Programando aplicações criptográficas com OpenSSL

ENIGMA

Criptografia

Criptografia

- Técnicas para ocultar o conteúdo de uma mensagem, de forma que só o recepiente consiga decifrar
- Permite transmitir mensagens sigilosas em meios inseguros (Internet)
- Criptografia simétrica: a mesma chave é usada para cifrar e decifrar

Cifra de César

- Rotaciona o alfabeto um número fixo de passos (ex. 2)
  - a -> C
  - b -> D
  - c -> E
  - ...
  - y -> A
  - z -> B

Cifra de César

- Com duas rotações
  - abacate -> CDCECVG
  - segredo -> UGITGFQ
- Com vinte rotações
  - abacate -> UVUWUNY
  - segredo -> MYALYXI

AES

- Atual padrão de criptografia simétrica
- Baseada em operações lógicas OR
- Se implementado corretamente, uma mensagem cifrada com AES-256 levará o tempo de toda a idade do universo para conseguir quebrar a mensagem com força bruta

Aplicações criptográficas (ou que usam criptografia):

Aplicações criptográficas (ou que usam criptografia):

- Navegadores (Firefox, Tor Browser, Chrome)
- Mensageiros (Telegram, WhatsApp)
- Criptografia de email (OpenPGP)
- Criptografia de disco (LUKS, VeraCrypt)

Aplicações criptográficas (ou que usam criptografia):

- Navegadores (Firefox, Tor Browser, Chrome)
- Mensageiros (Telegram, WhatsApp)
- Criptografia de email (OpenPGP)
- Criptografia de disco (LUKS, VeraCrypt)

No geral, programas que se comunicam ou que armazenam dados

Quero escrever meu próprio programa para..

- Me comunicar seguramente
- Navegar anonimamente
- Proteger meus arquivos

Quero escrever meu próprio programa para..

- Me comunicar seguramente
- Navegar anonimamente
- Proteger meus arquivos

Tem certeza?

Quero escrever meu próprio programa para..

- Desenvolvimento seguro é uma habilidade muito difícil e que exige muito estudo
- Lembre-se: não exite nada 100% seguro nem livre de bugs
  - Tor, GPG, Linux, Android, Signal, LUKS, ...
- Analise seu cenário de risco: provavelmente é melhor usar algo com uma diversidade de desenvolvedores e vários bugs resolvidos

Quero escrever meu próprio programa para..

- Mas vai fundo: aprender é sempre incrível!
- Você pode aprender para poder contribuir com projetos software livre/open source
- Quem sabe um dia começar um projeto seu

Algoritmos criptográficos

- Geralmente não é uma boa ideia escrever seu próprio algoritmo
- Algoritmos seguros
  - “ Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can't break. ”
  - A segurança de um algoritmo não está na sua obscuridade.

Bibliotecas criptográficas

- Da mesma forma que escrever aplicações seguras é difícil, bibliotecas criptográficas tambem são
- Processadores recentes disponibilizam instruções para criptografia (AES-NI)
- O kernel te prove uma API para usá-las

Bibliotecas criptográficas

- Algoritmo seguro vs implementação segura
- Operações criptográficas precisam ser muito eficientes
- Gerenciamento de memória preciso: apagar dados sensiveis da memória

Bibliotecas criptográficas

- Algoritmo seguro vs implementação segura
- Operações criptográficas precisam ser muito eficientes
- Gerenciamento de memória preciso: apagar dados sensiveis da memória
- Linguagem C cabe nesses dois requisitos

Bibliotecas criptográficas

- Algoritmo seguro vs implementação segura
- Operações criptográficas precisam ser muito eficientes
- Gerenciamento de memória preciso: apagar dados sensiveis da memória
- Linguagem C cabe nesses dois requisitos
- Contudo, C também é muito perigoso

