Week 2 - Group 3

1

Os Java Cards podem ser uma tecnologia adequada para um sistema de identificação em bibliotecas porque oferecem segurança, portabilidade, independência de hardware e baixo custo de produção.

Segurança

Os Java Cards possuem mecanismos criptográficos que garantem que os dados armazenados, como informações pessoais e histórico de empréstimos, estejam protegidos contra acessos não autorizados. Além disso, permitem autenticação segura, garantindo que apenas o utilizador legítimo possa aceder aos seus dados.

Portabilidade

Por serem cartões de pequenas dimensões, semelhantes a cartões bancários ou de identificação, os Java Cards são fáceis de transportar, permitindo que os utilizadores os tenham sempre consigo. Além disso, podem ser utilizados em sistemas de leitura contactless ou com chip.

Independência de Hardware

Os Java Cards são compatíveis com diferentes dispositivos e leitores, uma vez que seguem um padrão universal baseado em Java. Isto significa que podem ser utilizados em qualquer biblioteca equipada com um sistema de leitura compatível, sem necessidade de hardware específico.

Baixo Custo

Devido à sua independência de hardware, os Java Cards tornam-se uma alternativa de baixo custo porque não exigem infraestruturas complexas para funcionar. Como dito anteriormente, podem ser utilizados em diversos dispositivos sem necessidade de hardware exclusivo, reduzindo os custos de implementação e manutenção.

Outro fator que contribui para o baixo custo é a possibilidade de atualização do software sem necessidade de substituição do hardware. Isto significa que novas funcionalidades podem ser adicionadas ao sistema da biblioteca sem custos adicionais com novos cartões ou dispositivos físicos.

2

O Trusted Platform Module (TPM) e o Hardware Security Module (HSM) são dispositivos de segurança baseados em hardware que oferecem proteção para operações criptográficas e armazenamento seguro de chaves, mas diferem em propósito, implementação e uso. Ambos garantem a integridade dos dados e a

resistência contra ataques, armazenando chaves de forma segura e realizando funções como criptografia, assinatura digital e autenticação.

No entanto, o TPM é um chip embutido em computadores, servidores e dispositivos IoT, projetado para garantir a integridade do sistema, protegendo o processo de inicialização, armazenando credenciais e gerenciando chaves de criptografia locais.

Já o HSM é um dispositivo externo, como um cartão PCIe, módulo USB ou appliance de rede, utilizado principalmente em ambientes empresariais e na nuvem para proteger chaves criptográficas, garantir segurança em transações financeiras e atender a requisitos de conformidade como FIPS 140-2.

Assim, o TPM é ideal para segurança de dispositivos individuais, enquanto o HSM é essencial para proteção de dados críticos e gestão de chaves em ambientes empresariais e na nuvem.

3

Por que Intel SGX ou ARM TrustZone são mais adequados do que Hardware Security Modules (HSMs)?

O hospital precisa armazenar e processar dados clínicos sensíveis, o que exige segurança tanto na execução quanto no armazenamento dessas informações.

Intel SGX e ARM TrustZone permitem o processamento seguro de dados dentro de regiões isoladas da memória, protegendo as informações até mesmo contra ataques internos. Como os dados do hospital precisam ser acessados e processados frequentemente pelos funcionários, essas tecnologias são ideais, pois possibilitam a computação segura diretamente no processador, sem necessidade de transferências externas para zonas menos seguras que poderiam comprometer a segurança.

Por outro lado, os HSMs são projetados principalmente para operações criptográficas e armazenamento seguro de chaves, mas não são otimizados para processamento frequente de dados clínicos. Além disso, os HSMs são dispositivos externos, o que pode introduzir latência e dificultar o fluxo de trabalho hospitalar devido à necessidade de acesso contínuo a um hardware dedicado.

Outra vantagem do SGX e do TrustZone em relação aos HSMs é que essas tecnologias já vêm embutidas nos processadores, tornando a solução mais escalável e econômica. O uso de HSMs exigiria a aquisição de hardware adicional, o que poderia aumentar os custos e a complexidade da implementação.

Qual escolher? Intel SGX ou ARM TrustZone?

A escolha entre Intel SGX e ARM TrustZone depende de vários fatores, incluindo compatibilidade, segurança e facilidade de implementação.

Em termos de compatibilidade, o Intel SGX é mais adequado para servidores e computadores x86, que é a arquitetura mais usada, enquanto o ARM TrustZone é

mais indicado para dispositivos móveis, tablets e equipamentos IoT. Considerando que a infraestrutura hospitalar provavelmente utiliza servidores e desktops, o Intel SGX tende a ser a melhor escolha.

Em relação à segurança, o SGX oferece um nível de isolamento mais rigoroso, garantindo que nem mesmo o sistema operacional possa acessar os dados sensíveis armazenados nos enclaves. O TrustZone, por outro lado, cria dois ambientes distintos, Seguro e Não Seguro, mas não fornece um isolamento tão refinado quanto o SGX. Para um hospital, onde a proteção de dados clínicos é essencial, o Intel SGX pode ser a opção mais segura.

No que diz respeito à implementação, o SGX possui ampla documentação, ferramentas de suporte e simuladores, tornando o desenvolvimento mais acessível. Já o ARM TrustZone pode ser mais complexo de trabalhar, pois exige familiaridade com a arquitetura ARM, que pode ser menos comum em ambientes hospitalares.

