
TALES
FROM THE
CRYPT0

INTRO

Graeme Neilson

Security Researcher / Consultant

- Reverse Engineering
- Cryptography
- Network Infrastructure

graeme@aurasoftwaresecurity.co.nz

Andy Prow

Managing Director

- Security Training
- Social Engineering

andy@aurasoftwaresecurity.co.nz

Aura Software Security, New Zealand

SYNOPSIS

aura SOFTWARE SECURITY

- Uses
- Principles
- Hashes
- Symmetric Ciphers
- Public Key Crypto
- Crypto Challenge
- Conclusions

USES

aura SOFTWARE SECURITY

- Integrity - Hash
- Confidentiality -–Symmetric Encryption
- Authentication -–Hash, Public Key Crypto
- Non-Repudiation - Public Key Crypto
- Key Exchange - Public Key Crypto

Kerckhoffs' Principle

aura SOFTWARE SECURITY

Auguste Kerckhoffs, 1883

The security of a system should reside only in the key

Disco Principle

aura SOFTWARE SECURITY

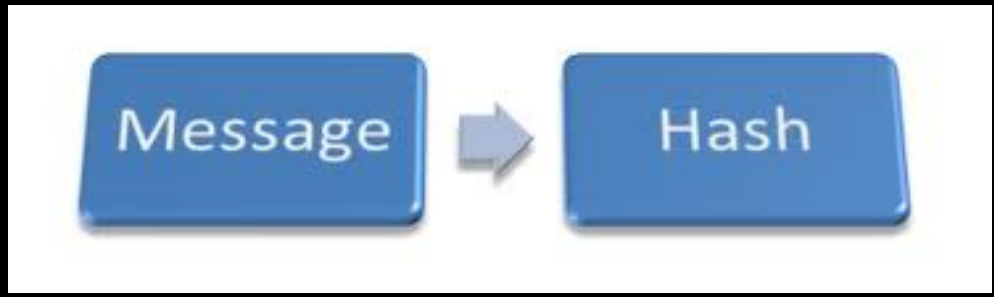
Don't Invent Super Crypto of your Own

HASHES

aura SOFTWARE SECURITY

- One way functions - non reversible - fixed size output
- Easy to compute for any message
- Infeasible to find a message that has a specific hash
- Infeasible to modify a message without changing the hash
- Infeasible to find different messages with the same hash

Algorithms:
MD2, MD4, MD5, SHA



PASSWORD PROBLEM

aura SOFTWARE SECURITY

Passwords stored in clear text in the database

```
public void ChangePassword(User user, string newPassword)
{
    user.Password = newPassword;
    user.Save();
}
```

```
public bool CheckPassword(User user, string password)
{
    return user.Password == password;
}
```

STORE HASH

aura SOFTWARE SECURITY

- Store a hash of the password
- Compare hashes

```
public void ChangePassword(User user, string newPassword)
{
    string hash = HashPassword(newPassword);
    user.PasswordHash = hash;
    user.Save();
}
```

```
public bool CheckPassword(User user, string password)
{
    string hash = HashPassword(password);
    return user.PasswordHash == hash;
}
```

MAKE HASH

aura SOFTWARE SECURITY

```
public string HashPassword(string input)
{
    UTF8Encoding encoder = new UTF8Encoding();
    SHA256Managed algorithm = new SHA256Managed();
    byte[] hashedDataBytes = algorithm.ComputeHash(encoder.GetBytes(input));
    return byteArrayToString(hashedDataBytes);
}
```

- All common platforms contain crypto libraries
- Use this library code
- It is simple code
- Just slect your algorithm

RAINBOW TABLES

aura SOFTWARE SECURITY

- Precompute hashes for a set of passwords
- Set of passwords defined by max length & character set
- Time versus memory trade off - less CPU more Storage

MS LANMAN

aura SOFTWARE SECURITY

Algorithm:

- Password is converted to uppercase
- Null-padded to 112 bits
- Split into two 56 bit values
- Each 56 bit value has null bits inserted every seven bits to create a 64 bit key”
- The constant string **KGS!@#$%**”is DES encrypted with each of the keys
- The two ciphertext values are concatenated to form the 128 bit LM hash

Weaknesses:

- Limited character set
- Passwords > 7 chars split in two and hashed separately
- No salt

