Execução de código arbitrário na urna eletrônica brasileira

SBSeg 2018

Diego Aranha (Unicamp), Pedro Barbosa (UFCG),
Thiago Cardoso (Hekima), Caio Lüders (UFPE),
Paulo Matias (UFSCar)

Propriedades de segurança

Não importando a tecnologia empregada, um sistema de votação precisa satisfazer algumas propriedades:

1. *Autenticação dos eleitores*: apenas eleitores autorizados podem votar
2. *Sigilo do voto*: voto deve ser secreto
3. *Integridade dos resultados*: resultado é justo
4. *Possibilidade de auditoria*: idealmente, sem especialização

Um breve histórico

1996 : Urnas eletrônicas em 30% das seções eleitorais

2000 : Primeiras eleições inteiramente eletrônicas

2002 : Primeira experiência com voto impresso

2006 : TSE passa a ser responsável pelo *software*

2008 : Migração para GNU/Linux

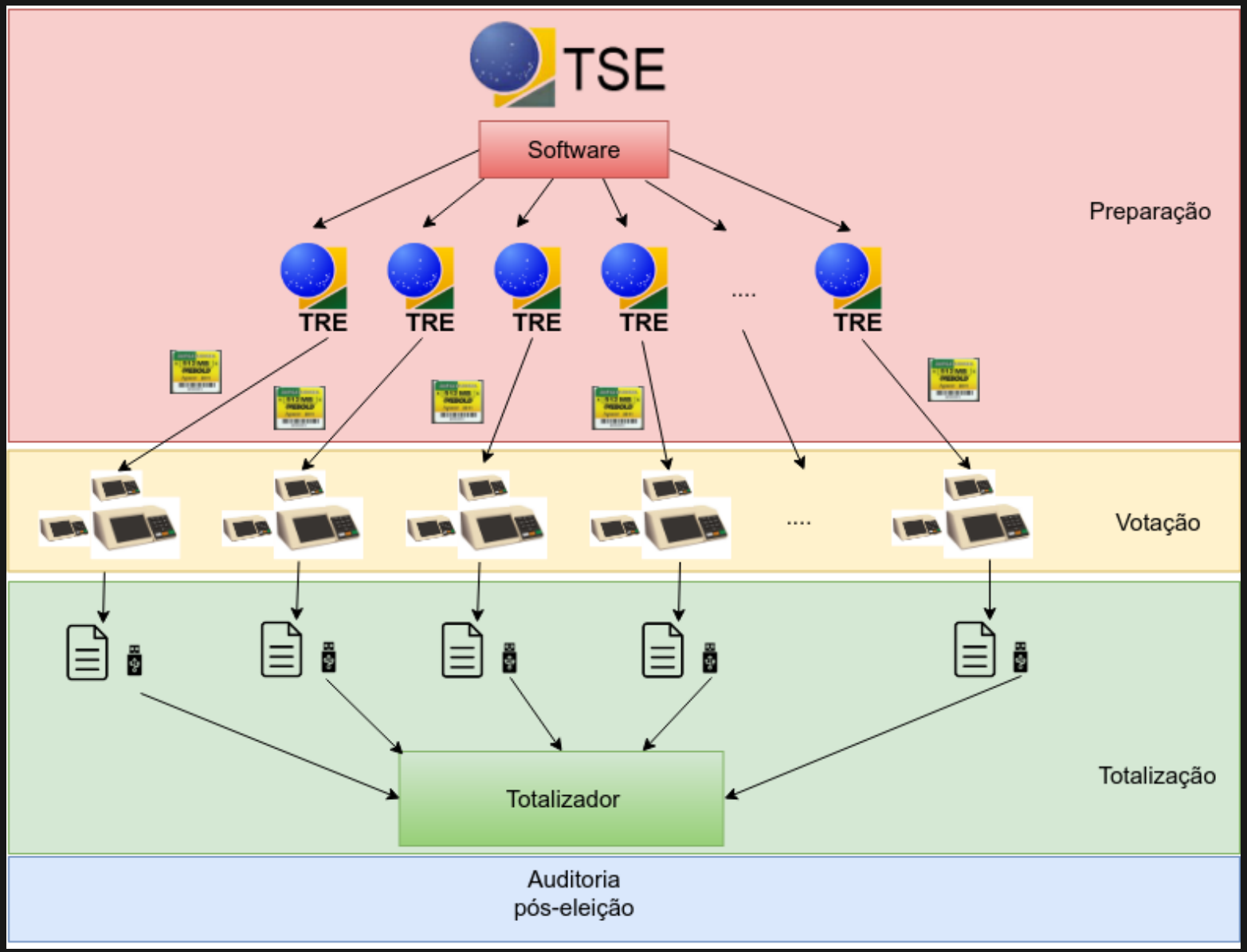
2009 : I Testes Públicos de Segurança (quebra de sigilo do voto)

