

SecCon **2017**



10
YEARS

Cybersecurity Innovation and the
Digital Economy



Cisco Confidential 2017 All Rights Reserved

Johns Hopkins and the Cryptographic Knowledge Base

Debra Baker, CISSP CCSP

>whoami

- Debra Baker, CISSP CCSP
- 20 years of practical experience in security starting in USAF – IBM - Entrust - Cisco
- Cisco Champion of JHU Crypto Knowledge Base
- Editor of CYS Report and Podcaster: <https://cysreport.com>
- Debra's Blog: <https://debinfosec.com>
- Distinguished SME in Information Security (ISC2)
- Twitter: @deb_infosec

Cryptography and the Internet



Factorization Bug Exposes Millions Of Crypto Keys To 'ROCA' Exploit



'All wifi networks' are vulnerable to hacking, security expert discovers

WPA2 protocol used by vast majority of wifi connections has been broken by Belgian researchers, highlighting potential for internet traffic to be exposed

SHA-1 hashes recovered for 320M breached passwords

Millions of embedded devices use the same hard-coded SSH and TLS private keys

The keys were hard-coded by manufacturers and can be used by attackers to launch man-in-the-middle attacks

Entropy drought hits Raspberry Pi harvests, weakens SSH security

Cryptographic Key Reuse Remains Widespread In Embedded Products

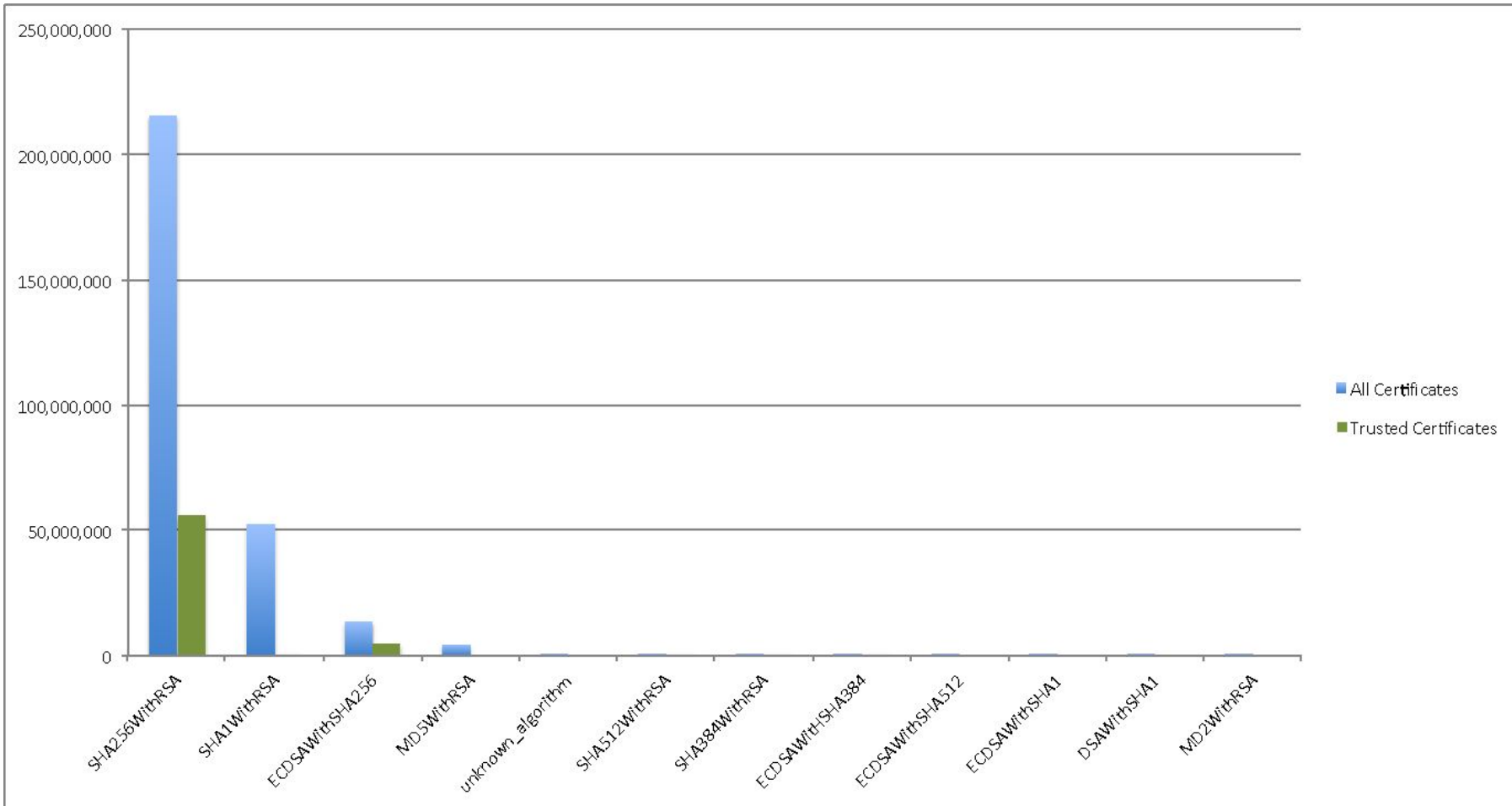
'Worrying' 9 Per Cent Of Encrypted Web Vulnerable To Private Key Attacks

Digital key reuse leaves millions of network devices vulnerable



Censys is a search engine that allows computer scientists to ask questions about the devices and networks that compose the Internet. Driven by Internet-wide scanning, Censys lets researchers find specific hosts and create aggregate reports on how devices, websites, and certificates are configured and deployed. [\[more information\]](#)

