Execução de código arbitrário na urna eletrônica brasileira

Diego F. Aranha, Pedro Barbosa, Thiago N. C. Cardoso, Caio Lüders, Paulo Matias

Unicamp, UFCG, Hekima, UFPE, UFSCar

Abstract. *Multiple serious vulnerabilities were detected during the last Public Security Tests of the Brazilian voting system. When combined, they allowed to compromise the main security properties of the system, namely ballot secrecy and* software *integrity. The insecure storage of cryptographic keys, hard-coded directly in source code and shared among all machines, allowed full content inspection of the* software *installation memory cards, after which two shared libraries missing authentication signatures were detected. Injecting code in those libraries opened the possibility of having complete control over the machine. As far as we know, this was the most in-depth exploitation of an official large-scale voting system ever performed during severely restricted tests.*

Resumo. *Múltiplas vulnerabilidades graves foram detectadas nos últimos Testes Públicos de Segurança da urna eletrônica brasileira. Quando combinadas, comprometeram o sigilo do voto e a integridade do* software*, as principais propriedades de segurança do sistema. O armazenamento inseguro de chaves criptográficas, inseridas diretamente no código-fonte e compartilhadas entre todas as urnas, permitiu a inspeção das mídias de instalação de* software*, em que duas bibliotecas compartilhadas sem assinaturas foram encontradas. A injeção de código nessas bibliotecas abriu a possibilidade de ter total controle sobre a urna. Até onde sabemos, esta foi a exploração mais profunda de um sistema de votação utilizado em larga escala realizada durante testes severamente restritos.*

1. Introdução

O Brasil é uma das maiores democracias do mundo, em que mais de 144 milhões de eleitores votarão eletronicamente nas próximas eleições gerais. As urnas eletrônicas foram introduzidas no país em 1996, após relatos frequentes de fraude durante o transporte e contagem de cédulas eleitorais. As eleições tornaram-se inteiramente eletrônicas no ano 2000, rapidamente seguidas em 2002 pela primeira experiência com uma trilha física de auditoria: o chamado *voto impresso*, que terminou descontinuado após alegações de alto custo e complexidade logística. Desde então, múltiplas tentativas de reintroduzir o recurso falharam, sendo a mais recente suspensa pelo Supremo Tribunal Federal.

Como resposta aos pedidos frequentes por maior transparência, o Tribunal Superior Eleitoral (TSE) começou a organizar Testes Públicos de Segurança (TPS) em 2009. Nesses eventos, investigadores independentes pré-aprovados podem examinar os mecanismos de segurança implementados no sistema por alguns dias, encontrar vulnerabilidades e sugerir correções. Após as primeiras edições em 2009 e 2012, o evento tornou-se mandatório no calendário eleitoral e agora acontece geralmente no ano anterior às eleições. Mesmo com acesso limitado às urnas eletrônicas, os TPS têm se mostrado uma alternativa útil para realizar análises independentes de segurança. A primeira com acesso ao código-fonte do sistema foi realizada em 2012 [Aranha et al. 2014], quando a equipe vencedora foi capaz

de montar um ataque contra o sigilo do voto baseado na geração insegura de números aleatórios; e documentar outros problemas de segurança no *software* de votação, como o armazenamento e compartilhamento inseguro de chaves criptográficas e a verificação insuficiente de integridade, resultados de um processo inseguro de desenvolvimento conduzido sob modelo de adversário inadequado.

Contribuições. Este trabalho continua e atualiza os esforços de análise de segurança da urna eletrônica, baseado nos resultados obtidos na edição de 2017 dos TPS. Em resumo, foi possível:

(i) Recuperar a chave criptográfica simétrica que protege as mídias de instalação de *software*, permitindo a leitura de seu conteúdo às claras. Como a chave é compartilhada entre todas as urnas, sua recuperação em uma eleição real poderia ter impacto nacional.
(ii) Detectar duas bibliotecas compartilhadas sem verificação de integridade, permitindo injeção direta de código arbitrário em suas funções para manipular seu comportamento e dos programas ligados, como o aplicativo oficial de votação.
(iii) Explorar a capacidade de injeção de código para violar o sigilo do voto de eleitores específicos a partir da manipulação de chaves criptográficas geradas dinamicamente; emitir comandos a partir de um teclado comum acoplado à urna; e manipular o registro cronológico de eventos gerados pelo *software* de votação.
(iv) Injetar código para manipular dinamicamente mensagens exibidas na interface gráfica, para orientar o eleitor a votar em certos candidatos ou partidos hipotéticos.

A capacidade de injetar código foi ainda estendida para manipular o resultado de uma eleição simulada, pois todos os requisitos estavam presentes. Foi obtido progresso substancial nessa direção, demonstrando-se que era possível impedir o armazenamento dos votos na cédula armazenada em memória, o que disparou um erro de consistência no *software*. O ataque foi então adaptado para manipular os votos antes de armazená-los em memória, mas o evento foi encerrado antes que a versão definitiva do ataque pudesse ser executada no equipamento.

Organização. A Seção 2 apresenta o sistema de votação eletrônica brasileiro e a Seção 3 resume resultados de análises de segurança anteriores. A Seção 4 descreve a preparação da equipe e observações coletadas durante a inspeção de código para formular os planos de teste. A Seção 5 detalha a execução dos planos de teste e achados correspondentes. A Seção 6 discute os resultados considerando alegações oficiais do TSE e a última seção finaliza o artigo com conclusões e perspectivas futuras.

