

ATIVIDADE 6

NOME: LUCAS MARTINS OLIVEIRA

MATRÍCULA: 202465058A

Ambiente de execução:

- Realizei a prática em um ambiente Linux Ubuntu 20.04

Preparação pré-prática:

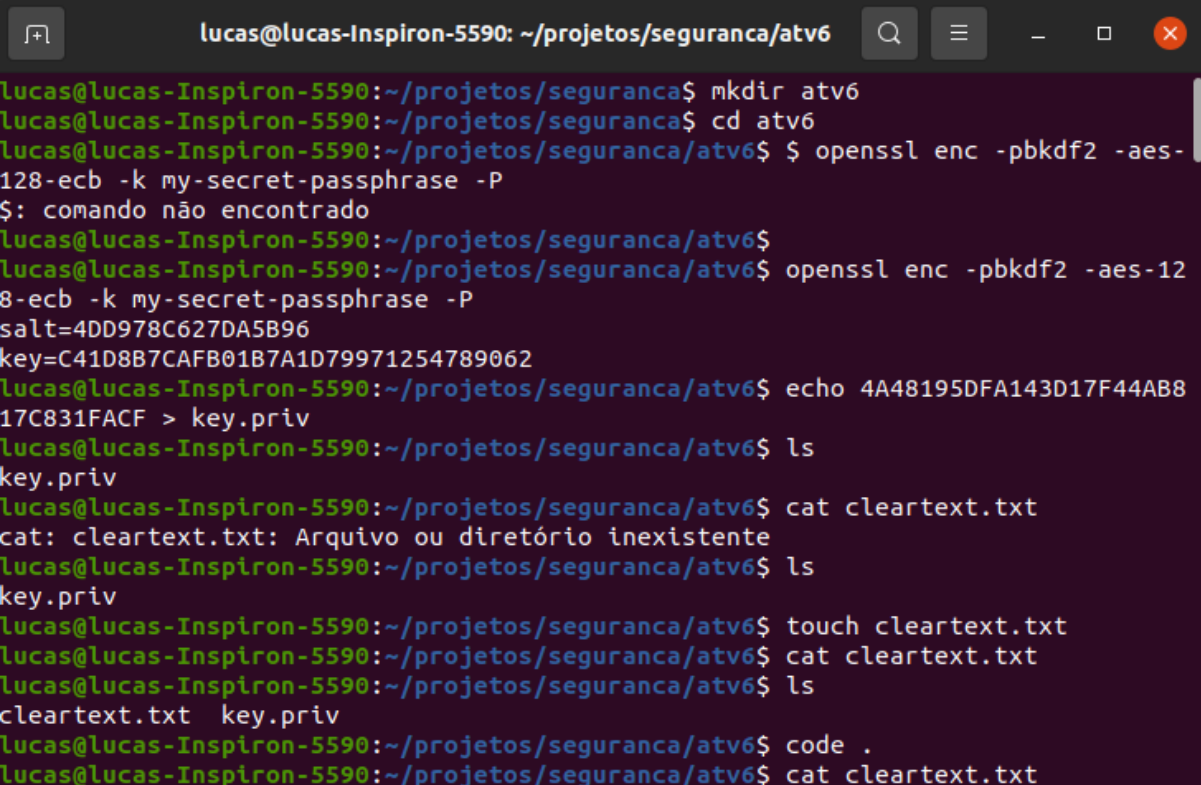
- Inicialmente, criamos um novo diretório o acessamos, isso nas duas primeiros comandos do terminal

- Em sequência, criamos uma chave AES de 128 bits usando a ferramenta OpenSSL

- Assim, após a realização desta etapa, foi nos retornado um saída com um “salt” e uma “key”, em que basicamente, a função do salt é evitar a repetição da mesma chave, no qual é apenas dados aleatórios usados como entrada adicional para a função hash