Certificates on the Internet



Certificates on the Internet



Certification Authorities	Trusted Certificates
Let's Encrypt:	40,868,332
COMODO CA Limited:	7,768,715
cPanel, Inc:	5,448,400
Symantec Corporation:	3,332,312
GeoTrust Inc:	1,977,954
GoDaddy.com, Inc:	1,819,771
GlobalSign nv-sa:	1,357,564
Western Digital Technologies:	824,914
DigiCert Inc:	503,847
Thawte, Inc:	483,361
StartCom Ltd.:	435,647
TrustAsia Technologies, Inc.:	300,467
Amazon:	264,489
WoSign CA Limited	183,553
Entrust, Inc:	148,104
Starfield Technologies, Inc:	146,947
Internet2:	87,281
TERENA:	84,249
CloudFlare, Inc:	66,916
ActalisS.pA/03358520967:	64,939



Source:theregister.co.uk

Security

Google to kill Symantec certs in Chrome 66, due in early 2018

This is how trust ends, not with a bang but with a whimper

Source:Threatpost

- A SHA-1 collision occurs when two distinct pieces of data hash to the same message digest.
- If an attacker can craft a collision, they could use it to create two different files that share the same SHA-1 hash value.

An update on SHA-1 certificates in Chrome

December 18, 2015

Posted by Lucas Garron, Chrome security and David Benjamin, Chrome networking

As [announced last September](#) and supported by [further recent research](#), Google Chrome does not treat SHA-1 certificates as secure anymore, and will completely stop supporting them over the next year. Chrome will discontinue support in two steps: first, blocking new SHA-1 certificates; and second, blocking all SHA-1 certificates.

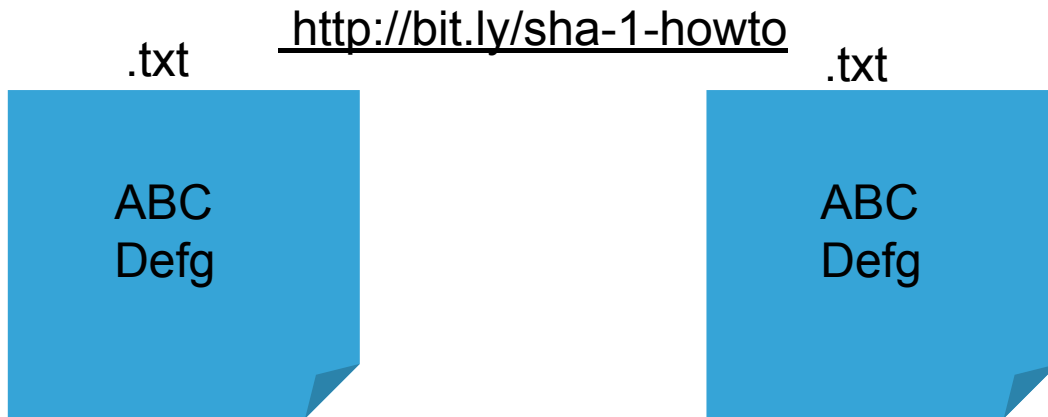
Step 1: Blocking new SHA-1 certificates

Starting in early 2016 with Chrome version 48, Chrome will display a certificate error if it encounters a site with a leaf certificate that:

1. is signed with a SHA-1-based signature

What is SHA-1?

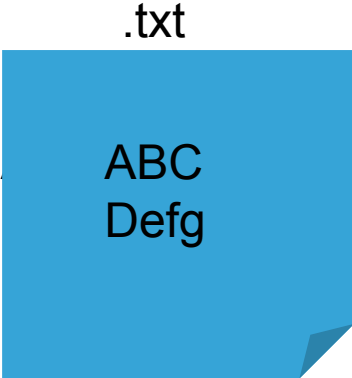
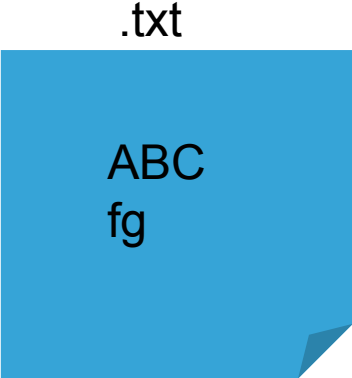
- **SHA-1 (Secure Hash Algorithm 1)** is a cryptographic hash function
- Digital Fingerprint of data (document, image, etc) – provides integrity
- One way hash is generated always same size, SHA-1 is 160 bits



94c41e93c5a19744c5061b2bfee82b85b08547f7 = 94c41e93c5a19744c5061b2bfee82b85b08547f7

What is SHA-1?

Make one change, a different fixed length 160 bit hash is value is generated

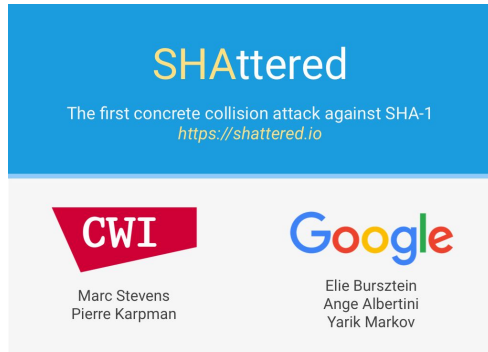


a80ce785e49c3fb30d8a4ffa5b79eace3ffc78e



94c41e93c5a19744c5061b2bfec82b85b08547f7

SHA-1 Attack “SHattered Attack”