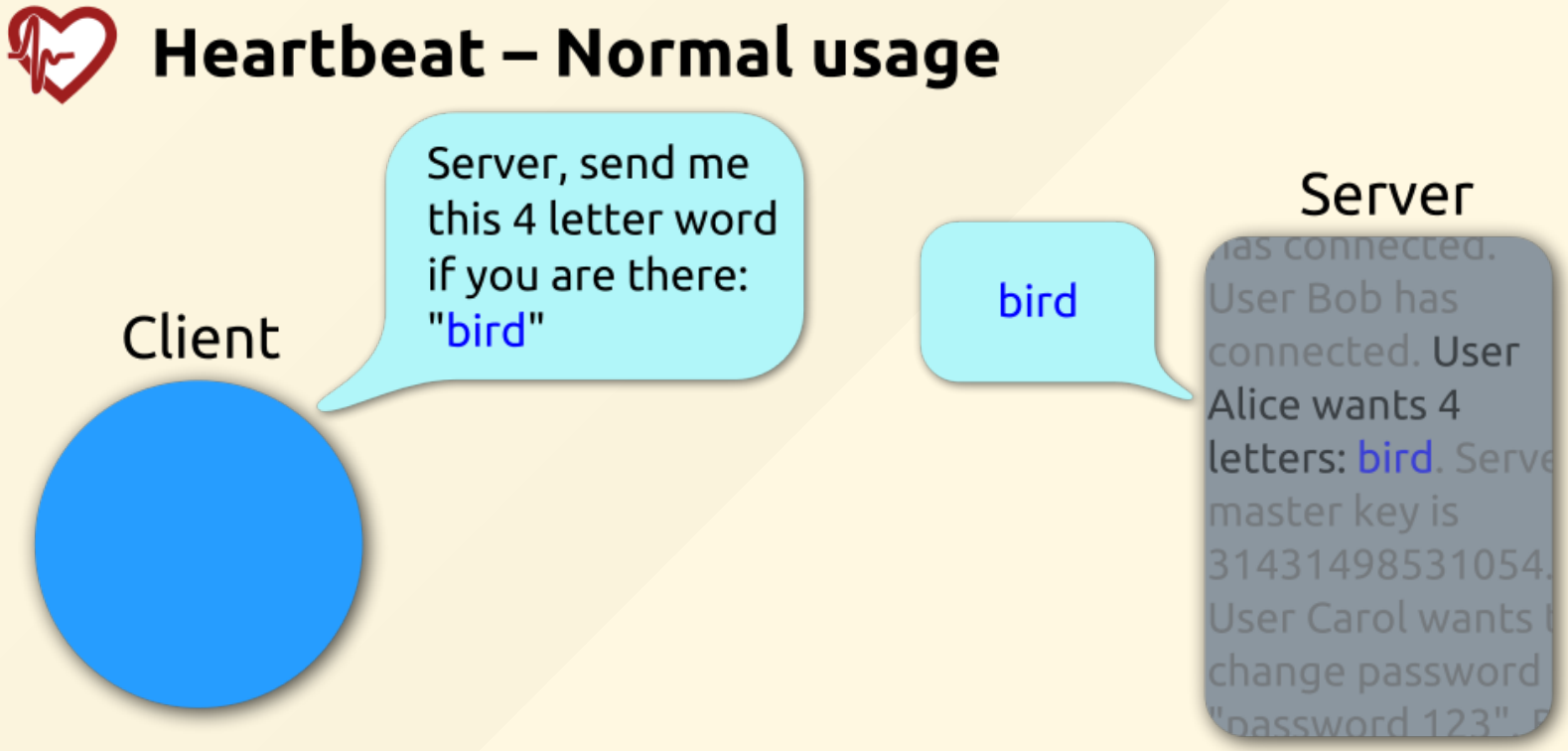
Boas práticas em C

- Validação da entrada
- Verificação de tamanhos
- "Higienize" a saída
- Limpeza das variáveis
- Checagem de erros
- Separação de codigos com privilégios diferentes
- Mantenha simples