2. Visão geral do sistema

As urnas eletrônicas foram introduzidas nas eleições de 1996 em 56 municípios. As primeiras máquinas foram fabricadas pela Unisys e equipadas com processadores Intel 80386. Em termos de *software*, empregavam um sistema operacional nacional compatível com o DOS chamado VirtuOS. Modelos posteriores mantiveram a mesma aparência e interface, mas adotaram outros componentes de *software*, como o sistema operacional Windows CE, com aplicativos desenvolvidos pela empresa Procomp, subsidiária brasileira da Diebold. Posteriormente, o sistema operacional GNU/Linux foi adotado e o desenvolvimento passou a ser responsabilidade do TSE. Os modelos mais recentes da urna eletrônica, introduzidos em 2009 e 2015, incluem um módulo de segurança em *hardware* (chamado MSD – *Master*

Security Device) para realizar tarefas críticas de segurança, como armazenar chaves criptográficas e verificar a integridade do *software* durante a inicialização. Em 2008, o Tribunal iniciou o cadastramento biométrico, atingindo recentemente metade da população.

2.1. Equipamento

A urna eletrônica (Figura 1) realiza a gravação digital direta do voto, sendo classificada como DRE (*Direct Recording Electronic*), ou seja, não há a presença de qualquer tipo de registro físico do voto conferível pelo eleitor. A máquina é composta por um terminal do mesário, utilizado para digitar comandos, títulos de eleitor ou coletar biometria; e outro terminal utilizado pelo eleitor para votar. Ambos os terminais são conectados por um cabo físico, uma decisão de projeto questionável do ponto de vista do sigilo do voto, já que o terminal do eleitor possui acesso tanto à identidade como às escolhas de cada eleitor [van de Graaf and Custódio 2002]. Além dos teclados de entrada, a comunicação com a urna é possível a partir de cartões de memória responsáveis por instalar o sistema operacional e componentes de *software* (*flash* de carga – FC) na memória interna da urna; e por armazenar os metadados de candidatos e eleitores (*flash* de votação – FV). Durante uma eleição, os arquivos são armazenados de forma redundante tanto na memória interna quanto na FV. Há uma porta USB na parte traseira da urna para se inserir um dispositivo chamado Memória de Resultados (MR), que armazena resultados e demais arquivos públicos da eleição. O acesso a todas as interfaces externas é protegido por lacres assinados por um juiz em cerimônia oficial.

Figura 1. A urna eletrônica brasileira e seus dois terminais.

2.2. Procedimento oficial

Na fase de desenvolvimento, o *software* é atualizado e inspecionado por instituições autorizadas, sob Termo de Sigilo, até poucos meses antes das eleições. Uma versão do *software* é compilada na cerimônia oficial de lacração e distribuída para os Tribunais Regionais Eleitorais (TREs), onde termina armazenada nas FCs geradas pelo sistema GEDAI-UE (Gerenciador de Dados, Aplicativos e Interface com a Urna Eletrônica). Esses cartões de memória, por sua vez, são transportados até as urnas para instalar o *software* oficial alguns dias antes das eleições. Segundo informação do TSE, cada FC instala até 50 urnas eletrônicas distintas. Ao contrário de outros países, a logística de distribuição permite atualizar o *software* a cada eleição.

O processo de instalação do *software* da FC na urna eletrônica é implementado pelo módulo SCUE (Sistema de Carga da Urna Eletrônica), armazenado na própria FC e utilizado para copiar os arquivos para a memória interna após auto-verificação de integridade. Após a carga, a máquina já pode ser inicializada pela primeira vez diretamente

da memória interna, quando esta executa também um teste de *hardware*. A atribuição de uma urna eletrônica para uma sessão eleitoral é autenticada por uma assinatura digital curta e o resultado é chamado Resumo de Correspondência. Uma base de dados contendo as atribuições é publicada posteriormente. No dia da eleição, executa-se o mesmo procedimento em todas as seções eleitorais:

1. Entre 7h e 8h da manhã, imprime-se a zerésima, documento oficial que supostamente atesta que nenhum voto foi previamente computado para qualquer candidato;
2. O mesário responsável abre a votação às 8h;
3. Os eleitores utilizam a urna eletrônica para inserir suas escolhas;
4. O mesário responsável encerra a votação às 17h;
5. Emitem-se vias do Boletim de Urna (BU) em papel, contendo a totalização parcial dos candidatos. Cópias do documento físico são assinadas, distribuídas e fixadas em lugar visível da seção eleitoral;
6. Gravam-se os chamados produtos públicos de votação, cujos principais componentes são a versão digital do BU para totalização, o registro cronológico de eventos (LOG) e o Registro Digital do Voto (RDV). Este último consiste em uma lista fora de ordem dos votos recebidos pela urna. Esses arquivos são assinados digitalmente e armazenados na MR.
7. O mesário rompe o lacre e retira a MR contendo os produtos públicos da eleição;
8. Os produtos públicos armazenados na MR são transmitidos para o totalizador a partir de uma rede privada, em um computador inicializado com um LiveUSB dedicado produzido pela Justiça Eleitoral e denominado JE Connect.

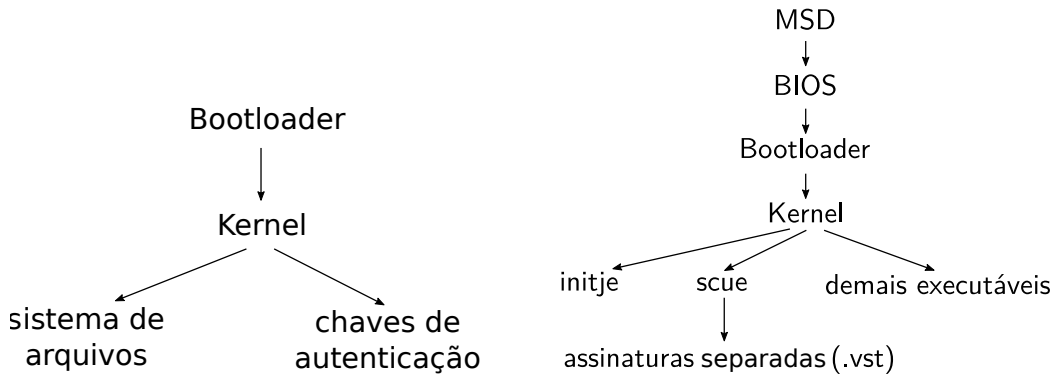
O papel do totalizador consiste em combinar todas as parciais no resultado declarado como oficial, etapa que dura poucas horas. Após 3 dias, versões digitais dos BUs recebidas pelo totalizador são disponibilizadas pelo TSE para comparação com versões em papel impressas no dia da eleição.