2012 : II TPS (quebra de sigilo do voto)

2016 : III TPS (quebra na integridade de resultados)

2017 : IV TPS (quebra na integridade de *software*)

Processo de votação brasileiro

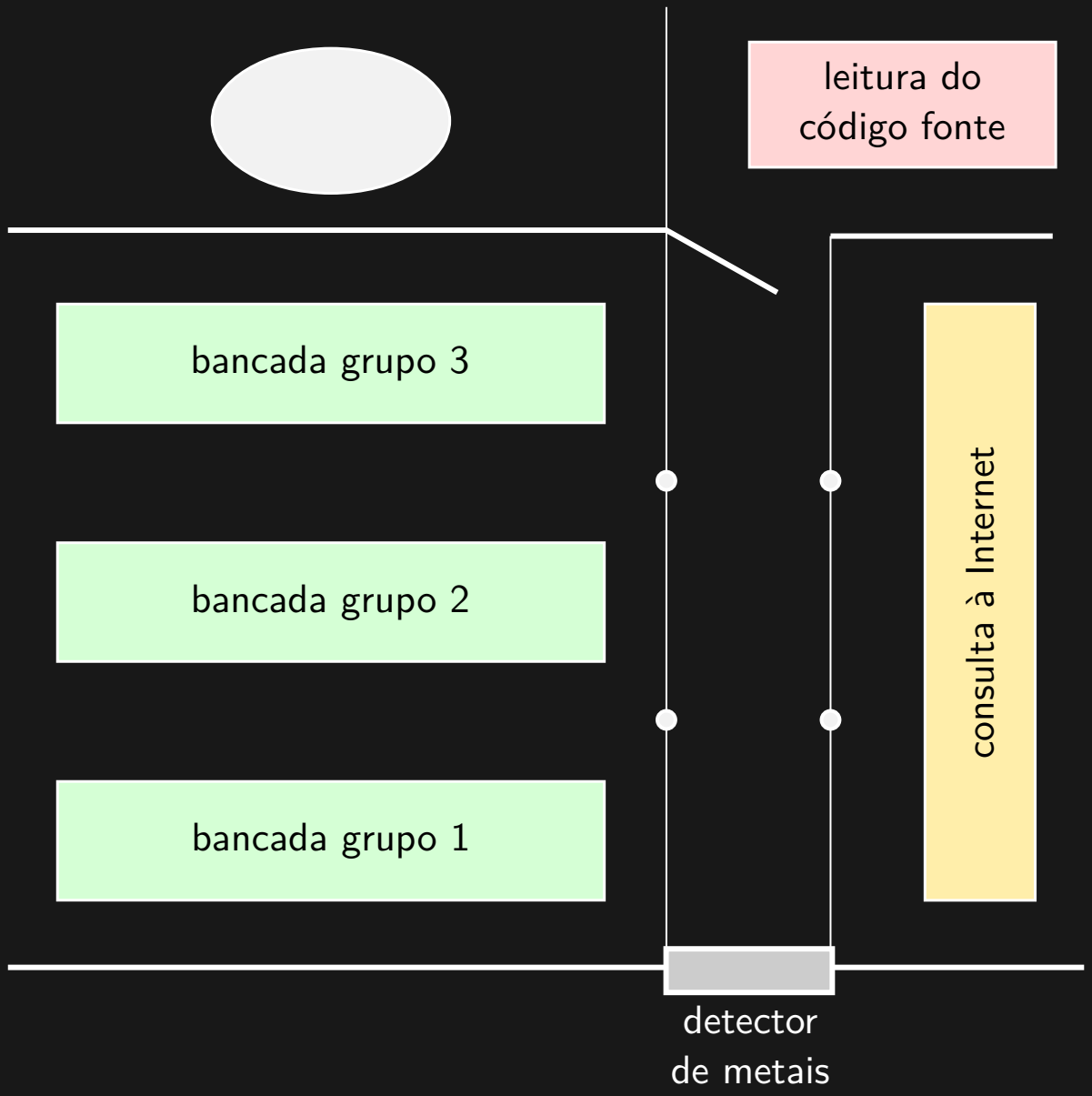


Instalação (carga) nas urnas

Como funciona o TPS?