SHattered

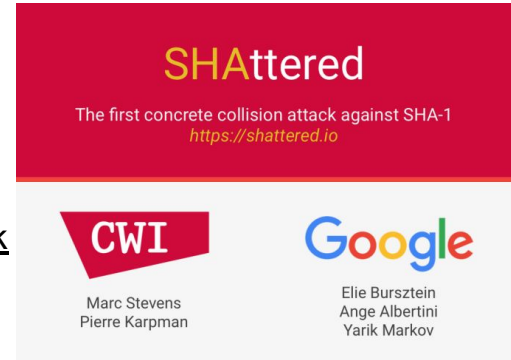
The first concrete collision attack against SHA-1
<https://shattered.io>

CWI
Marc Stevens
Pierre Karpman

Google
Elie Bursztein
Ange Albertini
Yarik Markov

<https://shattered.io>

<http://bit.ly/SHA1-ShatteredAttack>



SHattered

The first concrete collision attack against SHA-1
<https://shattered.io>

CWI
Marc Stevens
Pierre Karpman

Google
Elie Bursztein
Ange Albertini
Yarik Markov

38762cf7f55934b34d179ae6a4c80cadccb7f0a = 38762cf7f55934b34d179ae6a4c80cadccb7f0a

“Based on an **identical-prefix collision attack**: two files have the same predetermined beginning, followed by different inputs and an optional amount of identical data.” - techtarget

SHattered Attack – Does Not...

Allow an attacker to generate a collision with an existing file.

- For example, it's not possible to use this method to generate a malicious executable file which matches the signature of an existing legitimate executable.
- However, it would be possible for an attacker to generate two executable files which have the same SHA-1 hash but perform different actions when they run.

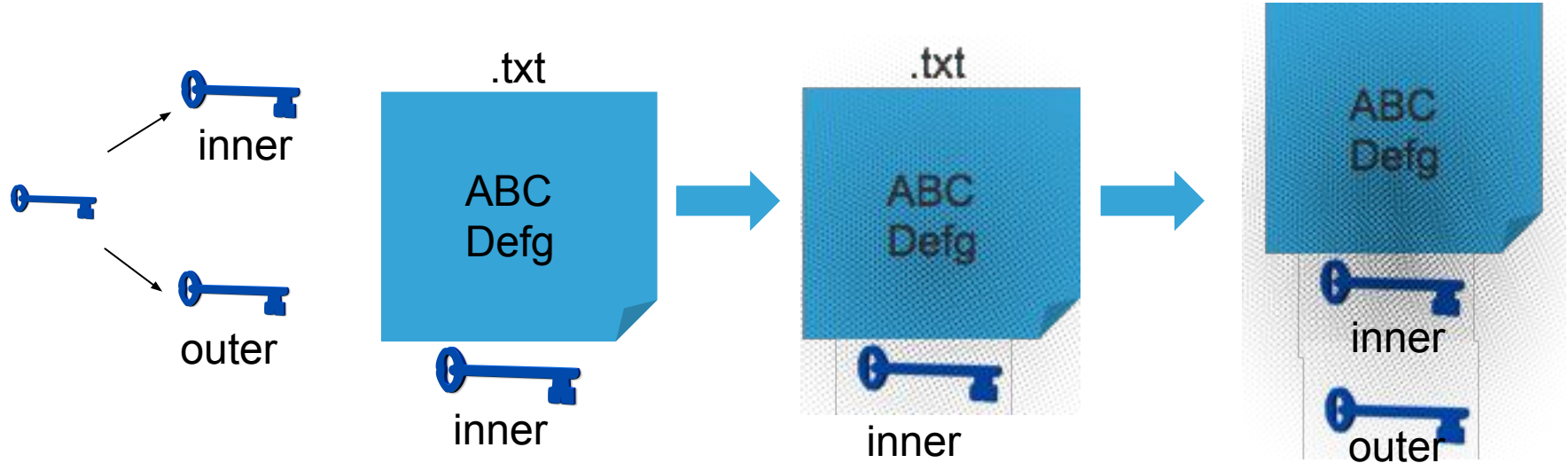
When is it still ok to use SHA-1?

- Checking the hash of an executable or file that we know is trusted (from trusted site)
- As a key derivation function, there are no known flaws in SHA-1, which is how dh-group14-sha1 is being used in SSH and IPSec.
- Similarly, HMAC-SHA1 is believed to be (still) safe as long as SHA1 is pseudorandom.

- Matthew Green

hmac-sha1- Why its safe

- HMAC uses two passes of hash computation.
- The secret key is first used to derive two keys - inner and outer. The first pass of the algorithm produces an internal hash derived from the message and the inner key. The second pass produces the final HMAC code derived from the inner hash result and the outer key.



dh-group14-sha1 - Why its safe

- SSH and IPsec uses both SHA-1 and hmac-sha1 when dh-group14-sha1 is used for key derivation
- TLS only uses hmac-sha1
- In [RFC 4253](#) Section 8 – 2, it says SHA-1 is “H” in the diffie-hellman key exchange used for key derivation:
- **H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K)**
(these elements are encoded according to their types),
and ***signature*** s on H with its private host key. S sends (K_S || f || s) to C. The signing operation may involve a second hashing operation.

dh-group14-sha1 – What is “H” hashing?