2.3. Ecossistema de *software*

A base de código disponibilizada no TPS 2017, marcada com a identificação "*A Hora da Estrela*", possui uma complexidade da ordem de 40 milhões de linhas de código, se incluídos todos os componentes disponíveis para inspeção. O subconjunto correspondente apenas à urna eletrônica possui 24 milhões de linhas, compondo uma distribuição GNU/Linux personalizada (UENUX) para a arquitetura Intel de 32 *bits*. Além das bibliotecas típicas em espaço de usuário, estão presentes o aplicativo oficial de votação (VOTA), o sistema de carga (SCUE), um Recuperador de Dados (RED), o Sistema de Apuração (SA) para inserção de resultados parciais em caso de contingência, e o Gerenciador de Aplicativos (GAP). Os componentes de baixo nível incluem um *bootloader* baseado no Syslinux[1] e um *kernel* Linux, ambos personalizados. O *bootloader* é disparado pela BIOS a partir de um endereço não padronizado e adaptado para decifrar o *kernel*. O *kernel* inclui suporte a dispositivos e mecanismos de segurança personalizados. Na ocasiäo, a base de código estava migrando o *kernel* da versão 2.6 para 3.18, ambas disponíveis para inspeção.

Cifração. A urna eletrônica implementa vários mecanismos com o objetivo de preservar a integridade do *software*. Para isso, combina múltiplas camadas de cifração, visando dificultar a inspeção e a extração de informação sensível por atacantes externos,

[1] *Syslinux bootloader:* `http://www.syslinux.org`



(a) Cifração encadeada em camadas na FC. As setas denotam posse de chaves criptográficas para decifrar o próximo nível.

(b) Cadeia de confiança da urna eletrônica. As assinaturas separadas são armazenadas em arquivos com extensão VST e verificadas pelo SCUE durante a carga.

Figura 2. Cifração e autenticação na urna eletrônica.

tanto da FC quanto na memória interna. Como exemplo de mecanismo de baixo nível, o *bootloader* contém uma chave AES-256 para decifrar a imagem do *kernel* sob o modo de operação ECB durante a inicialização do sistema. O *kernel* possui chaves embutidas para cifrar/decifrar arquivos individuais em um sistema de arquivos MINIX sob o modo AES-XTS; e para implementar um repositório de chaves criptográficas cifradas sob AES-256 em modo CBC. A Figura 2(a) apresenta as várias camadas de cifração, com módulos em níveis superiores sendo responsáveis por decifrar os módulos imediatamente inferiores.

Autenticação. O repositório de chaves mantido pelo *kernel* armazena *chaves de autenticação*, que consistem em chaves privadas para assinar resultados de eleição, chaves públicas e certificados para verificação de assinaturas. BIOS, *bootloader* e *kernel* recebem assinaturas digitais ECDSA instanciadas com a curva NIST P-521. A Figura 2(b) apresenta a cadeia de confiança, que começa no MSD e continua com cada componente autenticando o seguinte até que as aplicações de usuário sejam carregadas e executadas. Módulos de *kernel*, binários executáveis e bibliotecas compartilhadas são assinadas digitalmente com o algoritmo RSA e verificadas com uma chave pública de 4096 *bits* embutida no *kernel*. Além disso, os produtos públicos de eleição e todos os arquivos armazenados na FC e na FV recebem assinaturas digitais separadas, produzidas com o algoritmo Elgamal instanciado com a curva NIST P-256, e armazenadas com sintaxe ASN.1 em arquivos com extensão VST. Desde 2016, os BUs incluem autenticadores calculados com o algoritmo SipHash e um código QR assinado com o algoritmo EdDSA contendo os totais em formato amigável para leitura e comparação com as parciais divulgadas pelo TSE [Aranha et al. 2016].

Geração de números aleatórios. Muitos dos algoritmos criptográficos utilizados para cifração e autenticação necessitam de entropia, e há múltiplos algoritmos espalhados na base de código. As assinaturas Elgamal calculadas pelo MSD utilizam a família `xorshift` de geradores [Marsaglia 2003]. O mecanismo de embaralhamento de votos no RDV é implementado pela combinação de dois outros geradores: coleta de entropia do sistema operacional por meio do arquivo `/dev/urandom`, ou alternadamente de um gerador personalizado cujo algoritmo é uma variante de 32 *bits* da versão de 64 *bits* de um algoritmo obscuro chamado Sapparot-2 [Levin 2005].

3. Trabalhos relacionados

Máquinas de votar do tipo DRE são largamente criticadas na literatura científica, principalmente por suas falhas de projeto e implementação, impossibilidade de auditoria independente de *software* [Rivest 2008] e vulnerabilidade contra ataques internos. Modelos fabricados pela Diebold foram sujeitos a múltiplas análises de segurança com resultados desastrosos [Calandrino et al. 2007] e vulnerabilidades desde a manipulação de resultados eleitorais até a infecção por *malware* de outros equipamentos. A maioria dos problemas parece ter sido resultado de práticas inseguras de desenvolvimento e complexidade do *software* de votação. Outros trabalhos analisaram sistemas comparativamente mais simples de votação e encontraram problemas similares, como os ataques às máquinas holandesas Nedap [Gonggrijp and Hengeveld 2007], e ataques em *hardware* às máquinas EVM utilizadas na Índia [Wolchok et al. 2010]. O registro físico dos votos também não é panaceia: especialistas independentes já descobriram vulnerabilidades em sistemas híbridos com registro eletrônico e impresso, como os utilizados em partes da Argentina [Amato et al. 2015].