RANDOM SALTS

aura SOFTWARE SECURITY

- Solution is store a salt with the hash of the password
- Append salt to password before hashing

```
public void ChangePassword(User user, string newPassword)
{
    user.Salt = GenerateRandomSalt();
    string hash = HashPassword(user.Salt, newPassword);
    user.PasswordHash = hash;
    user.Save();
}
```

```
public bool CheckPassword(User user, string password)
{
    string hash = HashPassword(user.Salt, password);
    return user.PasswordHash == hash;
}
```

MAKE SALTED HASH

aura SOFTWARE SECURITY

- Use a cryptographicaly secure pseudorandom number generator!
- *System.random* is NOT random!

```
public byte[] GenerateRandomSalt()
{
    byte[] saltBytes = new byte[saltSize];
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
    rng.GetBytes(saltBytes);
    return saltBytes;
}
```

```
public string HashPassword(byte[] salt, string input)
{
    UTF8Encoding encoder = new UTF8Encoding();
    SHA256Managed algorithm = new SHA256Managed();
    byte[] saltedInput = JoinArrays(salt, encoder.GetBytes(input));
    byte[] hashedDataBytes = algorithm.ComputeHash(saltedInput);
    return byteArrayToString(hashedDataBytes);
}
```

JUNIPER NETSCREEN

aura SOFTWARE SECURITY

Password Hashing Algorithm:

- MD5 hash of username + '*:Administration Tools:*'"+ password
- Base64 encode the hash
- Insert the characters 'n' 'r' 'c' 's' 't' 'n'""

Examples:

nJ8aK7rVOo1Ico6CbsQFKNCtviAjTr
nPZmEerYEtdHcanJhsHGsSBtkrAV4
nKqqMDroCJPBc8lF2smLmCMtnNC
nNtMGWrpGPFJcNuMTsJKyPEtPhH
nKfNBWrbFpzNcaZAJs6M18HteGPU
nGH8EvrtD3/Dc4JDrsZEzyMtiFKLtn

Weaknesses:

- It's MD5!
- Salt is username and constant string - NOT random!

MD5 or SHA?

aura SOFTWARE SECURITY

- MD5 is not collision resistant
- Different files with the same hash can be created
- Different certificates with the same hash can be created
- MD5 is 128 bit and is less resistant to brute force (GPU)

- Use SHA-2 family

  SHA-256, SHA-512

SYMMETRIC CIPHERS

aura SOFTWARE SECURITY

The same key is used for encryption and decryption

- Lots of stream and block ciphers to choose from: SERPENT, TWOFISH, DES, 3DES, IDEA, RC4, RC5, RC6, AES, TWOFISH, BLOWFISH...
- Rijndael (aka AES) won the NIST Advanced Encryption Standard competition to replace the Data Encryption Standard (DES)

FORMS AUTH

aura SOFTWARE SECURITY

```
FormsAuthenticationTicket authTicket = new
    FormsAuthenticationTicket(1,                    // version
                              txtUserName.Text,     // user name
                              DateTime.Now,         // creation
                              DateTime.Now.AddMinutes(60), // Expiration
                              false,                // Persistent
                              roles                 // User Data
                              );

string encryptedTicket = FormsAuthentication.Encrypt(authTicket);

HttpCookie authCookie = new HttpCookie(FormsAuthentication.FormsCookieName,
                        encryptedTicket);
```


```
<machineKey
  validationKey="AutoGenerate | value[, IsolateApps]"
  decryptionKey="AutoGenerate | value[, IsolateApps]"
  validation="[SHA1 | MD5 | 3DES]"
  decryption="[Auto | AES | 3DES]"
  />
```

Keep the machine keys secure

MAKE KEY

aura SOFTWARE SECURITY

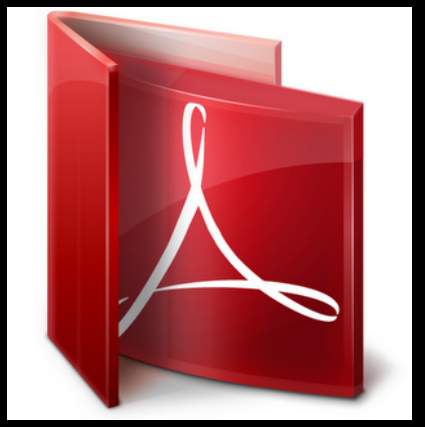
- Use Cipher Block Chaining (CBC) not Electronic Code Book (ECB)
- Initialisation Vector (IV) must be random and not reused
- Java & .Net create random IVs for you

```
// Create Key
KeyGenerator kg = KeyGenerator.getInstance(“AES");
SecretKey secKey = kg.generateKey();

// Create Cipher
Cipher aes = Cipher.getInstance(“AES/CBC/PKCS5Padding");
aes.init(Cipher.ENCRYPT_MODE, secKey);

// Create stream
FileOutputStream fos = new FileOutputStream(aesFile);
BufferedOutputStream bos = new BufferedOutputStream(fos);
CipherOutputStream cos = new CipherOutputStream(bos, aes);
ObjectOutputStream oos = new ObjectOutputStream(cos);
```