- The hash H is computed as the HASH of the concatenation of the following:
 - string V_C, the client's identification string (CR and LF excluded)
 - string V_S, the server's identification string (CR and LF excluded)
 - string I_C, the payload of the client's SSH_MSG_KEXINIT
 - string I_S, the payload of the server's SSH_MSG_KEXINIT
 - string K_S, the host key mpint e, exchange value sent by the client mpint
 - f, exchange value sent by the server mpint
 - K, the shared secret

dh-group14-sha2 & above – RFC (KEX) In Draft

Key Exchange (KEX) Method Updates and Recommendations for Secure Shell

Section 3.6

- group14 is still a reasonable size. [[RFC3526](#)]
- “This key exchange group uses SHA-1 which has security concerns [[RFC6194](#)]. However, this group is still strong enough and is widely deployed. This method is being moved from MUST to SHOULD to aid in transition to stronger SHA-2 based hashes.”
- This method will transition to SHOULD NOT when SHA-2 alternatives are more generally available.

Cryptographic Best Practices



No matter how good the algorithm, bad random numbers =



FAILURE

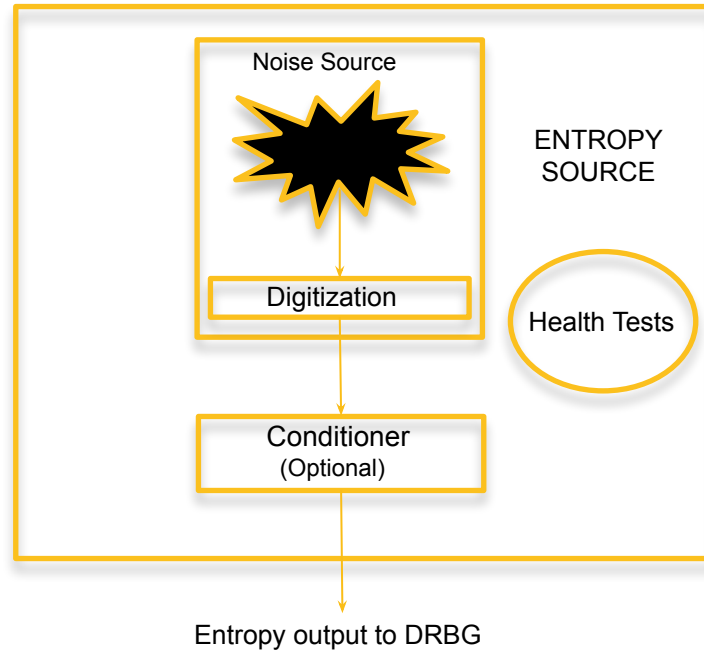
What is entropy?

- Entropy is true randomness
- Such as atmospheric pressure
- Used to seed the DRBG

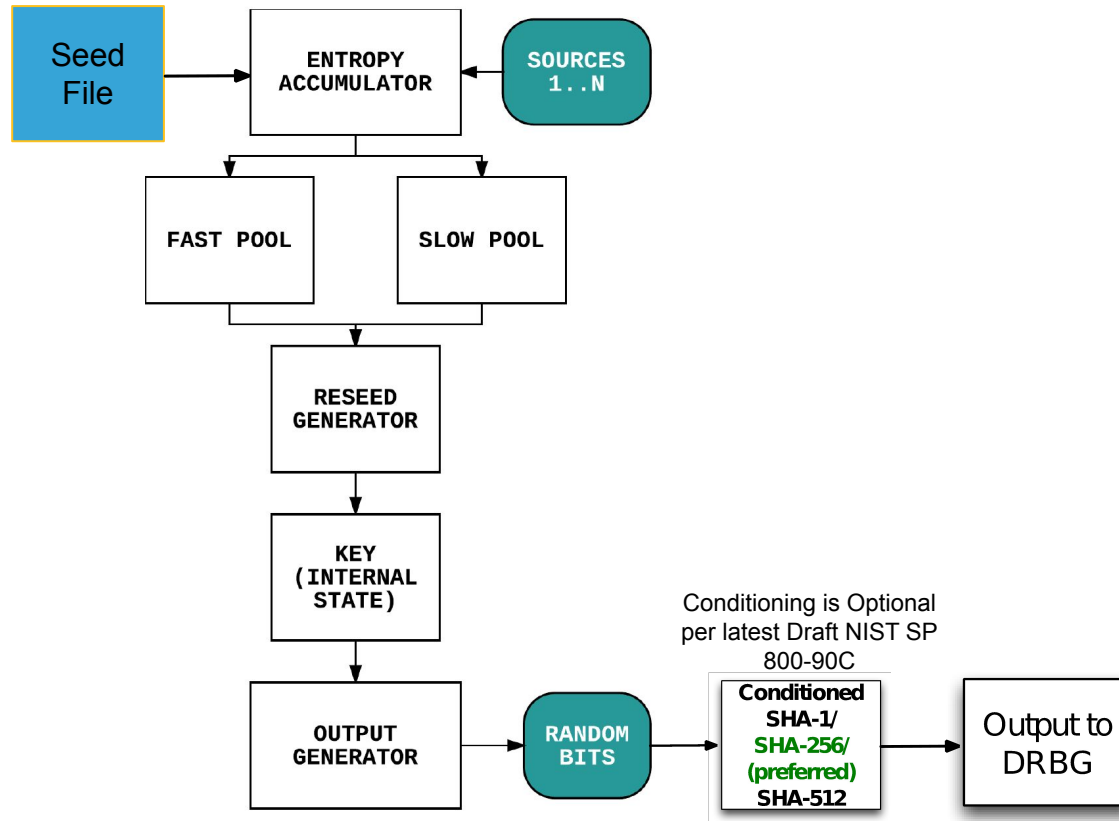
Random numbers and security

- Cryptographic key generation (MACSec, IPsec, SSH, TLS ...)
- Nonces and initialization vectors (802.11i, EAP, MACSec...)
- Padding schemes, digital signatures (DSA, OTPs...)
- Using poor random numbers (random != unique) can have catastrophic consequences. And cause severe embarrassment!