No caso do Brasil, os TPS têm sido a melhor oportunidade para especialistas independentes avaliarem a segurança dos mecanismos implementados no sistema de votação. O objetivo dos especialistas é violar as propriedades clássicas de segurança de qualquer sistema de votação, como sigilo do voto e integridade de resultados, e sugerir aprimoramentos para restaurar a segurança dos mecanismos afetados. O formato e escopo dos testes evoluiu com o tempo, e o evento recentemente tornou-se uma etapa oficial no calendário eleitoral. Na primeira edição dos TPS, uma competição organizada em 2009, pesquisadores não tiveram acesso ao código-fonte do sistema. Como resultado, a estratégia vencedora consistiu em quebrar o sigilo do voto pela captura de frequências de rádio emitidas pelo teclado da urna [TSE 2009]. Posteriormente, o TSE alegou ter blindado o teclado contra esse tipo de ataque.

Em 2012, especialistas independentes tiveram a primeira oportunidade de examinar o código-fonte do sistema sem Termo de Sigilo, o que foi suficiente para se descobrir as primeiras vulnerabilidades de *software*. A estratégia vencedora foi a recuperação em ordem de uma quantidade realista de votos coletados por uma urna eletrônica durante uma eleição simulada. O ataque foi baseado apenas em informação pública armazenada no RDV e conhecimento superficial sobre como os votos eram armazenados no arquivo. A principal vulnerabilidade foi uma chamada a `srand(time(0))` no código de embaralhamento de votos, seguida pela impressão do instante de tempo na zerésima. Com o conhecimento da ordem de votação, torna-se possível violar o sigilo do voto de uma seção eleitoral inteira; ou revelar o voto de uma autoridade importante cujo instante de tempo de votação seja conhecido, dado que o arquivo de LOG público armazena os instantes de votação em ordem. Outras falhas de projeto incluem compartilhamento massivo de chaves criptográficas, armazenamento inseguro de material secreto e escolha inadequada de algoritmos. O relatório da equipe vencedora [Aranha et al. 2014] descreve a experiência em detalhes.

Após um hiato em 2014, os TPS tiveram continuidade apenas em 2016, com a introdução de um Termo de Sigilo e extensão do escopo para incluir o subsistema GEDAI-UE e parte do sistema de transmissão. Nessa edição, apresentou-se o primeiro ataque bem-sucedido à integridade de resultados, pela forja de códigos de verificação de BUs validados pelo Sistema de Apuração em caso de contingência [TSE 2016]. Após pressão

substancial da comunidade técnica, o Termo de Sigilo foi relaxado no ano subsequente, permitindo a participação de mais especialistas; e o escopo foi estendido para incluir o *firmware* do MSD. O sistema de identificação biométrica continua fora de escopo.

4. Planejamento

A edição de 2017 dos testes foi composta por cinco fases: registro e aprovação dos participantes, inspeção de código, submissão de planos de teste, execução dos testes, e divulgação dos resultados. As regras para participação foram descritas em uma chamada pública oficial. Durante a fase de registro, entre Agosto e Setembro, pesquisadores individuais e times de até cinco membros submeteram suas informações de registro para seleção pela comissão organizadora.

4.1. Inspeção de código

A inspeção de código foi realizada em computadores com Ubuntu 14.04 providos pelo TSE. Não havia acesso à Internet e não foi permitido entrar ou sair do ambiente de inspeção com material eletrônico. Consequentemente, não foi permitido instalar *software* de preferência dos investigadores, por exemplo um editor de texto poderoso como o `vim`, ou um compilador C. As principais ferramentas utilizadas para buscar vulnerabilidades foram o comando `grep` e o editor de texto `nano`. Rapidamente percebeu-se que faltavam alguns arquivos e conteúdos de variáveis. Surpreendentemente, o TSE restringiu acesso às chaves criptográficas (removendo trechos do código-fonte), apresentando assim uma versão modificada e incompleta da base de código para os participantes.

Várias vulnerabilidades em potencial foram encontradas utilizando `grep`. Uma importante estava relacionada com a cifração do sistema de arquivos da FC, baseada em AES-XTS de 256 *bits* e com a chaves armazenadas no FC. Para cifrar, o AES-XTS utiliza duas chaves, como apresentado na Figura 3. Abaixo cada uma das variáveis observadas durante a inspeção de código são descritas:

- key_1: Primeira parte da chave do AES-XTS, de 256 *bits*, cujo valor estava embutido no *kernel*.
- key_2: Segunda parte da chave do AES-XTS, também de 256 *bits*, computada de acordo com o Algoritmo 1, no qual a função GET_BYTE(n) retorna o n-ésimo *byte* da partição da FC.
- i: Vetor de inicialização. Durante a inspeção de código, observou-se que i correspondia ao número do *inode* do arquivo a ser cifrado/decifrado.
- α: Elemento primitivo do corpo finito $GF(2^{128})$. A versão implementada pelo TSE desvia da especificação e reinicia o valor α^j a cada 256 blocos.
- P_j: O j-ésimo bloco de texto claro. Todos os blocos, exceto possivelmente o último, possuem 128 *bits*.
- C_j: O j-ésimo bloco de conteúdo cifrado, de 128 *bits*.

A decifração usa o mesmo esquema da Figura 3, exceto que o conteúdo cifrado serve como entrada (no lugar de P_j) e o texto claro é obtido como saída (C_j). Apesar do TSE ter removido as chaves criptográficas para a fase inspeção de código, a chave de cifração do sistema de arquivos key_1 permaneceu presente na árvore mais recente (versão 3.18) do *kernel*. Esse fato ocorreu devido a falha operacional da equipe técnica do TSE. No entanto, sabia-se que também era possível obter a chave key_1 através de engenharia reversa do *kernel* após este ser decifrado pelo *bootloader*.

