Crypto Failures
and how to avoid them

Guy Barnhart-Magen
DC27 - AppSec Village 2019

Who am I?

Father of two, hacker

BSidesTLV chairman and CTF Lead

(Lucky to speak at many conferences)

Today: Cyber Security Consultant

Before: Intel, Cisco and a couple of Startups

OS Hardening, Crypto, Embedded Security, Security of ML

@barnhartguy

OWASP AppSec Israel 2018

BSides TLV

t2.fi INFOSEC

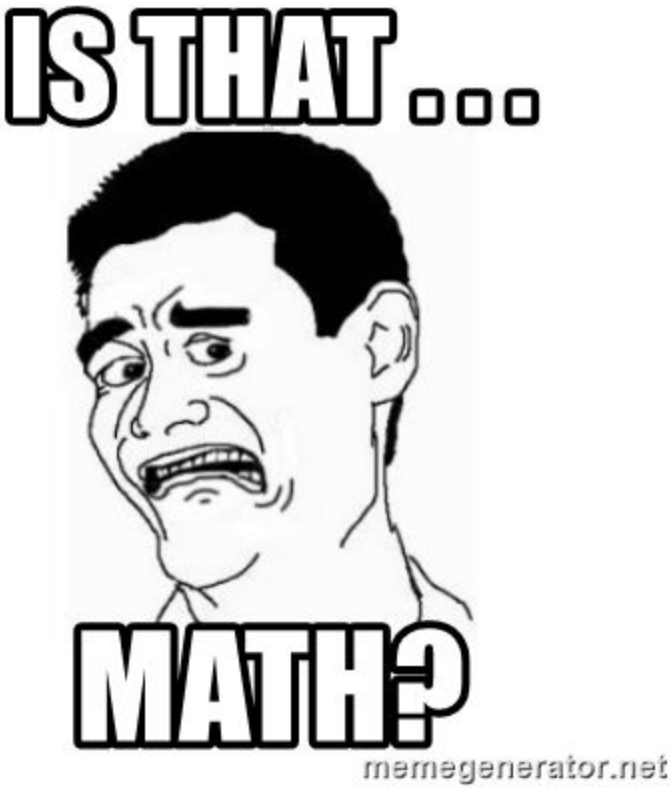
THE SAS CON

BSides LAS VEGAS

CRYPTO + PRIVACY VILLAGE

44CON

LKSAT SKYTA TALKS ATSKY


IS THAT...
MATH?
memegenerator.net

BASICS

Confidentiality	Keeping things secret
Integrity	Can we forge a message?
Authenticity	Can we trust that this is from the sender?

Cryptographic Hash Functions

One way function

1 bit change in input changes ~50% of output

Looks just like noise

```
SHA1("The quick brown fox jumps over the lazy dog")2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
```

```
SHA1("The quick brown fox jumps over the lazy cog")de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
```

2^{80} to bruteforce, 2019 attack: 2^{68}

This is now practical with ~$100K of Amazon EC2

Symmetric Cryptography

Two way function

1 bit change in input changes ~50% of output

Looks just like noise

Very fast

```
msg = aes.encrypt("The quick brown fox jumps over the lazy dog" + "\0"*21)caf5f61978f250e4f34533fcba2ffe541623aa5c6be805c27df12a00659a2d6857138f7eefcf4fa09d301200091cec8fc30614ad5e9ae2e2740f3bcc550468f6
```

```
msg = aes.encrypt("The quick brown fox jumps over the lazy cog" + "\0"*21)272e9f8c4d8b7e52800c9dddfdb96aa22fc0a28f69aa2ef90ce580d34b3bd29c9de87c59859e70e30cd050721bd28787c92cb54b47fc3b475c78d3731832d74d
```

Asymmetric Cryptography

Different operations to encrypt and decrypt

Very Slow

Create a Private and Public key pair (keep your private key secret)

```
PubKey, PrvKey = GenASymmKeys(RSA,4096)
```

Give everyone your public key

Now anyone can create a message, encrypt it with your public key

```
EncMsg = Enc(msg, PubKey)
```

Only you can decrypt it with `PrvKey`

```
msg = Dec(EncMsg, PrvKey)
```

Signatures

Prove your identity by proving you have a secret tied to your public identity

Very Slow

Having a `PrvKey` is equivalent to an identity (holding the key is proof you have the secret)

```
h = hash(msg)
signature = EncMsg(h, PrvKey)
```

Now anyone can verify that you signed `msg`, with your published `PubKey`

```
h = hash(msg)
h` = DecMsg(signature, PubKey)
```

`h == h`` proves you signed `msg`

Random

The chance to predict the next bit is 50%

TRNG - True Random Number Generator

PRNG - Pseudo Random Number Generator

```
random=PRNG(seed=TRNG)
```

What happens when the random isn't random?