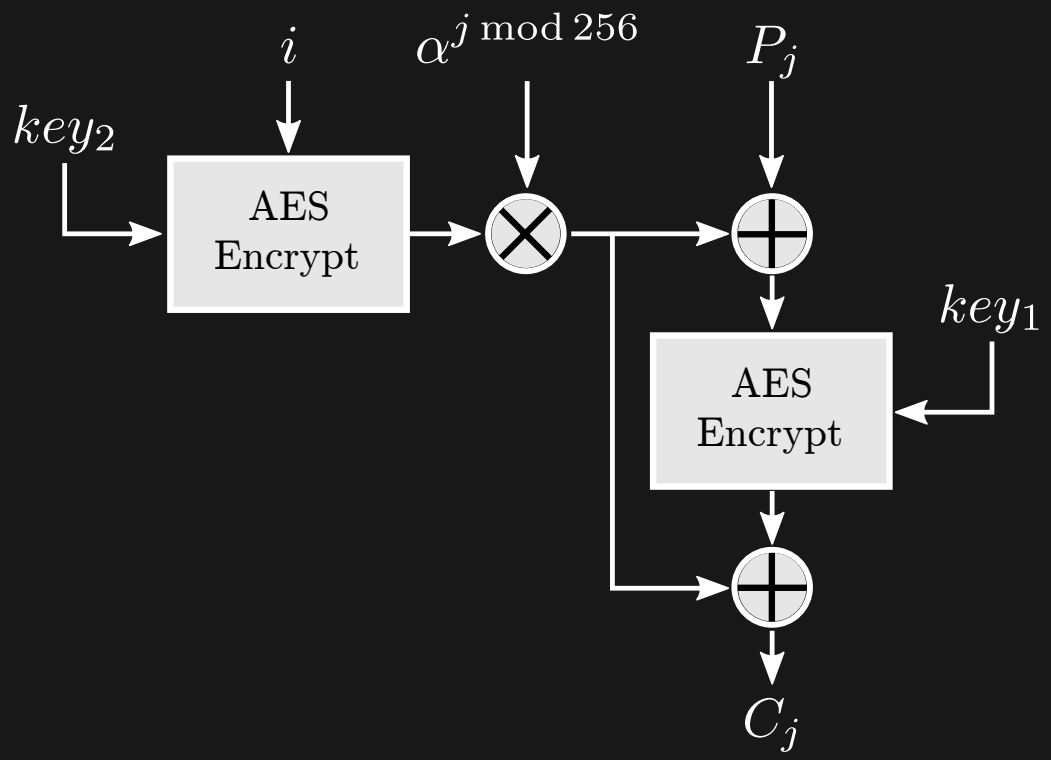
- Fase de inspeção dos códigos fonte
- Submetemos **planos de teste**
- TSE analisa e aprova os planos de teste
- Executamos os planos de teste em uma bancada com computador e urna eletrônica

Planta do ambiente



Fase de inspeção dos códigos fonte

Encontramos chave da mídia de instalação em claro no código fonte do *kernel* 3.18.