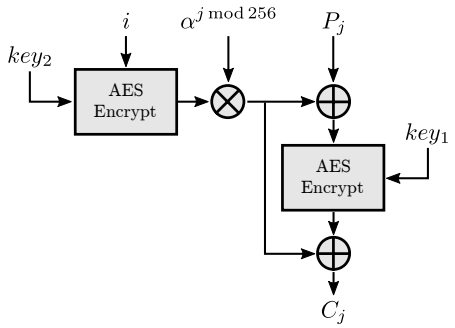


Figura 3. Cifração da FC, baseada no algoritmo AES-XTS.

function GET_KEY
  $key_2 \leftarrow \{\}$
  $\mathit{offset}_1 \leftarrow 512 + 7$
  $\mathit{offset}_2 \leftarrow 512 + 128$
  $b \leftarrow$ GET_BYTE(offset_1)
  for $n \leftarrow 0$ **to** 32 **do**
    $key_2[n] \leftarrow$ GET_BYTE($\mathit{offset}_2 + n$) $\oplus\ b$
  return key_2

Algoritmo 1. Obtenção da chave AES-XTS key_2 da FC.

4.2. Planos de teste

Para avaliar a segurança da urna, foi adotado o paradigma *GQM* (*Goal, Question, Metric*) [Basili 1992], uma forma de definir e avaliar os objetivos através de métricas. *GQM* define um modelo de medição em três níveis: conceitual (Objetivos), operacional (Questões de Pesquisa) e quantitativo (Métricas). Um objetivo especificado com *GQM* possui os seguintes atributos: *propósito, objeto de estudo, foco, parte interessada e contexto*. Um objetivo *GQM* que expressa os interesses durante os TPS é o seguinte: *O propósito do estudo foi avaliar a segurança* do *sistema eletrônico de votação quando este está em uso pelos eleitores no contexto de uma eleição*.

Naturalmente, todos os ataques precisavam encaixar-se no escopo definido para o TPS. A equipe submeteu quatro planos de teste, todos aprovados pelo TSE e listados abaixo na forma de Questões de Pesquisa (QPs).

QP1: É possível extrair chaves criptográficas da FC? Como não se sabia se a chave key_1 encontrada durante a inspeção de código era verdadeira, este plano consistia em realizar engenharia reversa para extrair chaves criptográficas.

QP2: É possível violar o sigilo do voto explorando o gerador de números pseudo-aleatórios? Este plano de teste consistia em verificar se a vulnerabilidade no sigilo do voto descoberta nos TPS 2012 foi de fato corrigida.

QP3: É possível inserir um dispositivo USB malicioso na urna? A urna possui duas entradas USB no terminal do eleitor e duas no terminal do mesário. Este plano de teste consistia em usar dispositivos USB programáveis, como o *Raspberry Pi Zero* e o *FaceDancer*, para tentar personificar o MSD.

QP4: É possível executar código remoto na plataforma *web* de totalização? O acesso à plataforma *web* é permitido apenas através de uma VPN (*Virtual Private Network*). No entanto, as credenciais da VPN provavelmente podem ser extraídas do JE Connect. Explorar uma vulnerabilidade no servidor *web* poderia permitir a falsificação dos resultados de totalização.

Para cada QP, a métrica considerada foi a quantidade de vulnerabilidades exploradas e demonstradas ao TSE. Assim, uma exploração bem-sucedida já seria o suficiente para suportar a referida QP de forma positiva.

5. Execução

De posse da chave criptográfica key_1 capturada do código-fonte e da posição fixa para a chave key_2, um programa para decifrar FCs foi inicialmente implementado em Python nas máquinas de inspeção de código. Como não havia compilador no ambiente de inspeção, o utilitário de linha de comando do OpenSSL foi chamado para decifrar cada bloco individual. O programa foi testado e funcionou corretamente em um protótipo do sistema de arquivos. Essa abordagem terminou ineficiente para arquivos grandes, devido à quantidade de processos disparados, o que motivou a escrita de um novo programa nos computadores de teste, desta vez utilizando o módulo *pycrypto*[2]. Dada a limitação no uso de papel para anotações, a chave criptográfica foi memorizada um *byte* por vez para ser transportada do ambiente de inspeção de código até a bancada de testes.

Decifração da FC. A inspeção do conteúdo decifrado da FC permitiu analisar com cuidado a cadeia de confiança da urna eletrônica. Como descrito na Figura 2(b), o processo tem início na verificação da assinatura da BIOS pelo MSD e termina com a validação de arquivos individuais a partir de assinaturas separadas armazenadas nos arquivos de extensão VST. Durante a inicialização e a instalação do *software*, os arquivos e assinaturas são verificadas. O entendimento da cadeia de confiança motivou a busca de vulnerabilidades adicionais nesse mecanismo. Assim, submeteu-se mais um plano de teste, apresentado a seguir em forma de Questão de Pesquisa:

QP5: É possível executar código arbitrário na urna eletrônica? Se existisse alguma vulnerabilidade na cadeia de confiança, talvez fosse possível violar a integridade do *software* e consequentemente executar código arbitrário. Tal ataque permitiria a um atacante ter total controle sobre a urna (inclusive, para adulterar os votos dos eleitores).

A primeira tentativa da equipe foi a inclusão de arquivos sem assinatura digital na FC, que não provocou interferência no processo de carga. O exame cuidadoso dos arquivos VST revelou que duas bibliotecas compartilhadas utilizadas pelo sistema de votação não possuíam assinaturas. Estas bibliotecas eram utilizadas para manter o registro cronológico de eventos (`libapilog.so`) e gerar chaves criptográficas para o RDV utilizando uma função de derivação de chaves HKDF baseada no algoritmo HMAC (`libhkdf.so`). Ao introduzir código estranho nessas bibliotecas para realizar uma chamada de sistema para imprimir a mensagem "`FRAUDE!`", ficou evidente que a equipe tinha capacidade para execução arbitrária de código, vetor de ataque principal utilizado no restante dos TPS. Uma vez que o tempo para o TPS era bastante restrito, a equipe não investigou as Questões de Pesquisa **QP2**, **QP3** e **QP4**, mas focou nas mais promissoras: **QP1** e **QP5**.