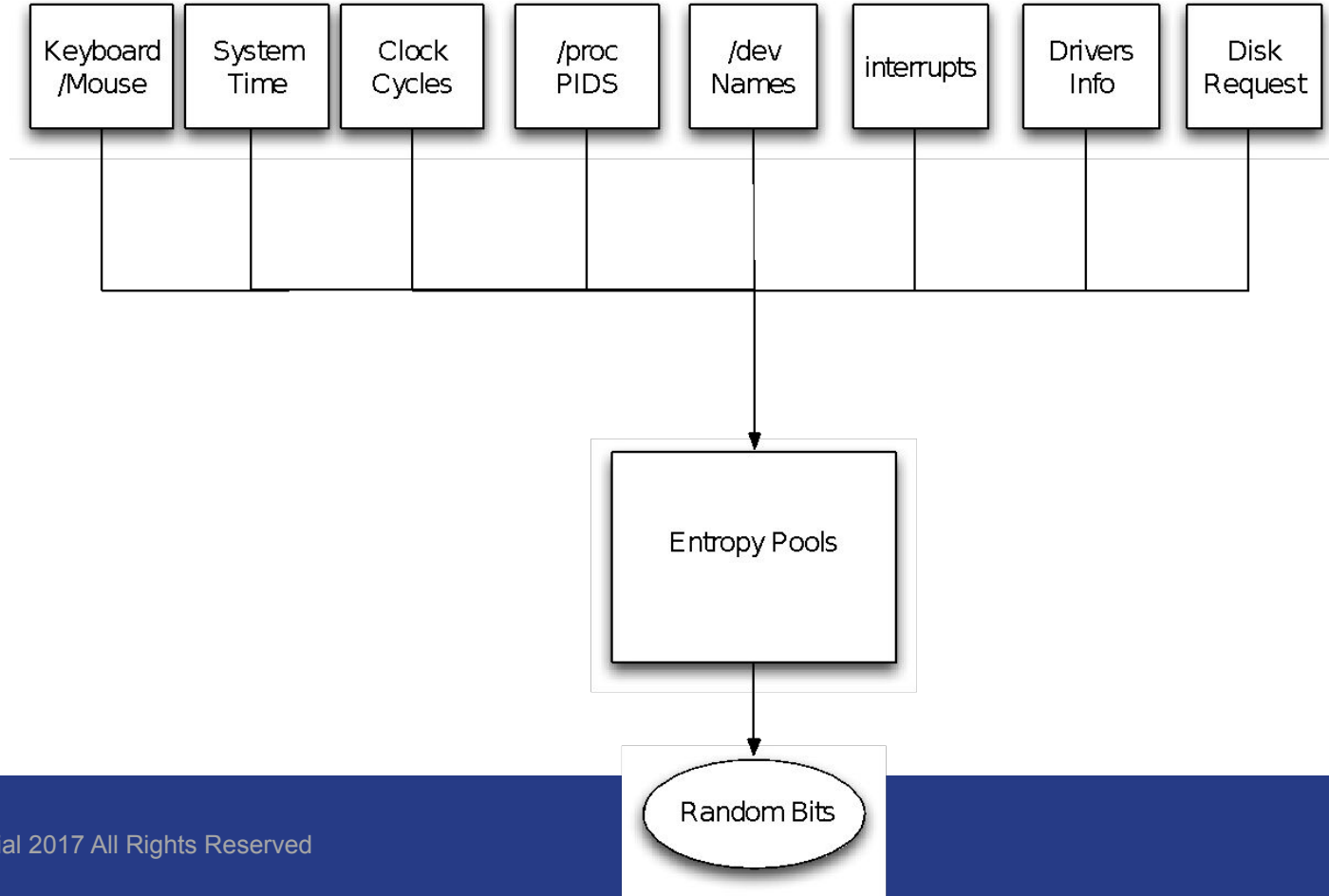
Entropy Overview



/dev/[u]random - Linux

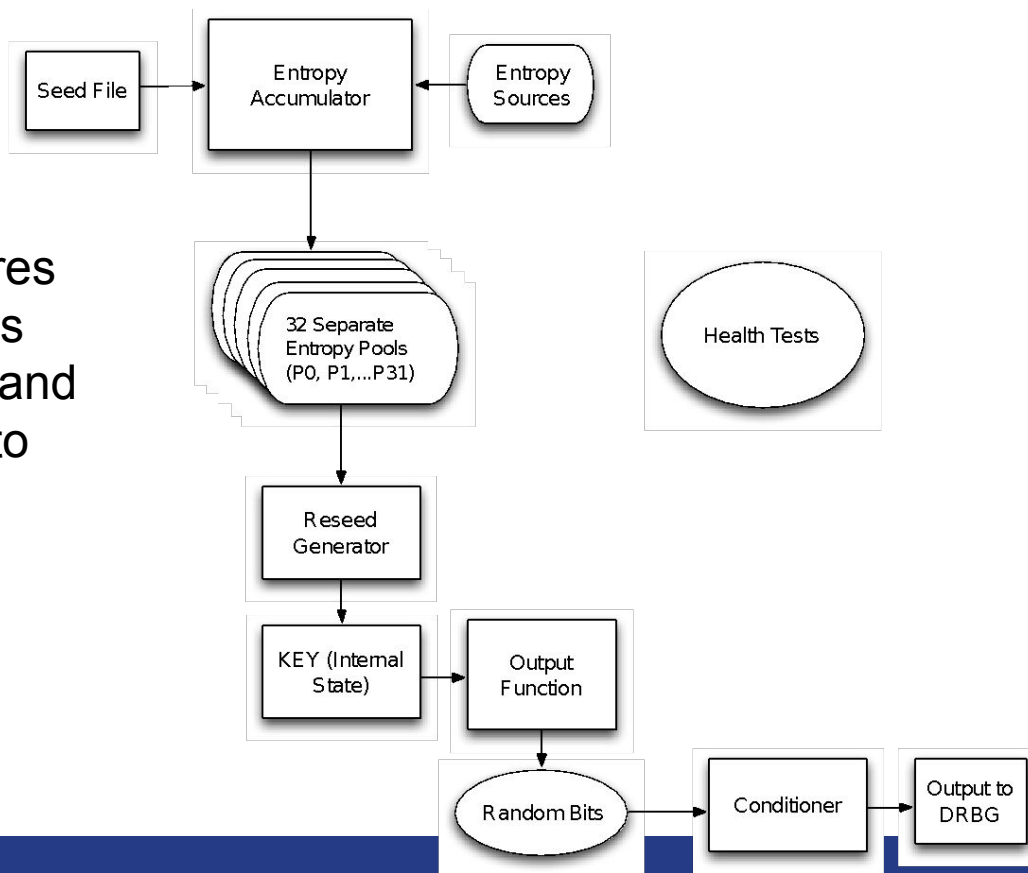


Typical Entropy Sources



Fortuna Entropy Source – FreeBSD 11.1 and above QNX 7.0

Fortuna ensures blocked unless have enough and random data to seed DRBG



/dev/[u]random and Embedded Systems

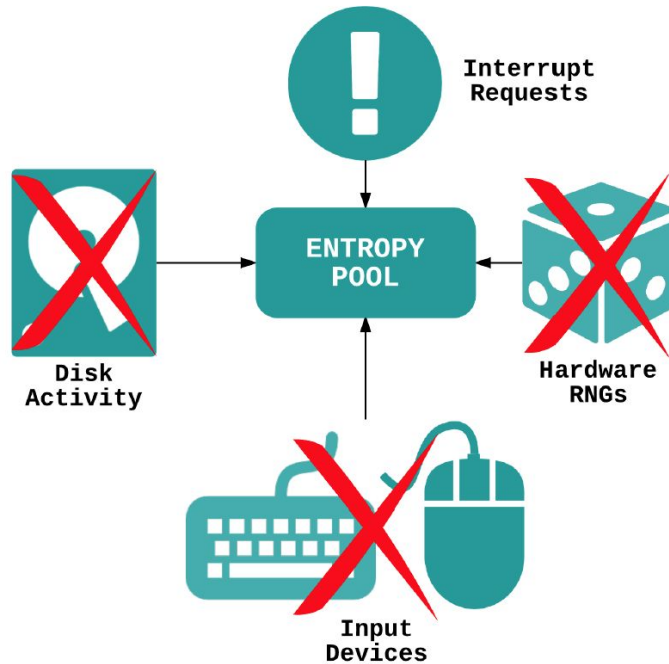


Image courtesy of: <http://samvartaka.github.io>

What to do?

- Use a Hardware RNG
 - Act2Lite, Cavium, on-board entropy chip intel (DRNG)
- Make sure your entropy pool is seeded on boot and reseeded properly
- Test the PRNG using NIST SP800-90B assessment tools
- Software-Only...
 - QNX 7.0 (random.c) – Fortuna
 - FreeBSD 11.1 – Fortuna
 - Truerand.c

In the News...

'All wifi networks' are vulnerable to hacking, security expert discovers

ONSemiExecMtg