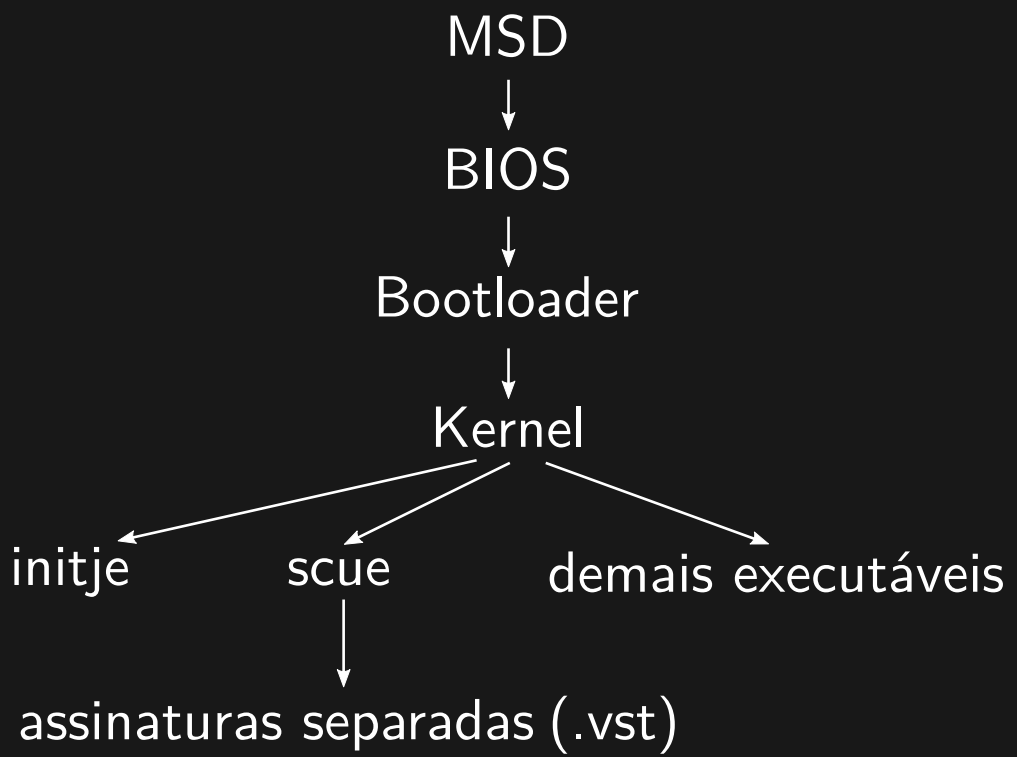
Planos de teste inicialmente submetidos

- **QP1**: É possível extrair chaves criptográficas da FC?
- **QP2**: É possível violar o sigilo do voto explorando o gerador de números pseudo-aleatórios?
- **QP3**: É possível inserir um dispositivo USB malicioso na urna?
- **QP4**: É possível executar código remoto na plataforma *web* de totalização?

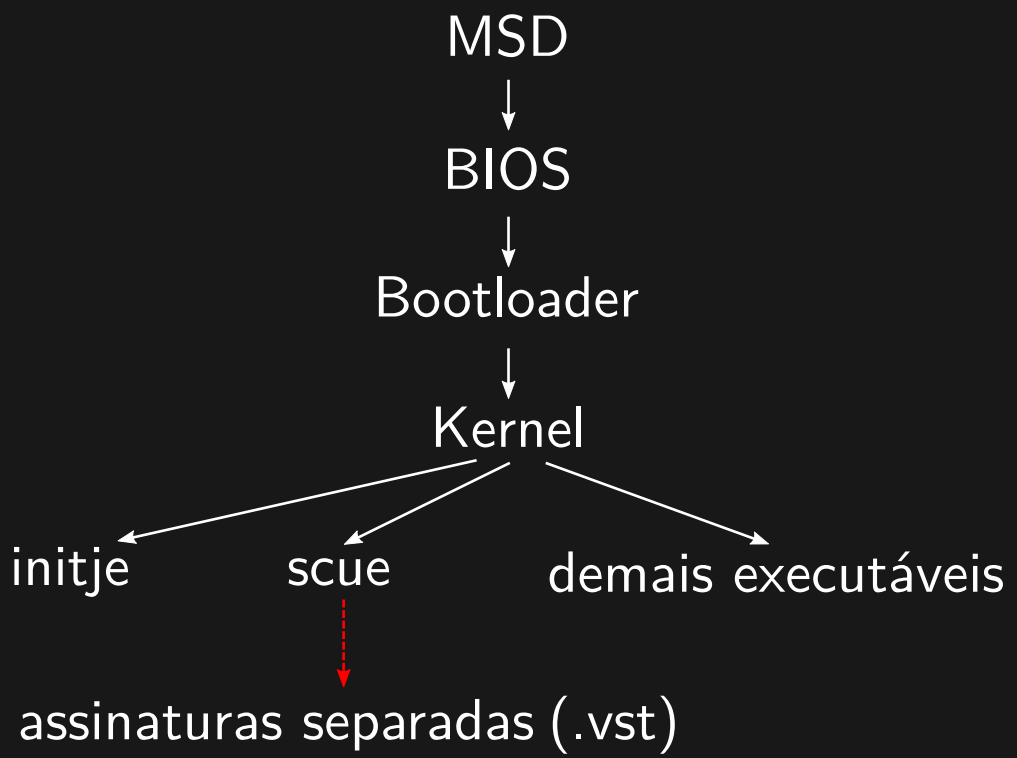
Chave encontrada no *kernel*: um atalho

```
class Codebook:
    def __init__(self, key):
        self.key = key

    def aes(self, key, block, mode='-d'):
        if mode == '-d':
            block += block  #need 256 bits to work with
                            #the command below
        p = Popen(['openssl','enc','-aes-256-ecb',mode,
                   '-K',key.encode('hex')],
                  stdin=PIPE,stdout=PIPE,
                  stderr=open('/dev/null','w'))
        p.stdin.write(block)
        p.stdin.close()
        return p.stdout.read()[:16] #but we ignore the last 128 bits

    def decrypt(self, block):
        return self.aes(self.key, block, '-d')

    def encrypt(self, block):
        return self.aes(self.key, block, '-e')
```