Manipulação do LOG. A capacidade de injetar código arbitrário foi então exercitada de múltiplas formas, a primeira delas pela introdução de código para imprimir a entrada de um teclado USB acoplado à urna. Em seguida, a mensagem `INFO` utilizada para marcar eventos informativos no LOG foi alterada para a cadeia `XXXX` no arquivo `libapilog.so`, ilustrando a possibilidade de modificar-se arbitrariamente o registro cronológico de eventos. Como esperado, uma eleição simulada preservou a alteração e a adulteração foi confirmada pelo exame do arquivo LOG armazenado na MR.

Quebra do sigilo do voto. Em outro ataque, adulterou-se a biblioteca `libhkdf.so` para inserir um retorno antecipado, forçando a chave derivada dinami-

[2]Ferramentas criptográficas em Python: `https://github.com/dlitz/pycrypto`

Figura 4. Reprodução da modificação dinâmica em uma das mensagens contidas no aplicativo VOTA. (a) Interface original. (b) Interface comprometida – o *software* de votação agora faz propaganda para um candidato hipotético.

camente para cifrar o RDV a assumir o valor zero. Como o RDV armazena cada voto depositado na urna eletrônica fora de ordem, o exame do arquivo antes e depois de voto por um eleitor específico revela as escolhas daquele eleitor. Com a biblioteca sempre calculando uma chave zerada, o arquivo RDV pôde ser trivialmente decifrado. Este ataque foi demonstrado no contexto de um mesário malicioso que extrai a FV durante a eleição, mas é evidente que o controle sobre o *software* permite injetar código que armazene os votos em ordem.

Interferência sobre a interface gráfica. Para comprometer o espaço de memória do aplicativo oficial de votação (VOTA), a equipe analisou o binário desmontado, buscando endereços de trechos de código interessantes ou variáveis globais. O binário estava compactado com UPX[3] em um formato fora do padrão, o que consumiu tempo adicional. Uma vez descompactado o binário, percebeu-se que não havia contramedidas para exploração de código, pois o binário não estava em formato independente de posição. Isso simplificou em muito a exploração, pois todos os endereços eram fixos. O código de ataque foi introduzido na biblioteca HKDF.

O ataque prosseguiu com a introdução da chamada de sistema `mprotect` para alterar permissões de páginas somente leitura, o que permitiu alterar a versão do *software* de votação (localizada na seção `.rodata`) para "A Hora da Treta", visível nos arquivos armazenados na MR após uma eleição simulada. Para tornar o ataque ainda mais evidente, a próxima mensagem a ser alterada foi "SEU VOTO PARA", conforme reprodução na Figura 4(b), de forma a instruir os eleitores a votarem no candidato de número 99.

Interferência sobre os votos. A análise do aplicativo VOTA prosseguiu até que o método `AdicionaVoto` foi encontrado. Esse método é chamado apenas quando o voto depositado já foi exibido na tela e confirmado pelo eleitor, de forma que interferir com seu funcionamento não deveria produzir qualquer evidência para o eleitor. Para comprometer o método, um programa em Python foi escrito para infectar o método com o código apresentado abaixo. Esse trecho de código carrega no registrador `eax` uma referência à variável *string* que contém o voto armazenado e carrega no registrador `edi` um ponteiro para os caracteres, finalmente escrevendo um caractere `9` no endereço subsequente.

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

[3]UPX: Ultimate Packer for eXecutables – `https://upx.github.io`

Testar cada nova versão do ataque na urna eletrônica tomava entre 30 e 40 minutos, devido ao procedimento de auto-teste exigido pela carga de uma nova versão do *software*. Por esse motivo, uma versão mais simples do ataque foi testada para abortar de maneira antecipada o método `AdicionaVoto` enquanto a exploração mais sofisticada era testada por simulação. Como o ataque mais simples prevenia que votos fossem armazenados na urna, o *software* de votação comportou-se como esperado e disparou um erro de consistência apontando que a cédula em memória estava vazia – nenhum voto digitado pelo eleitor foi armazenado. Os testes foram então interrompidos pontualmente às 18h do último dia, antes que o ataque mais sofisticado pudesse ser testado no equipamento real.

6. Discussão

No último dia dos TPS, um grupo de peritos criminais da Polícia Federal foi capaz de fazer engenharia reversa e recuperar a imagem do *kernel* às claras após fixar o *bootloader* em um endereço padrão, quando carregado em uma máquina virtual. Inspecionando a memória, eles obtiveram a mesma chave criptográfica do AES-XTS que havia sido encontrada durante a inspeção de código. Desta forma, há suporte à Questão de Pesquisa **QP1**: **Foi possível extrair chaves criptográficas da FC**. Esse fato confirma que os ataques subsequentes não possuíam premissas, e que o acesso ao código-fonte era desnecessário.

A conclusão é que a integridade do *software* e dos resultados depende do sigilo da chave simétrica armazenada no *bootloader*. Além de ser trivial para o time de desenvolvimento acessá-la, essa chave também pode ser obtida por um atacante externo, uma vez que é armazenada em texto claro nos cartões de instalação. O fato é que, este mecanismo sequer pode ser considerado cifração, mas sim uma forma bem mais fraca de ofuscação. Essas vulnerabilidades não são algo novo: o relatório dos TPS 2012 [Aranha et al. 2014] já havia indicado a fragilidade das chaves armazenadas de forma insegura no *software* de votação. Naquela versão, a mesma chave e vetor de inicialização (IV) eram compartilhados por todas as máquinas de votação para cifrar arquivos utilizando AES-256 no modo CBC. A única melhoria observada em 2017 foi a adoção de IVs variáveis e do modo XTS.