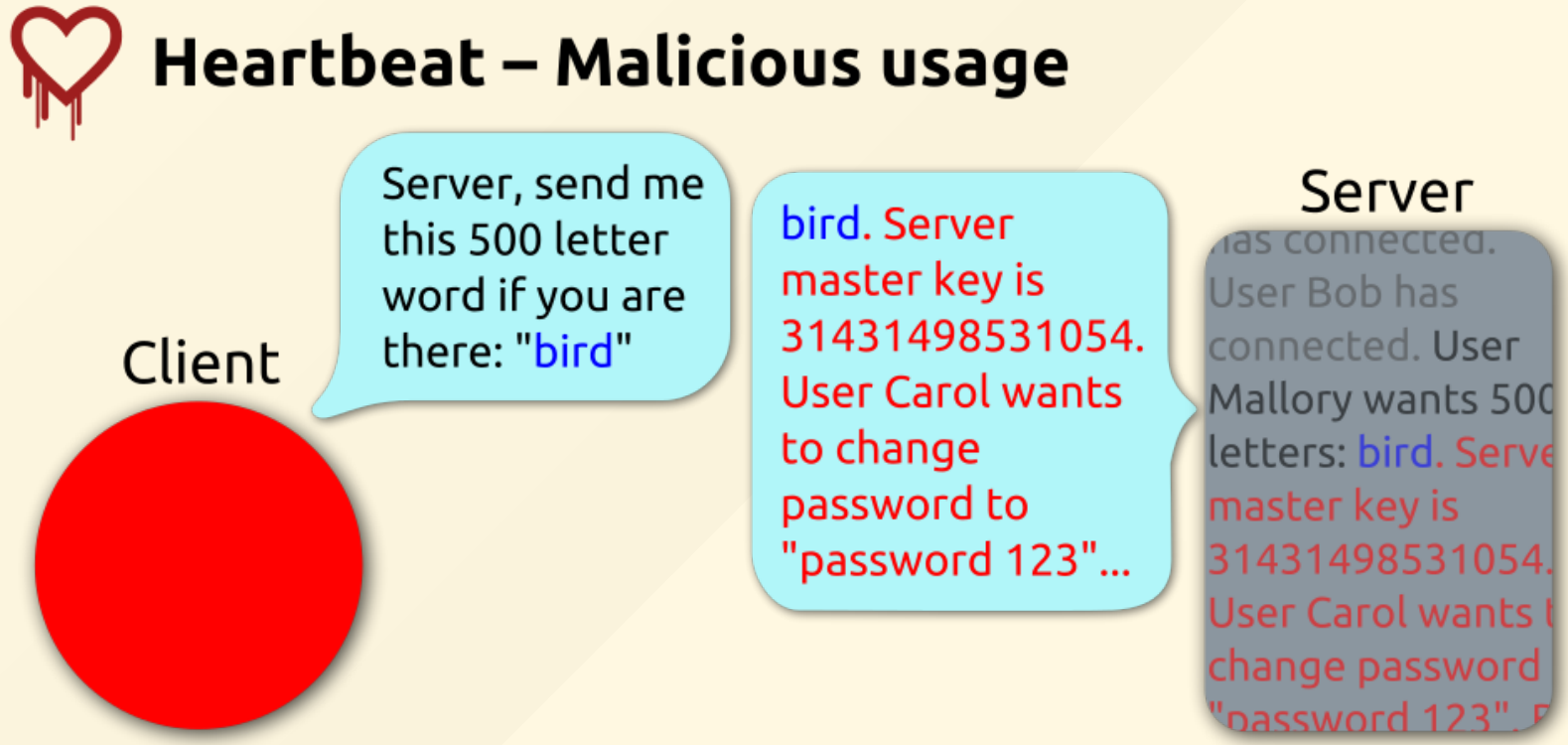
SSL da Apple

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
        OSStatus        err;
        ...

        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;
        ...

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

Heartbleed (OpenSSL)



Heartbleed (OpenSSL)



Sites e softwares afetados

- Google, YouTube, Gmail
- Facebook, Instagram
- Linkedin, Amazon, Dropbox
- Projeto Tor

OpenSSL

- Implementa o SSL, criptografia ponta-a-ponta entre cliente e servidor web
- Uma das implementações mais famosas e usadas hoje em dia, escrita em C
- Permite integração para acelaração de hardware e com padroes e certificaçõees de segurança
- Fornece um "front-end" para as chamadas ao kernel

OpenSSL

- OpenSSL prove três coisas:
  - `openssl-cli`: aplicação de linha de comando para realizar ações criptográficas (e.g. gerar certificado, criar par de chaves, cifrar arquivos, etc)
  - `libssl`: biblioteca para prover criptografia em rede, implementando o procotolo TLS/SSL
  - `libcrypt`: **prove operações criptográficas**