We often assume `nonce` values are random

CRYPTO USAGES, BUILDING BLOCKS

Bitcoin

Coin = Finding hash collisions (mining)

Wallets = Identities are tied to private keys

Transactions = Signing coin transfers from wallet to wallet

Blockchain = a decentralized database to store coins (**hash collisions**) and transactions

Wallets/Identities/**private keys** (anonymous), but not confidential - everything is recorded!

Lose your private key, you cannot **sign** transactions, lose access to those coins (frozen money)

Elliptic Curve Cryptography (ECC)

Asymmetric

Similar to RSA

Much faster

Wallet

```
Address = Hash(PubKey)
```

1DY5YvRxSwomrK7nELDZzAidQQ6ktjR

“This money I can spend, can now be spent by X”

BITCOIN TRANSACTION

12ff51d83eac5b69b222af02ffe68454b7d5501b6dcdf2f33fec25c8510d53fd

mined Feb 18, 2018 10:20:47 PM

1GDFhSXaemytQhqYZarabatwpwuZoMNHAG	7.17974665 BTC

33Xis87i9P1ui5to1PoDic2LVYfMUn2Tt2	0.06564117 BTC (U)
15QaHxet56z1ES36xSsyh8mQBE5NY97n2k	7.11144294 BTC (S)

FEE: 0.00266254 BTC

1 CONFIRMATIONS

7.17708411 BTC

Elliptic Curve Cryptography (ECC)

Known: G

h: hash of the msg

d: private key

k: random number

(r,s): signature

To create a transaction we need to sign our message

```
msg = "This money I can spend, can now be spent by X"
h = hash(msg)
k = random()
r = cross_op(k,G) //also random
sig = sign(h,r,k,d)
```

We publish `sig` on the blockchain, once accepted then X has the money

$$(r, s) = \left(r, \frac{h + rXd}{k}\right)$$

$$r = kXG$$

CRYPTO FAILS

What if `k` isn't random?

`k` isn't random

`r` isn't random either

`s` is public on the blockchain

1. Find a couple of transaction that share the same `k`
2. Do some algebra, extract `k`
3. Now use `k` and `s` to extract `d`
4. `d` is the private key
5. …
6. Profit!

$$(r, s) = \left(r, \frac{h + rXd}{k}\right)$$

$$r = kXG$$

THIS CAN'T HAPPEN IN THE REAL WORLD!

Author | Topic: Bad signatures leading to 55.82152538 BTC theft (so far) (Read 33421 times)

BurtW
Legendary
Online

Activity: 1218

I no longer support vanity addresses

Ignore

Bad signatures leading to 55.82152538 BTC theft (so far) #1
August 10, 2013, 10:53:13 PM

I have only seen this discussed in the newbies section so I thought I would open a thread here for a more technical discussion of this issue.

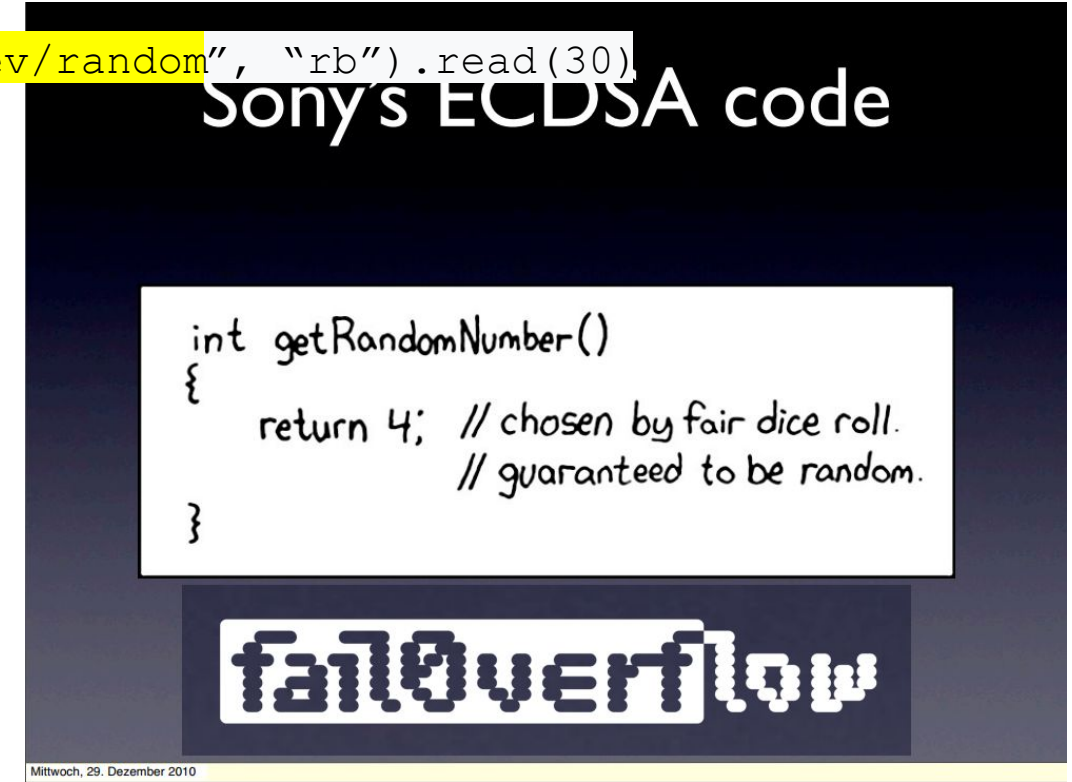
Several people have reported their BTC stolen and sent to https://blockchain.info/address/1HKywxiL4JziqXrzLKhmB6a74ma6kxbSDj

