browser at my online banking service, or at any other secure web site for that matter, a part of my browser's navigation bar probably glows a reassuring green, and when I click on it I get the message that I am connected to a web site run by the Acme Banking corporation, and that my connection to this web site has been encrypted to prevent eavesdropping. However, the web site's certificate was issued by some company that I've never met, and never even heard of! When I ask for more information I am told the domain name, the company to whom the certificate for this domain name was issued, the identity of the certificate issuer, and the public key value. I'm also reassuringly informed that the message I am viewing was encrypted before being transmitted over the Internet, and that this encryption makes it very difficult for unauthorized people to view information travelling between computers, and it is therefore very unlikely that anyone could read this page as it passes through the network. All very reassuring, and for the most part as true as we understand the strength of cryptographic algorithms in use today. The connection is using a Transport-Layer Security (TLS) connection, and the traffic is encrypted using a session key that should be impenetrable to all potential eavesdroppers.

But that's not the entire truth, unfortunately.

It may well be that your conversation is secure against eavesdropping, but it's only as secure as the ability of the other party to keep their private key a secret. If the other

[1] At the very least, if someone has managed to achieve this, then they are staying very quiet about it!

side of the conversation were to openly broadcast the value of their private key then the entire encryption exercise is somewhat useless. So obviously my local bank will go to great lengths to keep their private key value a secret, and I rely on their efforts in order to protect my conversations with them.

But even then it's not quite the full story.

Am I really talking to my bank? Or in more general terms, am I really talking to the party whom I thought I wanted to talk with?

The critical weakness in this entire framework of security is that the binding of certificates and keys to DNS names is not an intrinsic part of the DNS itself. It's not an extension of Secure DNS (DNSSEC). It's been implemented an add-on module where third parties generate certificates that attest that someone has a particular domain name. Oddly enough, these certification authorities may never have actually issued that particular domain name, as they are often disconnected with the DNS name registration business. Their business is a separate business activity where, after you have paid your money to a domain name registrar and secured your domain name, you then head to a domain name certification authority and pay them money (commonly they charge more money than the name registration itself) and receive a domain name certificate.

Certification Authorities

Who gets to be a certification authority? Who gets to say who has which domain name and what keys should be associated with that domain name?

Oddly enough the answer is, at a first level of approximation, just about anyone who wants to! I could issue a certificate to state that you have the domain name www.example.com and that your public key value is some number. The certificate I issue that that effect would be little different to the certificates issued by everyone else. Yes, my name is listed as the certificate's issuer, but that's about it in terms of difference between this certificate and set of certificates you already trust via your browser.

So what is stopping everyone from being a Certification Authority? What is preventing this system descending into a chaotic environment with thousands of certificate issuers?

This is where the browser software folk (and other application developers of secure services) step in. In practice it requires a lot of effort, capability, diligence and needless to say, some money, to convince a browser to add your Certification Authority's public key to its list of trusted Certificate Authorities. You have to convince the browser folk that you are consistently diligent in ensuring that you issue certificates to the "correct" holders of domain names and that you undertake certificate management practices to the specified level of integrity and trust. In other words you have to demonstrate that you are trustworthy and perform your role with consistent integrity at all times. You then get listed with all the other trusted certification authorities in the browser, and users will implicitly trust the certificates you issue as part of the internet's security framework.

How many folk are trusted Certification Authorities? How many entities have managed to convince browser manufacturers that they are eminently trustable folk? If you are thinking that this is a special role and should be undertaken by only a very select and

suitably small number of folk who merit such absolute levels of trust for the global Internet, maybe two or three such folk, then, sadly, you are very much mistaken.

Look in your browser in the preferences area for your list of trusted Certification Authorities, and keep your finger near the scroll button, as you will have to scroll through a large number of such entities! My browser has some 80 such entities, including, one government ("Japanese Government"), a PC manufacturer ("Dell Inc"), numerous telcos, and a few dedicated certificate issuers, including DigiNotar.

Do I know all these folk that I am meant to trust? Of course not! Can I tell if any of these organizations are issuing rogue certificates, deliberately, or far more likely, inadverently? Of course not!