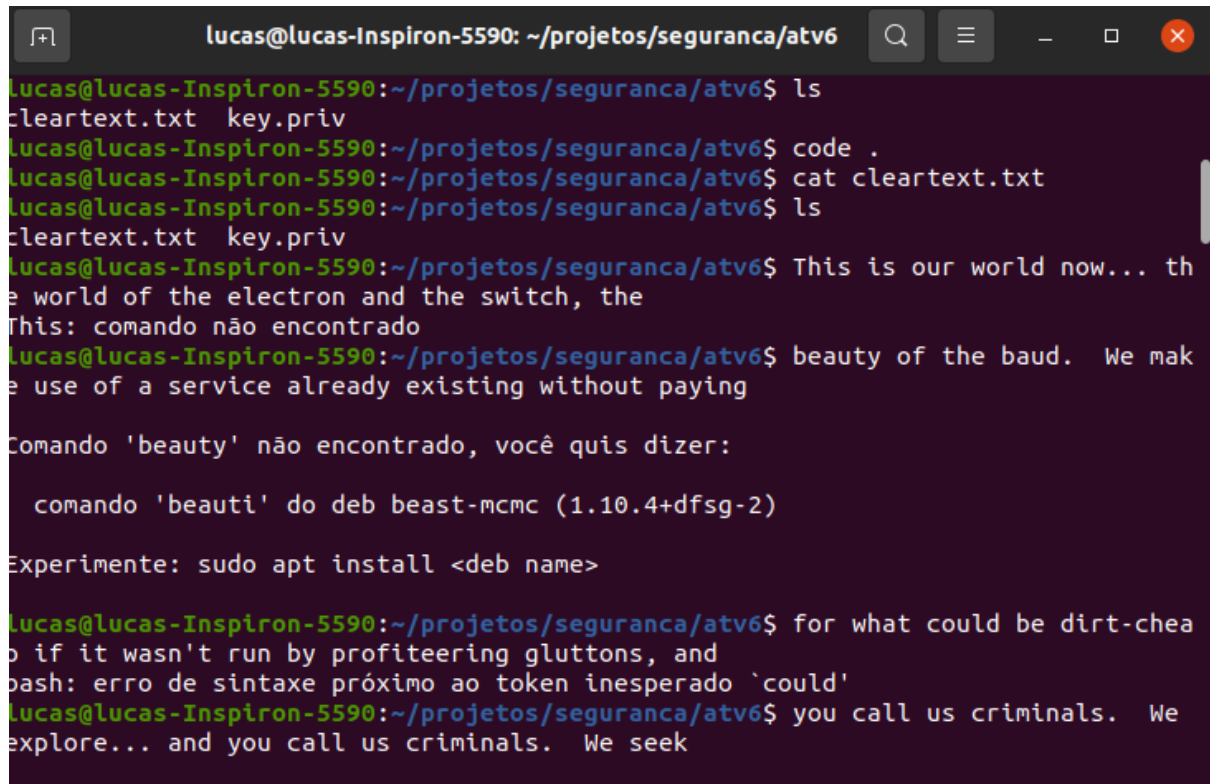
- Salvamos a chave em um arquivo “key.priv”, para assim realizar a criptografia AES

- Criamos o arquivo “cleartext.txt”



```
lucas@lucas-Inspiron-5590: ~/projetos/seguranca/atv6
lucas@lucas-Inspiron-5590:~/projetos/seguranca$ mkdir atv6
lucas@lucas-Inspiron-5590:~/projetos/seguranca$ cd atv6
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ $ openssl enc -pbkdf2 -aes-128-ecb -k my-secret-passphrase -P
$: comando não encontrado
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ openssl enc -pbkdf2 -aes-128-ecb -k my-secret-passphrase -P
salt=4DD978C627DA5B96
key=C41D8B7CAFB01B7A1D79971254789062
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ echo 4A48195DFA143D17F44AB817C831FACF > key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cat cleartext.txt
cat: cleartext.txt: Arquivo ou diretório inexistente
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ touch cleartext.txt
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cat cleartext.txt
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
cleartext.txt  key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ code .
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cat cleartext.txt
```