Concluindo, dado esse contexto, onde a maioria dos sistemas hospitalares opera em servidores baseados em x86 e há a necessidade de um isolamento mais forte dos dados clínicos, o Intel SGX é provavelmente a melhor escolha. No entanto, caso haja necessidade de suporte para dispositivos móveis e IoT, o ARM TrustZone pode ser considerado como uma alternativa viável para essas plataformas específicas.

4

A execução especulativa é uma técnica usada em processadores modernos para melhorar o desempenho ao executar instruções antes de saber se elas realmente serão necessárias. Em vez de esperar que uma condição seja resolvida (como um if-statement em um código), o processador faz uma previsão do caminho mais provável e começa a executar as instruções de forma especulativa.

Se a previsão estiver correta, a execução continua sem interrupções; se estiver errada, os resultados especulativos são descartados e o processador corrige o caminho executando a instrução correta. Essa técnica é amplamente usada para otimizar o tempo de processamento.

Execução Eager

Na execução eager o processador executa todas as possíveis ramificações de um if-statement simultaneamente, sem aguardar a resolução da condição. Assim que a condição é avaliada, os caminhos desnecessários são descartados, e o programa segue apenas com o resultado correto.

Essa abordagem elimina penalidades causadas por predições erradas, pois o caminho correto já está pronto para ser usado. No entanto, ela apresenta um alto desperdício de recursos, pois o processador executa instruções que podem nunca ser utilizadas.

Vantagem: Evita penalidades de predição errada.

Desvantagem: Gasta muitos recursos computacionais e energia ao executar instruções desnecessárias.

Execução Preditiva

Na execução preditiva, o processador não executa todos os caminhos como na execução eager. Em vez disso, ele prevê qual caminho é mais provável e executa apenas esse caminho antecipadamente.

Se a previsão estiver correta, a execução ocorre de forma eficiente, sem desperdício de recursos. No entanto, se a previsão estiver errada, o processador descarta os resultados incorretos e executa o caminho correto, o que gera uma penalidade de desempenho devido à necessidade de reexecução.

Vantagem: Mais eficiente que a execução eager, pois usa menos recursos.

Desvantagem: Se a previsão for errada, ocorre uma penalidade de desempenho, pois o processador precisa descartar a execução errada e refazer a correta.

5

a)

O ataque Lucky13 afeta sistemas que utilizam implementações do TLS (Transport Layer Security) com o modo CBC (Cipher Block Chaining). Como o problema decorre do próprio funcionamento do protocolo, qualquer implementação de TLS que utilize CBC pode estar vulnerável caso não tenha sido devidamente corrigida.

Sistemas que ainda utilizam TLS 1.1 ou TLS 1.2 sem as atualizações de segurança apropriadas continuam suscetíveis ao ataque. No entanto, o TLS 1.3 elimina essa vulnerabilidade, pois removeu completamente o suporte a CBC, adotando apenas cifradores baseados em AEAD (Authenticated Encryption with Associated Data), como ChaCha20-Poly1305 e AES-GCM.

As implementações vulneráveis podem ser: Servidores web; VPNs; bibliotecas criptográficas populares, como OpenSSL; Smart Cards;

b)

O Lucky13 é um ataque de cronometragem (timing attack) que explora diferenças mínimas no tempo de processamento de mensagens encriptadas em TLS com CBC.

A vulnerabilidade surge devido ao uso do esquema MAC-then-Encrypt (MtE) no TLS com CBC, no qual o código de autenticação da mensagem (MAC) é calculado antes da encriptação. Durante a desencriptação, o servidor segue os seguintes passos:

1. Decifra a mensagem usando o modo CBC.

2. Verifica e remove o padding adicionado para alinhar o tamanho do bloco.
3. Calcula e verifica o MAC para garantir a integridade da mensagem.

O problema ocorre porque diferentes tipos de padding e variações na verificação do MAC resultam em tempos de processamento ligeiramente diferentes. Estas variações permitem que um atacante envie mensagens manipuladas e meça o tempo de resposta do servidor, inferindo informações sobre o conteúdo encriptado.

c)

O ataque Lucky13 é especialmente perigoso para dispositivos de hardware criptográfico, como smart cards ou Hardware Security Modules (HSMs), por três razões principais:

1. **Tempos de resposta mais previsíveis:**

   Dispositivos de hardware, como smart cards, possuem execução altamente determinística, ou seja, realizam operações criptográficas com pouca variação no tempo de resposta. Isso facilita ataques baseados em cronometragem, pois o atacante pode medir diferenças temporais de forma precisa e inferir informações sobre a mensagem cifrada.

2. **Dificuldade em aplicar atualizações de segurança:**

   Diferente de sistemas baseados em software (como servidores web), dispositivos de hardware muitas vezes não podem ser facilmente atualizados para corrigir vulnerabilidades. Muitos smart cards e HSMs antigos ainda utilizam implementações vulneráveis de TLS, o que os torna alvos fáceis para ataques temporais.

3. **Acesso direto ao hardware pelo atacante:**

   Em muitos cenários, o atacante pode ter acesso físico ao dispositivo vulnerável. Por exemplo, um terminal de leitura de smart card pode ser manipulado para enviar requisições controladas e medir diretamente os tempos de resposta sem interferências de rede. Isso torna ataques como o Lucky13 ainda mais eficazes, já que elimina variáveis imprevisíveis presentes em servidores ou redes.