Após decifrar as FCs, constatou-se que duas bibliotecas compartilhadas não possuíam assinaturas digitais. Dessa forma, eram vulneráveis à injeção de código arbitrário, o que forneceu capacidades desproporcionais a um atacante. Assim, também há suporte à Questão de Pesquisa **QP5**: **Foi possível executar código arbitrário na urna eletrônica**. Como apresentado anteriormente, foram várias as demonstrações de que os ataques obtinham total controle sobre o *software* da urna eletrônica. As duas bibliotecas compartilhadas não tiveram suas assinaturas separadas calculadas devido à lista de arquivos para verificação não ser gerada por um processo automatizado. Em seu relatório, a equipe de desenvolvimento do TSE afirma que, além das assinaturas separadas, as bibliotecas continham assinaturas RSA internas para conferência pelo *kernel*. Apesar disso, um erro no código de verificação (e em seu teste unitário), preveniram que a manipulação das bibliotecas fosse detectada [TSE 2018].

O relatório dos TPS 2012 já havia defendido a tese de que o RDV não serve a qualquer propósito de segurança e apenas fragiliza o sigilo do voto. Qualquer ataque bem sucedido ao *software* de votação pode também comprometer a integridade do RDV [Aranha et al. 2014]. A prevenção de ataques ao RDV em si depende da implementação de um gerador de números aleatórios seguro. Apesar de melhorias significa-

tivas terem sido implementadas em relação ao gerador observado em 2012, as modificações podem não ser suficientes. O gerador `xorshift` e o algoritmo Sapparot-2 não são adequados para aplicações criptográficas, fato explicitado pelos seus próprios autores.

Contramedidas. A equipe técnica do TSE publicou recentemente um relatório detalhando as contramedidas implementadas em resposta às vulnerabilidades encontradas pelos investigadores [TSE 2018], das quais vale destacar o armazenamento de chaves criptográficas e fortalecimento dos mecanismos de verificação de integridade.

A equipe do TSE afirmou que as chaves de cifração não serão mais definidas no código-fonte nas próximas versões do *software*. Essas chaves serão calculadas em tempo de execução com ajuda da BIOS, aumentando a complexidade da ofuscação [TSE 2018]. Essa decisão de implementação foi justificada pelo fato de nem todas as urnas terem o dispositivo MSD disponível, o que limita a possibilidade do seu uso para armazenamento das chaves. Devido à restrição de que qualquer máquina deve ser capaz de substituir qualquer outra no dia da eleição, o TSE considera um risco a existência de duas versões distintas do *software* em operação. Isso sugere que a segurança do sistema é ditada pelo modelo mais antigo em operação, neste caso a urna de 2007 sem a presença do MSD.

Recomendações. As chaves de cifração utilizadas na FC (bem como outras chaves criptográficas) deveriam ser segregadas à menor unidade possível (seção eleitoral, local de votação ou cidade), para reduzir o impacto em caso de vazamento. No médio prazo, recomenda-se projetar uma arquitetura de distribuição de chaves para mover todas as chaves criptográficas para dentro do perímetro de segurança do MSD.

Os procedimentos de verificação de integridade podem ser aprimorados pela adoção de boas práticas, ainda que o recurso seja de eficácia intrinsecamente limitada, por exemplo com a automatização de processos críticos e implementação de testes, inclusive para os casos negativos (exercitando a falha dos mecanismos de proteção). A lista de arquivos que devem ser assinados não deveria ser gerada manualmente e a verificação de assinaturas deve ser testada em cenários diferentes do ideal (chave e mensagens corretas).

Para incremento no sigilo do voto, recomenda-se: (i) remover do RDV campos não utilizados por eleitores faltantes para prevenir uma busca exaustiva nas sementes possíveis; (ii) adotar algoritmos mais fortes e padronizados ou ler do `/dev/urandom` diretamente, caso a entropia coletada tenha qualidade suficiente. A longo prazo, recomenda-se a eliminação do arquivo RDV, com alteração na lei que obriga sua existência.

Alegações oficiais invalidadas. Não há documento formalizando o modelo adversarial ou propriedades de segurança consideradas pelo TSE durante o projeto e desenvolvimento do sistema eletrônico de votação. Isso complica análises de segurança, dado que o adversário torna-se um alvo móvel, mudando convenientemente dependendo do ataque. Felizmente, o TSE publicou um documento com respostas a perguntas frequentes que defende alguns dos mecanismos de segurança e estabelece algumas propriedades esperadas [TSE 2014]. Os resultados aqui descritos contradizem várias dessas alegações:

- Na página 10, "*Não é possível executar aplicativos não autorizados na urna eletrônica. Da mesma forma, também não é possível modificar nenhum aplicativo da urna.*" Os ataques foram capazes de incluir e modificar duas bibliotecas compartilhadas na FC, cuja execução posterior violou a integridade do *software*.
- Entre as páginas 12 e 14, "*A urna eletrônica não é vulnerável a ataques externos.*

(...) Isso é garantido pelos diversos mecanismos de segurança, baseados em assinatura digital e criptografia, que criam uma cadeia de confiança entre hardware *e* software *e impedem qualquer violação da urna eletrônica.*" Máquinas DRE são sabidamente inseguras contra ataques internos com controle sobre o *software*, mas foi demonstrado como um atacante externo com acesso à FC também pode manipular o *software* antes da eleição. Como cada FC instala até 50 urnas, há um fator de amplificação que reduz obstáculos e o custo de um ataque em larga escala.