Exemplo

- No nosso caso, vamos escrever uma aplicação em C com uma operação de criptografia simétrica
- Precisamos saber o seguinte:
  - Algoritmo e modo a ser utilizado
  - Chave e vetor de inicialização

Exemplo

- Algoritmo criptográfico (ex AES, RSA)
- Modo do algoritmo (ECB, CBC)
- Chave criptográfica da mensagem
- Vetor de inicialização: parâmetro utilizado individualmente por cada mensagem cifrada, garante que mensagens iguais sejam cifradas de forma diferente

Vetor de inicialização

- Tony: `DCF0C50A96DC2D9B05F2AC4C24CB9B93`
- Lelê: `E0B554B341FF5632DE241FBF4B1DBB37`
- Tony: `6742FA9512C8C2ACE6942974C8C848FC`
- Lelê: `824783C3B272FF7129F9E153EC10D1AE`
- Tony: `C90BD345639368D951A8B5E267427514`
- Lelê: `E0B554B341FF5632DE241FBF4B1DBB37`
- Tony: `AC797939F55E87C361A8F02B4CEA1A08`
- Lelê: `824783C3B272FF7129F9E153EC10D1AE`

Vetor de inicialização

1. Seu nome é Leandro?
2. Seu nome é Italo?
3. Sua senha é `1234`?
4. Você gosta de presunto?

Vetor de inicialização

- 1. Tony: `DCF0C50A96DC2D9B05F2AC4C24CB9B93`
- 1. Lelê: `E0B554B341FF5632DE241FBF4B1DBB37`
- 2. Tony: `6742FA9512C8C2ACE6942974C8C848FC`
- 2. Lelê: `824783C3B272FF7129F9E153EC10D1AE`
- 3. Tony: `C90BD345639368D951A8B5E267427514`
- 3. Lelê: `E0B554B341FF5632DE241FBF4B1DBB37`
- 4. Tony: `AC797939F55E87C361A8F02B4CEA1A08`
- 4. Lelê: `824783C3B272FF7129F9E153EC10D1AE`

Vetor de inicialização

- 1. Tony: `DCF0C50A96DC2D9B05F2AC4C24CB9B93`
- 1. Lelê: **`E0B554B341FF5632DE241FBF4B1DBB37`**
- 2. Tony: `6742FA9512C8C2ACE6942974C8C848FC`
- 2. Lelê: ***`824783C3B272FF7129F9E153EC10D1AE`***
- 3. Tony: `C90BD345639368D951A8B5E267427514`
- 3. Lelê: **`E0B554B341FF5632DE241FBF4B1DBB37`**
- 4. Tony: `AC797939F55E87C361A8F02B4CEA1A08`
- 4. Lelê: ***`824783C3B272FF7129F9E153EC10D1AE`***

Programando do jeitinho OpenSSL

- No OpenSSL, trabalhamos com "contexto criptográfico"
- Uma estrutura `EVP_CIPHER_CTX` guarda tudo sobre a operação de cifragem
  - Inicializamos o contexto
  - Atualizamos o contexto
  - Finalizamos o contexto
- `if(função() != 1) trata_erro();`
  - Tudo é passível de erro!

Forks do OpenSSL

- LibreSSL
- BoringSSL

Programando

Vamos precisar de:

- Uma função de gerar a chave
- Uma função que, dado um arquivo e uma chave, cifra o arquivo
- Uma função para decifrar o arquivo

Olhando o código 👀

https://gitlab.com/andrealmeid/openssl-demo/

Referências

- https://people.freebsd.org/~syrinx/presentations/openssl/OpenSSL-Programming-20140424-01.pdf
- https://medium.com/@amit.kulkarni/encrypting-decrypting-a-file-using-openssl-evp-b26e0e4d28d4
- https://www.imperialviolet.org/2014/02/22/applebug.html
- https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption

Referências

- https://www.kernel.org/doc/html/v4.12/crypto/index.html
- https://en.wikipedia.org/wiki/AES_instruction_set
- https://software.intel.com/en-us/blogs/2012/01/11/aes-ni-in-laymens-terms