As you can see the address currently contains 55.82152538 stolen coins.

It has been noticed that the coins are all transferred in a few hours after a client improperly signs a transaction by reusing the same random number. As discussed here:

THIS CAN'T HAPPEN IN THE REAL WORLD!

```
m = open("/dev/random", "rb").read(30)
```



Anonymity vs. Confidentiality

Anonymous - no one knows your identity

Confidential - no one knows what you're doing

Bitcoin is not Anonymous, all transactions are recorded on the blockchain

Remember WannaCry?

DEPLOYING BAD CRYPTO

Goto Fail

Chain of trust is established by verifying signatures, all the way to a trusted root

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1,
&hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx,
&hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
```

Goto Fail

Chain of trust is established by verifying signatures, all the way to a trusted root

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1,
&hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx,
&signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx,
&hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
```

Bitlocker

Bitlocker (SW) will use HW if available

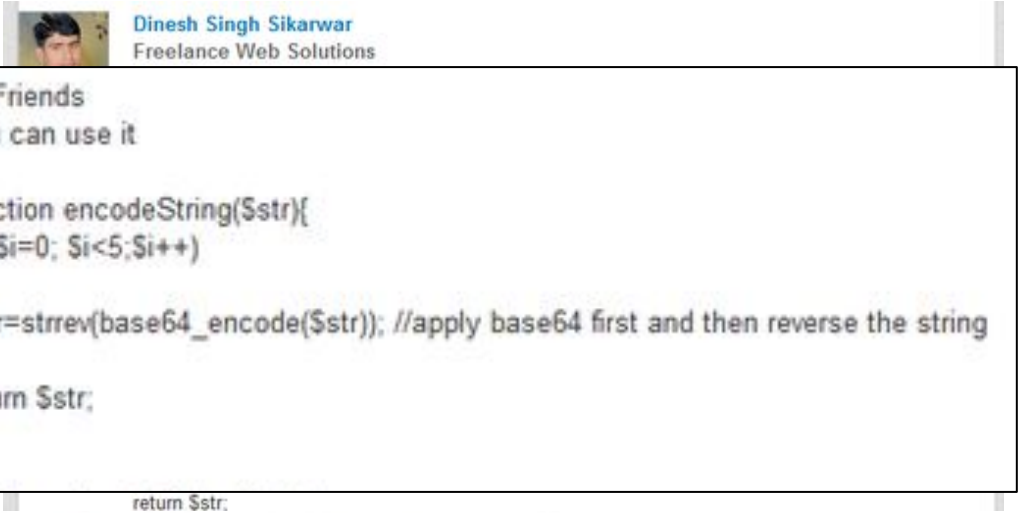
Several self encrypting drives had weak firmware, leading to attacks

- Password and DEK not linked
- Single DEK for entire disk
- non-random DEK
- Wear leveling
- Zero buffer as password

Writing your own crypto

Crypto is hard for experts

Never roll your own crypto!