- Na página 22, "*O arquivo de log é mais um mecanismo de transparência e auditoria disponibilizado pela Justiça Eleitoral.*" O arquivo de LOG está sujeito à adulteração caso o *software* esteja sob controle do atacante, logo o LOG não serve como ferramenta de auditoria do *software* de votação.
- Na página 33, "*(...) afirmar que a partir da posse da chave do sistema de arquivos é possível gerar mídias 'de diferente teor' é incorreto. (...) Na verdade, todos os arquivos que requerem integridade e autenticidade são assinados digitalmente.*" Como demonstrado, de posse da chave de cifração da mídia, foi possível alterar duas bibliotecas compartilhadas importadas por outros programas. Esses arquivos modificados foram reinseridos na FC gerando um novo cartão de instalação válido.

O TSE alega que a cifração do sistema de arquivos é uma das várias barreiras de segurança contra ataques externos. Esta seria, portanto, projetada para servir como primeiro obstáculo contra adversários mal informados, ao passo que outros mecanismos criptográficos proveriam defesas contra adversários mais sofisticados. Embora a primeira alegação seja tecnicamente válida, uma vez que a cifração do sistema de arquivos não passa de uma ofuscação, observou-se que capturar as chaves de cifração forneceu poder desproporcional ao adversário, que se torna capaz de escolher ou revelar outras chaves mais importantes. Isso aconteceu porque a decifração permitiu inspecionar o conteúdo da FC e detectar vulnerabilidades graves nos mecanismos de verificação de integridade, consequentemente violando as principais propriedades de segurança do sistema.

7. Conclusões

Os TPS oferecem uma oportunidade importante para se colaborar com o incremento de segurança do sistema eletrônico de votação, mas algumas alterações poderiam torná-los mais efetivos: minimizar a burocracia e intervenções durante os testes; aumentar agilidade dos procedimentos internos de autorização; ampliar o escopo para incluir identificação biométrica e partes da infraestrutura de transmissão e totalização; ampliar a duração do evento; tornar o código-fonte amplamente disponível. Em particular, vale observar que o tempo consumido para montar os ataques aqui descritos não é representativo para uma tentativa real de fraude, em que as condições de trabalho são muito diferentes.

O *software* de votação da urna eletrônica brasileira ainda não satisfaz requisitos mínimos de segurança e transparência e está muito aquém da maturidade esperada de um sistema crítico em produção há mais de 20 anos. Recomenda-se ao TSE revisar cuidadosamente suas práticas de desenvolvimento e considerar novamente a adoção do voto impresso para fornecer garantias fortes do funcionamento correto do sistema no dia da eleições, acompanhando a experiência de outros países. Espera-se que os resultados aqui descritos contribuam com o debate no país a respeito da introdução de um registro físico de votos individuais como uma forma de aprimorar segurança e transparência do sistema de votação.

Referências

Amato, F., Oro, I. A. B., Chaparro, E., Lerner, S. D., Ortega, A., Rizzo, J., Russ, F., Smaldone, J., and Waisman, N. (2015). Vot.Ar: una mala elección. https://ivan.barreraoro.com.ar/vot-ar-una-mala-eleccion/.

Aranha, D. F., Karam, M. M., Miranda, A., and Scarel, F. (2014). *Software vulnerabilities in the Brazilian voting machine*, pages 149–175. IGI Global.

Aranha, D. F., Ribeiro, H., and Paraense, A. L. O. (2016). Crowdsourced integrity verification of election results - An experience from Brazilian elections. *Annales des Télécommunications*, 71(7-8):287–297.

Basili, V. R. (1992). Software modeling and measurement: The goal/question/metric paradigm. Technical report, University of Maryland at College Park, MD, USA.

Calandrino, J. A., Feldman, A. J., J. A. Halderman, D. W., and H. Yu, W. P. Z. (2007). Source Code Review of the Diebold Voting System. Available at https://jhalderm.com/pub/papers/diebold-ttbr07.pdf.

Gonggrijp, R. and Hengeveld, W. (2007). Studying the Nedap/Groenendaal ES3B voting computer: A computer security perspective. In *EVT*. USENIX Association.

Levin, I. O. (2005). Sapparot-2: Fast Pseudo-Random Number Generator. http://www.literatecode.com/sapparot2.

Marsaglia, G. (2003). Xorshift RNGs. *Journal of Statistical Software, Articles*, 8(14):1–6.

Rivest, R. L. (2008). On the notion of 'software independence' in voting systems. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 366(1881):3759–3767.

TSE (2009). Acompanhamento da execução do Plano de Teste. https://web.archive.org/web/20160107163923/http://www.tse.gov.br/internet/eleicoes/arquivos/Teste_Sergio_Freitas.pdf.

TSE (2014). Sistema Eletrônico de Votação. Perguntas Mais Frequentes. 2ª Edição. http://www.justicaeleitoral.jus.br/arquivos/tse-perguntas-mais-frequentes-sistema-eletronico-de-votacao.

TSE (2016). Teste Público de Segurança do Sistema Eletrônico de Votação: Compêndio. http://www.tse.jus.br/hotsites/catalogo-publicacoes/pdf/teste-publico-de-seguranca-2016-compendio.pdf.

TSE (2018). Respostas às vulnerabilidades e sugestões de melhorias encontradas no teste público de segurança 2017. http://www.justicaeleitoral.jus.br/arquivos/relatorio-tecnico-tps-2017-1527192798117.

van de Graaf, J. and Custódio, R. F. (2002). Tecnologia Eleitoral e a Urna Eletronica – Relatório SBC 2002. http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view.download&catid=77&cid=107.

Wolchok, S., Wustrow, E., Halderman, J. A., Prasad, H. K., Kankipati, A., Sakhamuri, S. K., Yagati, V., and Gonggrijp, R. (2010). Security analysis of India's electronic voting machines. In *ACM Conference on Computer and Communications Security*, pages 1–14. ACM.