The structural weakness in this system is that a client does not know which Certification Authority, or even which duly delegated subordinate entity of a Certification Authority was used to issue the "genuine" DNS certificate. When a client receives a certificate as part of the TLS initialization process, then as long as any one of the listed trusted Certification Authorities are able to validate the presented certificate, even if it's the "wrong" Certification Authority, then the client will proceed with the session with the assumption that the session is being set up with the genuine destination. In other words the entire certification setup is only as strong, or as weak, as the weakest of the certification authorities! It really does not matter to the system as a whole if any single Certification Authority is 'better' at its task than the others, as every single certified domain name is only protected to the extent that the 'weakest' or most vulnerable trusted Certification Authority is capable of resisting malicious attack and subversion of its function! (Indeed, one could argue that there is scant motivation for any trusted Certification Authority to spend significantly more money to be "better" than the others, given that their clients are still as vulnerable as all the other clients of all the other Certification Authorities. In other words there is no overt motivation for market differentiation based on functional excellence!) So all certificates are only as strong as the weakest of the Certification Authorities. And therein lies the seed of this particular hacking episode.

The Hack

The hack itself appears these days to be just another instance of an online breakin to a webserver. The webserver in question was evidently running the service platform for DigiNotar, and the hacker was able to mint some 344 fraudulent certificates, where the subject of the certificate was valid, but the public key was created by the hacker. A full report of the hacking incident was published by Fox-IT (http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2011/09/05/diginotar-public-report-version-1/rapport-fox-it-operation-black-tulip-v1-0.pdf)

To use these fraudulent certificates in an attack requires a little more than minting fraudulent certificates. It requires traffic to be redirected to a rogue web site that impersonates the web page that is under attack. This requires collusion with a service provider to redirect client traffic to the rogue site, or a second attack, this time on the Internet's routing system, in order to perform the traffic redirection.

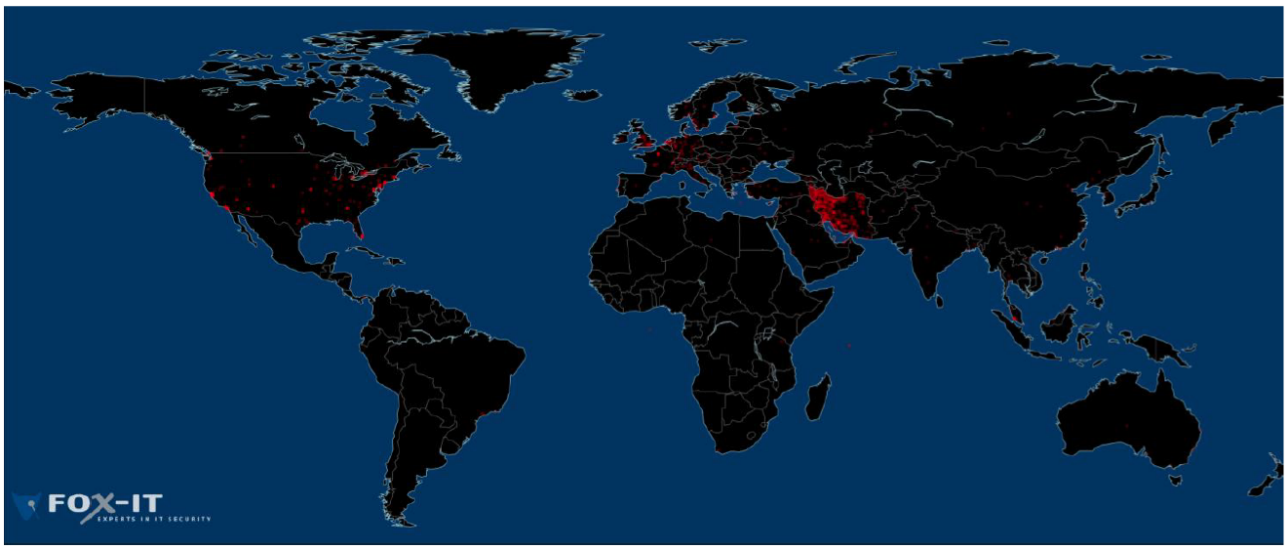
So minting the fraudulent certificates is just one part of the attack. Were these fake certificates used to lure victims to fake web sites and eavesdrop on conversations between web servers and their clients? Lets look at the client's validation process to see if this question can be answered.