WPA2 protocol used by vast majority of wifi connections has been broken by Belgian researchers, highlighting potential for internet traffic to be exposed



WPA2 Krack Attack - What Happened?

- WPA2 has been broken – Belgian Univ Researchers
- “The attack works against all modern protected wifi networks.
- Forces nonce reuse on the client
- Weakness in the design of the **WPA2 protocol** in which the client can be forced into reusing a key

WPA2 Krack Attack - What Does This Mean?

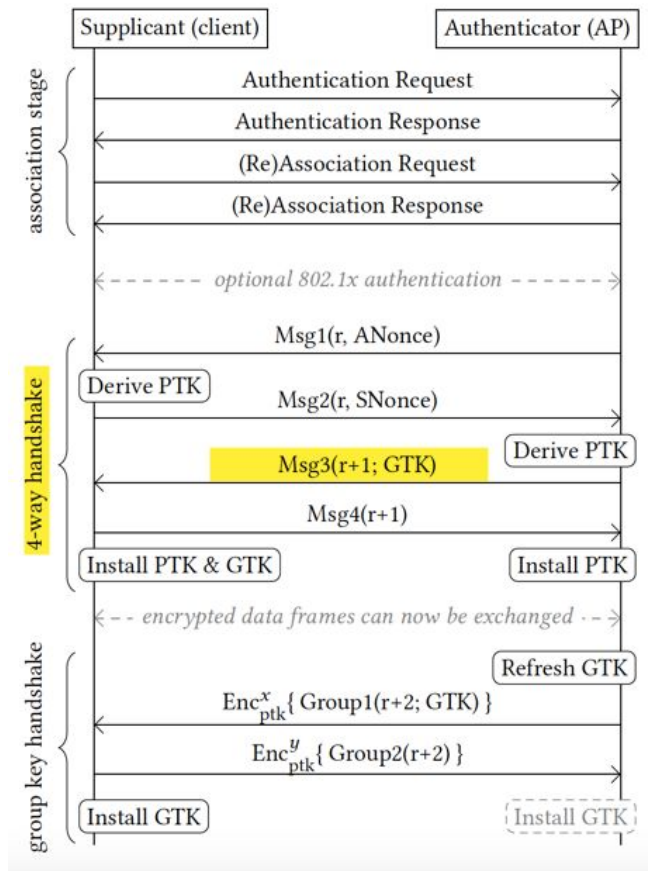
- WPA2 is no longer secure.
- Reveals sensitive information such as credit card numbers, passwords, or usernames, and so on.
- “Depending on the network configuration, it is also possible to inject and manipulate data. For example, an attacker might be able to inject ransomware or other malware into websites.”