ADOBE ACROBAT

aura SOFTWARE SECURITY

- Acrobat 2.0 - 6.0 RC4 / MD5 40 bit encryption
- Acrobat 7.0 - 8.0 AES 128 bit encryption
- Acrobat 9.0 - AES 256 & 128 bit encryption
- Adobe 9 made the encryption function more efficient
- Much faster to brute force ACROBAT AES 256 than ACROBAT AES 128



PUBLIC KEY CRYPTO

aura SOFTWARE SECURITY

Public Key for encryption and a Private Key for decryption

The key pairs are mathematically related such that using the key pair together achieves the same result as using a symmetric key twice.

- Relies on mathematical operations that require 'little work' but whose inverse operations take 'lots of work'
- Testing if a number is prime or multiplying two prime numbers takes little work
- Prime factoring a large integer takes a LOT of work.

MAKE KEY PAIR

aura SOFTWARE SECURITY

- Random Random RANDOM!

```
//Generate a key pair

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA”, "SUN");

SecureRandom random = SecureRandom.getInstance("SHA1PRNG”, "SUN");

keyGen.initialize(1024, random);

KeyPair pair = keyGen.generateKeyPair();
PrivateKey priv = pair.getPrivate();
PublicKey pub = pair.getPublic();

//Create a Signature object, initialize it with the private key

Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");

dsa.initSign(priv);
```

DEBIAN V OPENSSL

aura SOFTWARE SECURITY

- All keys generated on Debian systems Sep 2006 - May 2008.
- To fix unitialised variable Debian patched OpenSSL.
- The seed for the random number generator became the curent PID (1 to 32768)
- For each (algorithm & key size) only 32767 key values AND:

- Keys generated at boot time < 500 value
- User generated keys probably 500-10,000
- Most keys probably 1-3000 value

SONY v FAILOVER

aura SOFTWARE SECURITY

- Only signed executables should be run
- Elliptic Curve Digital Signature Algorithm used to make keys
- The required random number is always the same.
- Given two signatures we can calculate the private key - oops!

CRYPTO CHALLENGE

aura SOFTWARE SECURITY

OTHER ATTACKS

aura SOFTWARE SECURITY

- Cut and paste code from the Interwebs
- Brute force passphrase / password for SSH private key
- Brute force weak password so no need to crack hash
- Compromise CA to create fradulent certificates (COMODO)
- Malware sniffs VPN keys from memory
- Malware modifies crypto algorithm in memory to weaken keys

RSA SECUREID

aura SOFTWARE SECURITY

OPINION:

- RSA network and SecureID source code compromised
- Security of SecureID resides not in the code but in the random seeds for the tokens and server
- Attackers may have gained seeds or know how to generate
- seeds for clients of SecureID
- Security is now only the password / PIN

FACT:
We have some clients that are going to be reissued SecureID tokens..

CONCLUSIONS

aura SOFTWARE SECURITY

The security of a system should reside only in the key

- Do not do DISCO
- Protect your keys
- Hash with **SHA**
- Encrypt with **AES**
- Randomness is key
- Developers should be trained

RESOURCES

aura SOFTWARE SECURITY

- Practical Cryptography, Bruce Schneier
http://www.schneier.com/book-practical.html

- Applied Cryptography, Bruce Schneier
http://www.schneier.com/book-applied.html

- Debian OpenSSL Tools
http://digitaloffense.net/tools/debian-openssl/

- Console Hacking 2010, fail0verflow, 27th Chaos Communication Congress

- Dynamic Cryptographic Trapdoors, Eric Filiol,
ESIEA Laval CVO Lab & French DoD, CanSecWest 2011

aura SOFTWARE SECURITY

QUESTIONS?