When starting a TLS session, the server presents the client with a certificate that contains the server's public key. The client is expected to validate this certificate against its locally held set of public keys that are associated with trusted certification authorities. Here's the first vulnerability. The client is looking for any locally cached trusted key to validate this certificate. It is not looking as to whether a particular public key validates this certificate. Lets say that I have a valid certificate issued by the Trusted Certification Authority Inc., for my domain name, www.example.com. Lets also say that the server belonging to another Certification authority, ACME Inc, is compromised, and a fake certificate is minted. If a user is misdirected to a fake instance of www.example .com, and the bad server passes the client this fake certificate, the client will accept this certificate. The client will accept this fake certificate as valid because the client has no a priori knowledge that the only key that should validate a certificate for www.example .com belongs to the Trusted Certification Authority Inc. When the key belonging to Acme Inc validates this certificate and ACME is a trusted entity according to my browser, then that's good enough to proceed.

Actually that's not the full story. What if I wanted to cancel a certificate? How do certificates get removed from the system and how do clients know to discard them as invalid. A diligent client (and one who may need to check a box in their Browser's preference pane to include this functionality) uses a second test for validity of a presented certificate, namely the Online Certificate Status Protocol (OCSP). This is a protocol used by clients to see if an issued certificate has been subsequently revoked. So once the certificate has been validated against the locally held public key, a diligent application will then establish a secure connection to the certification authority's OCSP server and query the certificate's status.

So this allows fraudulent certificates to be promptly removed from circulation. It assumes of course that clients use OCSP diligently and that the certification authority's OCSP server has not also been compromised in an attack, but in an imperfect world this is at least another measure of relative defence.

The OCSP server logs can also provide an indication of whether the fraudulent certificates have been used by malware servers, because if the certificate was presented to the client, and the client passed it to an OCSP serer for validation, then there is a record of use of the certificate. The Fox-IT report contained an interesting graphic, reproduced below, which shows the geolocation of the source addresses of clients who passed a bad *.google.com certificate to OCSP for validation. The hotspot in red has a strong correlation to Iran.



*Geolocation of OCSP requests for the rogue *.google.com certificate*
Reproduced from DigiNotar Certificate Authority breach "Operation Black Tulip" report, Fox-IT, 5 September 2011

Obviously this attack requires some considerable sophistication and capability, hence the suspicion that the attack may have had some form of state or quasi-state sponsorship, and hence the headlines from the Guardian, quoted at the start of this article, that described this attack as an incident of cyberwarfare of one form or another.

Plugging the Hole?

This is not the first such incident that has driven a hole in the security framework of the Internet, and it's my confident guess that it won't be the last.

It's also a reasonable guess that the evolution of the sophistication and capability that lie behind these attacks point to a level of resourcing that leads some to the view that various state-sponsored entities may be getting involved in these activities in one way or another.

So can we fix this?

It seems to me that the critical weakness that was exploited here was the level of disconnection between domain name registration and certificate issuance. The holders of the domain names were unaware that fraudulent certificates had been minted and were being presented to users as if they were the real thing. And the users had no additional way of checking the certificate's validity by referring back to information contained in the domain name system that was placed there by the domain name holder. The end user is unable to refine the search for a trusted certification authority that will validate the presented certificate from all locally cached trusted certification authorities to the one certification authority that was actually used by the domain name holder to certify their public key value. So is it possible to communicate this additional information to the user in a reliable and robust manner?

The last few years has seen the effort to secure the Domain Name system, DNSSEC, gather momentum. The root of the DNS is now DNSSEC-signed, and attention is now being focused on extending the interlocking signature chains downwards through the DNS hierarchy. The objective is a domain name framework where the end client can validate that the results returned from a DNS query contains authentic information that was entered into the DNS by the delegated authority for that particular DNS zone.

What if we were able to place certificates, or references to certificates into the DNS and protect them via DNSSEC? This is the area under study by the "DNS-based Authentication of Named Entities (DANE) Working Group of the IETF (http://datatracker.ietf.org/wg/dane/). There are a number of scenarios being considered in the working group at present, but the one of interest here does not replace the framework of Certification Authorities and domain name certificates, but it adds another phase of verification of the presented certificate .