Inspeção da cadeia de confiança



Elo fraco: assinaturas separadas



Elo fraco: assinaturas separadas

- Ausência de assinatura das bibliotecas **libhkdf.so** e **libapilog.so** nos arquivos de assinaturas separadas (VST).
- Interferir com código do SCUE por meio de bibliotecas ligadas a este permitiria contornar totalmente VSTs, pois o *kernel* só verificava o binário principal.

- Introdução da **QP5**: É possível executar código arbitrário na urna eletrônica?
- Tempo restrito: equipe decidiu deixar de investigar **QP2–4**.

Execução arbitrária de código

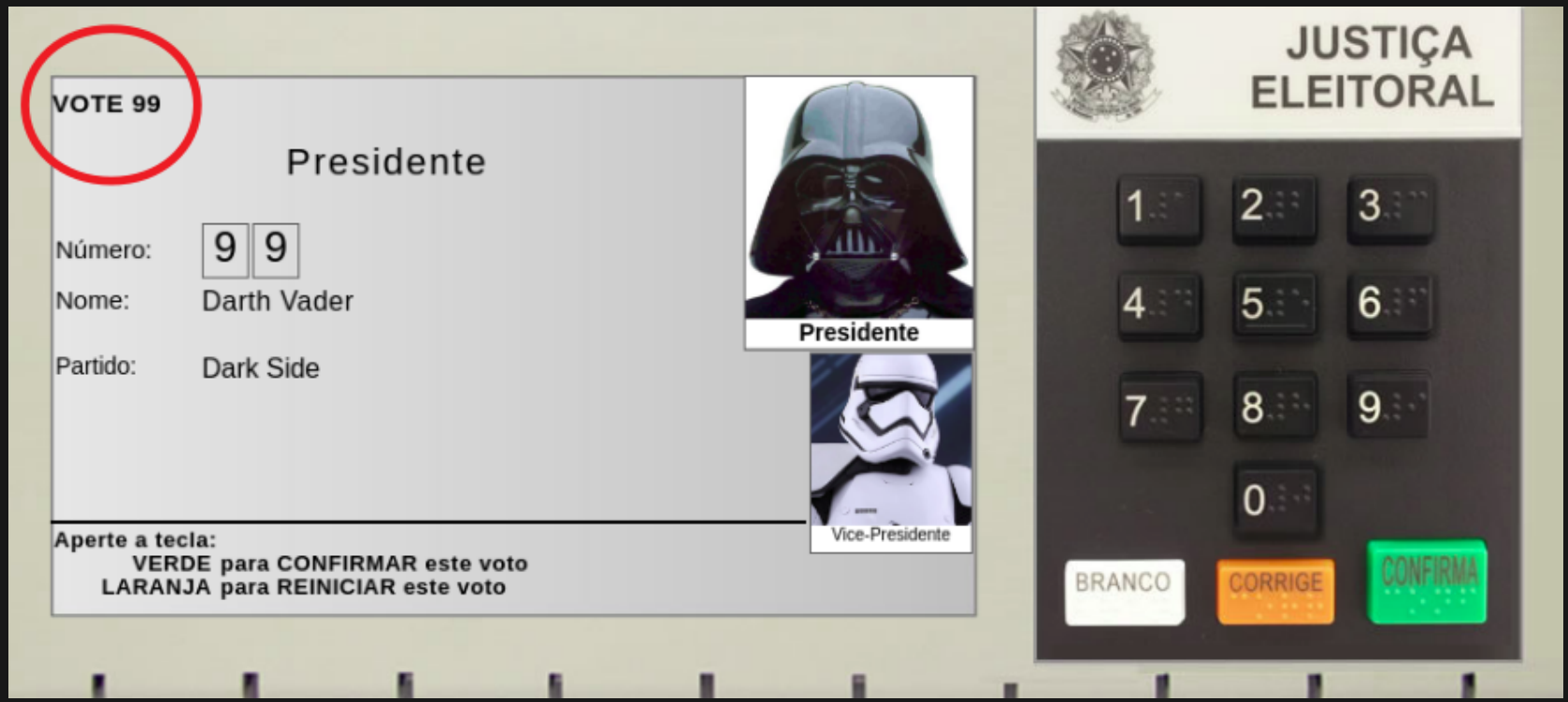
Método: Alterar **libhkdf.so** ou **libapilog.so** para injetar código malicioso.