- Inserimos manualmente o texto fornecido de exemplo no arquivo, .txt, criado



```
lucas@lucas-Inspiron-5590: ~/projetos/seguranca/atv6
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
cleartext.txt  key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ code .
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cat cleartext.txt
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
cleartext.txt  key.priv
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ This is our world now... the
world of the electron and the switch, the
This: comando não encontrado
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ beauty of the baud. We mak
e use of a service already existing without paying

Comando 'beauty' não encontrado, você quis dizer:

comando 'beauti' do deb beast-mcmc (1.10.4+dfsg-2)
Experimente: sudo apt install <deb name>
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ for what could be dirt-chea
p if it wasn't run by profiteering gluttons, and
bash: erro de sintaxe próximo ao token inesperado 'could'
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ you call us criminals. We
explore... and you call us criminals. We seek
```

- Criamos agora também o arquivo, .txt, ciphertext.txt, usando comando touch
- E posteriormente, criptografamos a mensagem usando a chave a nossa chave obtida inicialmente, por meio novamente da ferramenta OpenSSL
- No qual, pode-se observar uma mensagem printada no terminal, em que ela fica irreconhecível

Prática 1:

- Inicialmente, baixamos a imagem
- Convertemos a imagem de .png, para o formato .ppm, para que possamos realizar as encriptações na imagem
- Dividimos o arquivo, em cabeçalho

```
lucas@lucas-Inspiron-5590: ~/projetos/seguranca/atv6
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ wget https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png
--2025-06-16 22:09:43-- https://upload.wikimedia.org/wikipedia/commons/a/af/Tux.png
Resolvendo upload.wikimedia.org (upload.wikimedia.org)... 2a02:ec80:700:ed1a::2:b, 195.200.68.240
Conectando-se a upload.wikimedia.org (upload.wikimedia.org)|2a02:ec80:700:ed1a::2:b|:443... conectado.
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 11913 (12K) [image/png]
Salvando em: "Tux.png"

Tux.png          100%[=====] 11,63K  --.-KB/s   em 0s

2025-06-16 22:09:44 (81,7 MB/s) - "Tux.png" salvo [11913/11913]

lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
ciphertext.txt  cleartext.txt  key.priv  Tux.png
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ convert Tux.png tux.ppm
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
ciphertext.txt  cleartext.txt  key.priv  Tux.png  tux.ppm
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ head -n 3 tux.ppm > header.txt
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
```

- Depois em corpo
- Após isso, encriptamos o corpo e voltamos com cabeçalho junto ao corpo

```
lucas@lucas-Inspiron-5590: ~/projetos/seguranca/atv6
lucas@lucas-Inspiron-5590:~$ cd projetos/seguranca/atv6/
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
body.bin      ciphertext.txt  header.txt  tux-ecb-enc.ppm  tux.ppm
body.ecb.enc  cleartext.txt  key.priv   Tux.png
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ tail -n +4 tux.ppm > body.bin
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ openssl enc -aes-128-ecb -in body.bin -K $(cat key.priv) -out body.ecb.enc
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cat header.txt body.ecb.enc > tux-ecb-enc.ppm
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ ls
body.bin      ciphertext.txt  header.txt  tux-ecb-enc.ppm  tux.ppm
body.ecb.enc  cleartext.txt  key.priv   Tux.png
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ code .
```

- E como resultado, obtemos:
- E este pode ser explicado pelo fato de que no modo ECB, cada bloco de texto simples é criptografado individualmente e independentemente com a mesma chave



PRÁTICA 2:

- Para realização desta, clonei o repositório da prática que fiz a prática 1, e o renomeie como prática _2
- Inicialmente geramos o vetor de inicialização usando “rand”, novamente, da ferramenta OpenSSL
- E assim, encriptamos usando OpenSSL passando o modo -aes-128-cbc e o Vetor de Inicialização com a opção -iv
- Agora adicionamos o cabeçalho a imagem criptografada


```

lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ code .
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6$ cd pratica_2
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ ls
body.bin      ciphertext.txt  header.txt      tux-ecb-enc.ppm  tux.ppm
body.ecb.enc  cleartext.txt   key.priv        Tux.png
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ openssl rand -hex
16 > iv.txt
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ ls
body.bin      ciphertext.txt  header.txt      key.priv          Tux.png
body.ecb.enc  cleartext.txt   iv.txt          tux-ecb-enc.ppm  tux.ppm
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ cat iv.txt
c78942665ef122e234a1c88059a28032
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ openssl enc -aes-
128-cbc -in body.bin -K $(cat key.priv) -iv $(cat iv.txt) -out body.cbc.enc
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$ cat header.txt bo
dy.cbc.enc > tux-cbc-enc.ppm
lucas@lucas-Inspiron-5590:~/projetos/seguranca/atv6/pratica_2$

```

- Imagem resultante:

- E este pode ser explicado pelo fato de que No modo CBC, um bloco criptografado (texto cifrado) é usado como entrada para criptografar o próximo bloco



Texto sobre explicando o <https://sweet32.info/>

O artigo/paper aborda uma vulnerabilidade em algoritmos de criptografia de bloco mais antigos, como o Triple-DES e o Blowfish. No qual, a comunidade criptográfica já tinha conhecimento de que esses algoritmos, em virtude de utilizarem blocos de 64 bits, eram teoricamente suscetíveis a ataques de aniversário. O que o artigo demonstra é que tais ataques se tornaram praticamente aplicáveis em cenários reais, notadamente no TLS (utilizado para HTTPS) e no OpenVPN, onde esses algoritmos de cifra ainda são amplamente empregados. Em essência, mesmo que o algoritmo em si não apresente falhas criptográficas diretas, o tamanho reduzido do bloco o torna vulnerável a colisões de dados que podem comprometer a confidencialidade de informações sensíveis.

A problemática reside no fato de que algoritmos de cifra de bloco que operam com blocos de 64 bits (a exemplo do Triple-DES e Blowfish) são consideravelmente mais suscetíveis a "ataques de aniversário" do que aqueles que utilizam blocos de 128 bits (como o AES). Isso ocorre porque, em modos de operação comuns, como o CBC, a segurança de uma cifra de bloco começa a se degradar quando a quantidade de dados criptografados sob a mesma chave se aproxima de um limite crítico. Para ciphers de 64 bits, este limite corresponde a aproximadamente 32 GB. Acima desse volume, a probabilidade de ocorrência de colisões entre blocos de texto cifrado aumenta significativamente. Se um atacante for capaz de monitorar uma conexão de longa duração (como uma sessão HTTPS ou VPN) e injetar tráfego malicioso (por exemplo, via JavaScript em um navegador), ele pode forçar a ocorrência dessas colisões e, conseqüentemente, inferir informações sensíveis, como cookies de sessão ou credenciais de autenticação. O artigo em questão demonstra a viabilidade prática desse ataque, permitindo a recuperação de segredos em menos de dois dias.

Portanto, a partir da leitura pude observar que o aspecto mais notável desta pesquisa é a forma como uma vulnerabilidade teoricamente conhecida há anos se materializa agora como uma ameaça real e prática, impulsionada pela evolução da capacidade computacional e pela persistência no uso de cifras legadas. A constatação de que 1-2% das conexões HTTPS ainda empregam Triple-DES e que o Blowfish é o algoritmo padrão no OpenVPN representa um risco desnecessário para a segurança dos dados. Compreendo que, enquanto futuros profissionais da área, devemos estar em constante atualização e jamais subestimar o impacto de falhas que possam parecer "pequenas" ou "antigas". A priorização de cifras mais robustas (como AES-128 ou 256) e a implementação da renegociação frequente de chaves são medidas cruciais para mitigar esses riscos e assegurar a proteção de nossas aplicações e redes. Trata-se, portanto, de um processo contínuo de conscientização e aprimoramento em segurança cibernética.