The "Use Cases" document from the DANE working group illustrates the proposed approach (http://datatracker.ietf.org/doc/draft-ietf-dane-use-cases/?include_text=1)

I'll quote a few paragraphs from this document. The first paragraph describes the form of attack that was perpetrated in June and July this year on the DigiNotar CA. Its not clear to me if the text predates this particular attack or not, but they are closely aligned in time:

> Today, an attacker can successfully authenticate as a given application service domain if he can obtain a "mis-issued" certificate from one of the widely-used CAs -- a certificate containing the victim application service's domain name and a public key whose corresponding private key is held by the attacker. If the attacker can additionally insert himself as a man in the middle between an client and server (e.g., through DNS cache poisoning of an A or AAAA record), then the attacker can convince the client that a server of the attacker's choice legitimately represents the victim's application service.

So how can DNSSEC help here?

> With the advent of DNSSEC [RFC4033], it is now possible for DNS name resolution to provide its information securely, in the sense that clients can verify that DNS information was provided by the domain holder and not tampered with in transit. The goal of technologies for DNS-based Authentication of Named Entities (DANE) is to use the DNS and DNSSEC to provide additional information about the cryptographic credentials associated with a domain, so that clients can use this information to increase the level of assurance they receive from the TLS handshake process. This document describes a set of use cases that capture specific goals for using the DNS in this way, and a set of requirements that the ultimate DANE mechanism should satisfy.
>
> Finally, it should be noted that although this document will frequently use HTTPS as an example application service, DANE is intended to apply equally to all applications that make use of TLS to connect to application services named by domain names.
>
> "Use Cases and Requirements for DNS-based Authentication of Named Entities (DANE)", R. Barnes, BBN, work in progress: draft-ietf-dane-use-cases-05.txt

Does DANE represent a comprehensive solution to this security vulnerability?

I'd hesitate to be that definitive. As usual with many aspects of security the objective of the defender is to expend a smaller amount of effort in order to force an attack to spend a far larger amount of effort. From this perspective the DANE approach appears to offer significant promise, as it interlocks a number of security measures and forces a potential attacker to compromise a number of independent systems simultaneously. Within the DANE framework the attacker cannot attack any certification authority, but must compromise a particular certification authority, and the attacker must also attack DNSSEC and compromise the information contained in signed DNS responses for that domain in order to reproduce the effects of the attack described here. This seems to fit the requirement of a small amount of additional defensive effort by the server and the client, creating a significantly larger challenge to the attacker.

But there are a number of preconditions here for this approach to be effective.

- DNSSEC needs to be ubiquitously deployed and maintained
- Issued DNS certificates need to published in the secure DNS zone using the DANE framework

- Client DNS resolvers need to be not only DNSSEC-aware, but enforce DNSSEC outcomes

- Applications, including browsers, need to validate the certificate that is being used to form the TLS connection against the information provided by a validated DNS response for the DANE credentials for that DNS zone.

It's probably not perfect, but it is a large step forward along a path of providing more effective security in the Internet.

Unfortunately, all of this is not an instant solution ready for widespread use today, or even tomorrow. It is possible that we could see this in widespread use in a couple of years time, but, sadly, it is more likely that this particular activity of securing the DNS and its use in the Internet will not receive adequate levels of attention and associated financial resourcing in the coming years, and the overall activity of ubiquitous adoption of DNSSEC and the use of this by a DANE framework for certificates in the DNS may well take upward of 5 years before we see any significant levels of use. Until then there is the somewhat worrisome prospect of little change in the framework of the Internet's security from that used today, and the equally concerning prospect that this particular hacking event will not be the last.

Acknowledgement

I am indebted to Olaf Kolkman of NLnet Labs for a stimulating conversation about this attack and the implications for securing the Internet. NLnet Labs is one of a small number of innovative and highly productive research groups that has developed considerable levels of expertise in this area of security and the DNS. (http://nlnetlabs.nl/)

Postscript

Once you lose that essential element of trust, your continued existence as a trusted Certification Authority is evidently a very limited one. On Tuesday 20th September 2011 the Dutch company DigiNotar was officially declared bankrupt in a Haarlem court.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.

www.potaroo.net