WPA2 Krack Attack - Details

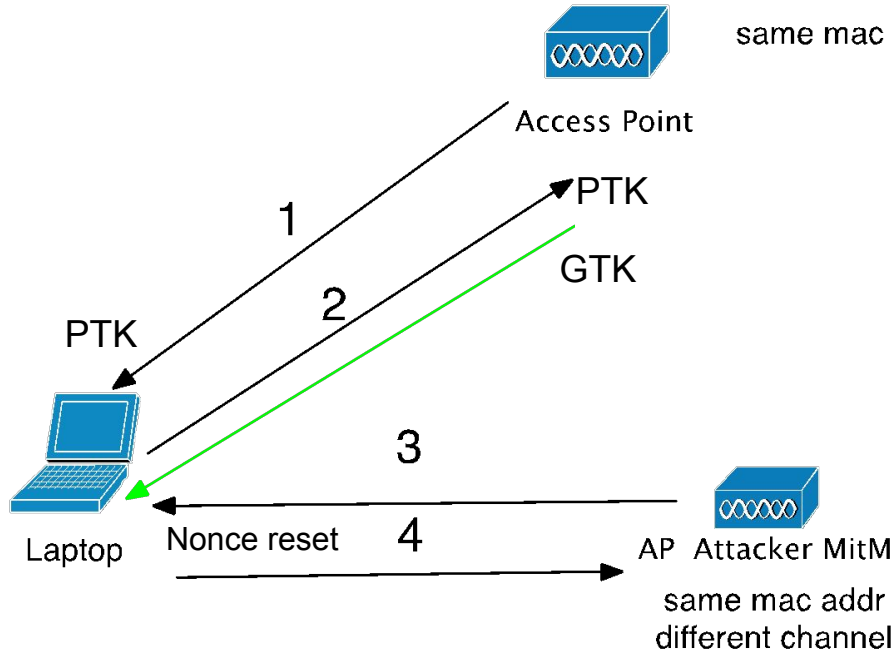
- Devastating against Linux and Android 6.0 or higher.
 - **Android and Linux can be tricked into (re)installing an all-zero encryption key**
- If the client uses either the WPA-TKIP or GCMP encryption protocol, instead of AES-CCMP,
 - **Nonce reuse enables an adversary to not only decrypt, but also to forge and inject packets.**
- Note that the attacks **do not recover**
 - **the password of the Wi-Fi network.**
 - (any parts of) the fresh encryption key that is negotiated during the 4-way handshake.

WPA2 Krack Attack – The Details Please...

PTK = Session Key



Krack Attack 4-way handshake



same mac addr

1. AP (Authenticator) sends ANonce to client (supplicant) and client derives the PTK
2. Client sends SNonce to AP and the AP derives the PTK
3. Access Point (AP) will retransmit message 3 if it did not receive an appropriate response as acknowledgment, client resets it nonce and retransmits so hacker has key (Linux and Android zero out key) (Win & IOS don't accept the retransmission, but are susceptible to the GTK)
4. Malicious AP derives PTK and client installs PTK and GTK

WPA2 Krack Attack – How to Protect Yourself

- Attack is executed against the client
- Note: Only access points that support the Fast BSS Transition handshake (802.11r) can be vulnerable, but all clients are vulnerable...

Remediation:

- Patch all devices phones, tablets, computers, wireless Apps
- What if I don't have a patch for my AP?
 - Disable client functionality on AP (which is used in repeater modes)
- Use a VPN, (not free) recommended: F-Secure Freedom, Cisco Anyconnect
- Use TLS when accessing websites



JHU Crypto Knowledge Base

<http://bit.ly/jhu-sha1>

References

JHU Knowledge Base: <http://bit.ly/jhu-sha1>

TLS:

- <https://testssl.sh>
- <https://www.ssllabs.com/ssltest/>
- <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf>
- <https://feistyduck.com>

For FIPS 140 related information go to:

<http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401vend.htm>

NIST: <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>

NIAP Protection Profiles: <https://www.niap-ccevs.org/Profile/PP.cfm>

Entropy Links:

<http://bit.ly/truerand>

<https://github.com/freebsd/freebsd/blob/master/sys/dev/random/fortuna.c>

NIST SP800-90B tools:

https://github.com/usnistgov/SP800-90B_EntropyAssessment

<https://hardenedbsd.org/>

Krack Detector Tool:

<https://securityaffairs.co/wordpress/65229/hacking/krackdetector.html>

Debra's Contact Info

- Editor of CYS Report and podcast:
<https://cysreport.com>
- Debra's Blog: <https://debinforec.com>
- Twitter: @deb_infosec
- LinkedIn:
<https://www.linkedin.com/in/debrabakernc/>