Diversas demonstrações dando suporte a **QP5**:

- Imprimir "`FRAUDE!`" durante a inicialização da urna.
- Manipular *log* para mostrar "`XXXX`" em vez de "`INFO`".
- Quebrar o sigilo do voto, por meio de mudança da chave do registro digital do voto (RDV), permitindo contar votos antes e depois de um eleitor entrar na seção:

  `[98, 99, null, null]` $\rightarrow$ `[98, 99, null, 99]`
- Interferir sobre páginas de memória somente leitura:
  "`A Hora da Estrela`" $\rightarrow$ "`A Hora da Treta`".

Demonstração: interferir sobre a interface gráfica



Demonstração: interferir com armazenamento dos votos

Método: Alterar código do método `void AdicionaVoto(uint8_t cargo, int tipo, std::string &voto)`, da classe `InfoEleitor` do aplicativo de votação.

- Inserir uma instrução `ret` causou erro de asserção (tamanho do `std::vector` da cédula deveria ser diferente de zero) apenas depois de pressionado "Confirma".
- Código para desvio de votos obteve sucesso em um simulador de urna:

```
mov eax, [ebp+0x14]  ; std::string&
mov edi, [eax]       ; char*
mov al, '9'
stosb
stosb
```

Término do TPS 2017

- Grupo da Polícia Federal deu suporte à **QP1** (É possível extrair chaves criptográficas da FC), obtendo a chave do sistema de arquivos por meio de extração da memória de uma máquina virtual.
- Análise superficial da **QP2** (É possível violar o sigilo do voto explorando o gerador de números pseudo-aleatórios?): urna utiliza variante do gerador Sapparot-2 (não criptográfico) intercalado com gerador do Linux.
- Demos suporte à **QP5** (É possível executar código arbitrário na urna eletrônica) por meio de diversas demonstrações.

Contramedidas adotadas pelo TSE

- Uso de trecho da Flash da BIOS como chave para cifrar a mídia.
  Limitação: Flash da BIOS pode ser extraída por um atacante uma única vez, de qualquer urna do país.
- Compilação de binários PIE.
  Limitação: Endereços podem ser calculados a partir de informações contidas na pilha.
- Inclusão de todos os arquivos nos VSTs.
  Limitação: Verificação pode ser desabilitada interferindo-se com SCUE.
- Ligação estática com libapilog, libhkdf, *etc.*
  Limitação: A libc poderia ser infectada.
- Verificação de assinatura de bibliotecas pelo *kernel*.
  Limitação: Cadeia de confiança complexa, muito código customizado.
  É difícil atestar que não existem outras falhas.

Conclusão

Recomenda-se ao TSE:

- Reduzir a burocracia e ampliar escopo e duração dos testes.
- Revisar práticas internas de desenvolvimento.
- Considerar novamente a introdução de um registro físico do voto.

Mais informações em

https://github.com/epicleet/tps2017

https://urnaeletronica.info