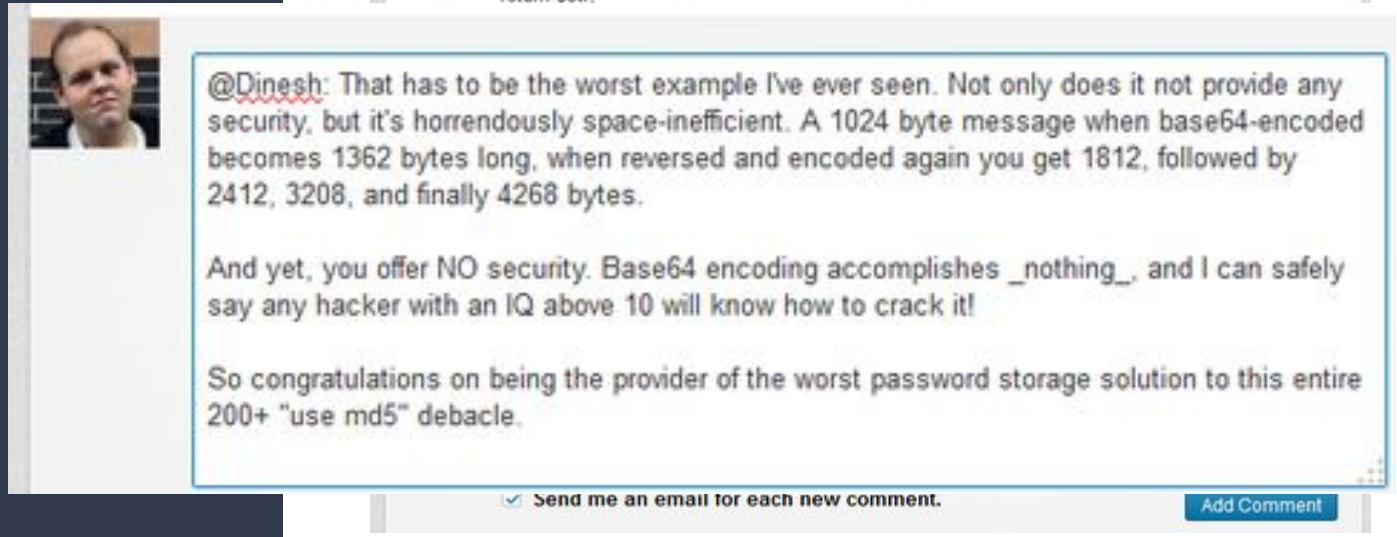


Dinesh Singh Sikarwar
Freelance Web Solutions

Hi Friends
you can use it

```
function encodeString($str){
for($i=0; $i<5;$i++)
{
$str=strrev(base64_encode($str)); //apply base64 first and then reverse the string
}
return $str;
}
```

return $str;



@Dinesh: That has to be the worst example I've ever seen. Not only does it not provide any security, but it's horrendously space-inefficient. A 1024 byte message when base64-encoded becomes 1362 bytes long, when reversed and encoded again you get 1812, followed by 2412, 3208, and finally 4268 bytes.

And yet, you offer NO security. Base64 encoding accomplishes _nothing_, and I can safely say any hacker with an IQ above 10 will know how to crack it!

So congratulations on being the provider of the worst password storage solution to this entire 200+ "use md5" debacle.

Send me an email for each new comment.

Add Comment

CryptHook

Hook Send/Recv system calls

Uses GCM to authenticate

Encrypt with a fixed key

Same key and no sequence numbers leads to:

- Replay attack
- Message re-ordering
- Selectively dropping messages
- No session key - no forward secrecy

IS EVERYTHING ON FIRE?

WHERE TO GET ADVICE?

https://www.reddit.com/r/crypto

https://crypto.stackexchange.com/

Consultants/Experts

Crypto Review Firms

Never roll your own crypto!

LIBRARIES YOU SHOULD LOOK AT

Always vet your own libraries

Don’t trust people that roll their own crypto

Performance is a major differentiator

Which feature do you need? Attack model?

There are more options than OpenSSL

https://en.wikipedia.org/wiki/Comparison_of_cryptography_libraries

https://github.com/sobolevn/awesome-cryptography#frameworks-and-libs

BoringSSL

Golang/crypto

NaCL

NSS

WolfCrypt

MatrixSSL

WHAT CAN YOU DO?

Leave implementation of crypto algorithms to experts, update when possible

Take classes, courses, training and workshops - become an expert yourself

Read a lot, have a relevant degree (Math, EE, CS) - follow up on papers and conferences

Review your code by experts, save a lot of headache down the road

Outsource to others, where possible

THANK YOU!

Questions?

@barnhartguy