Random Number Generation

- Use: NIST SP800-90A Rev 1:
 - *HMAC_DRBG (SHA-256, SHA-512) (512 recommended)*
 - *CTR_DRBG (AES),*
 - *AES Hash_DRBG (any)*
- *Avoid: DUAL_EC_DRBG and ANSI X9.31 using AES*
 - *Hardware based RNG is always best ex. ACT-2Lite, Octeon*
 - *Otherwise /dev/urandom with truerand.c pulling random data from the clock variance between the real time clock and CPU clock.*

Asymmetric Cryptography

□ Use:

- RSA 2048, 3072, 4096
 - ✓ RSA 2048 is recommended;
 - ✓ CSFC: RSA 3072 or above;
 - ✓ RSA 4096 Quantum proof
- ECDSA with P-256, P-384, P-521
 - ✓ P-256 is recommended;
 - ✓ CSFC: P-384

□ *Avoid: DSA any key size*

Hashing

□ Use:

- SHA-256, SHA-384, SHA-512
 - ✓ SHA-256 is recommended;
 - ✓ CSFC: SHA-384
- `hmac-sha1`, `hmac-sha2-256`, `hmac-sha2-512`
- `AEAD_AES_128_GCM`, `AEAD_AES_256_GCM`

□ *Avoid: SHA-1, hmac-md5, hmac-sha1-96*

cPPs cryptography – Best Practices (TLSv1.2)

□ Use: **TLsv1.2** (Recommended)

□ **Avoid: TLsv1.1 or lower or SSLv3 or lower**

□ TLS Recommended Ciphers:

~~TLS_RSA_WITH_AES_128_CBC_SHA~~

~~TLS_RSA_WITH_AES_192_CBC_SHA~~

~~TLS_RSA_WITH_AES_256_CBC_SHA~~

~~TLS_DHE_RSA_WITH_AES_128_CBC_SHA~~

~~TLS_DHE_RSA_WITH_AES_192_CBC_SHA~~

~~TLS_DHE_RSA_WITH_AES_256_CBC_SHA~~

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_192_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_192_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

~~TLS_RSA_WITH_AES_128_CBC_SHA256~~

~~TLS_RSA_WITH_AES_192_CBC_SHA256~~

~~TLS_RSA_WITH_AES_256_CBC_SHA256~~

~~TLS_DHE_RSA_WITH_AES_128_CBC_SHA256~~

~~TLS_DHE_RSA_WITH_AES_256_CBC_SHA256~~

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_192_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_192_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS Best Practices

Avoid the following ciphers:

TLS_DHE_RSA_WITH_AES_128_CBC_SHA

TLS_DHE_RSA_WITH_AES_256_CBC_SHA

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

Unless:

dh group 14 (2048 bit) or above for key exchange (kek).

Otherwise:

Key sizes less than dh group 14 susceptible to logjam attack

TLS Best Practices

Avoid the following ciphers:

TLS_RSA_WITH_AES_128_CBC_SHA

TLS_RSA_WITH_AES_192_CBC_SHA

TLS_RSA_WITH_AES_256_CBC_SHA

TLS_RSA_WITH_AES_128_CBC_SHA256

TLS_RSA_WITH_AES_192_CBC_SHA256

TLS_RSA_WITH_AES_256_CBC_SHA256

Why?

- Do not provide perfect forward secrecy
- Same RSA public/private key pair is used to both authenticate and encrypt the Symmetric Key used for encryption

TLS Recommended Ciphers – Best Practices

▣ <https://weakdh.org/sysadmin.html>

- **Network Security Assessment, 3rd Edition (O'Reilly) By: Chris McNab**
- ▣ Use X.509v3 certificates for mutual authentication for server to server communications whenever possible.
- ▣ Use: secp256r1, secp384r1, secp521r1
- ▣ Server and Client must reject any connections offering SSL 1.0, SSL 2.0, SSL 3.0, TLS 1.0
- ▣ TLS Server must support mutual authentication with X.509v3 certificates
- ▣ For most websites, using RSA keys stronger than 2048 bits and ECDSA keys stronger than 256 bits is a waste of CPU resources and might impair user experience. Similarly, increasing the strength of the ephemeral key exchange beyond 256 bits for ECDHE has little benefit.

SSHv2

- For SSHv2 key exchange:
 - Use: `diffie-hellman-group14-sha1` for SSH key exchange
 - *Avoid: `diffie-hellman-group1-sha1(768 bit)`, `ecdh-sha2-nistp256`, `ecdh-sha2-nistp384`, and `ecdh-sha2-nistp521`*
- `diffie-hellman-group2` and below susceptible to the logjam attack
- Ecdh does not provide perfect forward secrecy
- NIST curves may be unsafe per SafeCurves web site <http://safecurves.cr.yp.to> - *Lange and Bernstein*
- All sessions must be rejected if remote client or server is only advertising a non compliant Common Criteria algorithms and key sizes in hello. Disable export algorithms.

SSHv2

- SSH transport data encryption:

- Use: **aes128-ctr**, **aes256-ctr**, **AEAD_AES_128_GCM**, **AEAD_AES_256_GCM**

- *Avoid: aes128-cbc, aes256-cbc*

- SSH authentication public key algorithms:

- Use: **ssh-rsa**, **ecdsa-sha2-nistp256**, **ecdsa-sha2-nistp384**, **ecdsa-sha2-nistp521**, **x509v3-ecdsa-sha2-nistp256**, **x509v3-ecdsa-sha2-nistp384**, **x509v3-ecdsa-sha2-nistp521**

- *Avoid: ssh-dsa*

- SSH transport data integrity:

- Use: **hmac-sha1**, **hmac-sha2-256**, **hmac-sha2-512**, **AEAD_AES_128_GCM**, **AEAD_AES_256_GCM**

- *Avoid: hmac-md5, hmac-sha1-96*

