Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aprimorando a Detecção de Vulnerabilidades em APIs Criptográficas Java: Uma Abordagem Qualitativa Integrando CogniCrypt, CryptoGuard e LibScout

Guilherme Andreúce S. Monteiro

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Rodrigo Bonifacio de Almeida

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aprimorando a Detecção de Vulnerabilidades em APIs Criptográficas Java: Uma Abordagem Qualitativa Integrando CogniCrypt, CryptoGuard e LibScout

Guilherme Andreúce S. Monteiro

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Rodrigo Bonifacio de Almeida (Orientador)
CIC/UnB

Prof. Dr. Donald Knuth
Stanford University

Dr. Leslie Lamport
Microsoft Research

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 19 de setembro de 2023

Dedicatória

Eu dedico este trabalho a minha esposa que me apoiou e incentivou durante todo o processo de desenvolvimento deste trabalho. Dedico também aos meus pais, que sempre me apoiaram e me incentivaram a estudar mesmo sem entender muito bem o que eu estava fazendo. Também aos meus amigos que me ajudaram a manter a sanidade durante o processo de desenvolvimento deste trabalho.

Agradecimentos

Agradeço ao Prof. Dr. Rodrigo Bonifacio de Almeida pela persistencia e paciência em me orientar durante o desenvolvimento deste trabalho. Agradeço também em especial ao Luis Amaral por não só ter me ajudado com tudo o que foi necessário como também por ter me incentivado a continuar quando eu estava prestes a desistir.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este estudo apresenta uma abordagem inovadora para aprimorar a detecção de vulnerabilidades em APIs criptográficas Java, visando fortalecer a segurança de aplicações baseadas nessa tecnologia. Para isso, integramos as ferramentas CogniCrypt e CryptoGuard com o LibScout, permitindo a identificação precisa da origem de warnings relacionados a bibliotecas externas. Essa abordagem qualitativa representa um avanço significativo na promoção da segurança em aplicações Java, contribuindo para um ecossistema digital mais resiliente e protegido contra potenciais ameaças cibernéticas. Ao incorporar a identificação da origem dos warnings, também possibilitamos sugestões diretas aos desenvolvedores das bibliotecas, otimizando o processo de correção de vulnerabilidades. No entanto, enfrentamos desafios ao analisar código obfuscado e ao utilizar clusters e datasets no LibScout, evidenciando a necessidade de aprimoramentos nessa ferramenta. A integração proposta neste trabalho representa um passo significativo em direção à segurança abrangente de dados sensíveis e sistemas críticos em aplicações Java.

Palavras-chave: CogniCrypt, Eclipse, Segurança do código, Análise

Abstract

This study introduces an innovative approach to enhance the detection of vulnerabilities in Java cryptographic APIs, aiming to strengthen the security of applications built on this technology. By integrating the tools CogniCrypt and CryptoGuard with LibScout, we enable the precise identification of the source of warnings related to external libraries. This qualitative approach represents a significant advancement in promoting security in Java applications, contributing to a more resilient digital ecosystem protected against potential cyber threats. The incorporation of warning source identification also allows for direct suggestions to library developers, streamlining the vulnerability correction process. However, we encountered challenges when analyzing obfuscated code and utilizing clusters and datasets in LibScout, highlighting the need for improvements in this tool. The integration proposed in this work represents a significant step towards comprehensive security for sensitive data and critical systems in Java applications.

Keywords: CogniCrypt, Eclipse, Code Security, Analisys

Sumário

1 Introdução **1**

1.1 Introdução 1

Referências **3**

Apêndice **3**

A Fichamento de Artigo Científico **4**

Anexo **8**

I Documentação Original `UnB-CIC` (parcial) **8**

Capítulo 1

Introdução

1.1 Introdução

A criptografia, enquanto disciplina fundamental no campo da segurança da informação, desempenha um papel preponderante na salvaguarda de sistemas digitais e na proteção de dados sensíveis contra ameaças cibernéticas. Com a crescente complexidade das aplicações e a diversificação de bibliotecas e frameworks disponíveis, torna-se imperativo o desenvolvimento de ferramentas automatizadas capazes de identificar potenciais falhas e vulnerabilidades nas interfaces de programação de aplicações (APIs) criptográficas.

O advento da linguagem CrySL representou um notável avanço nesse panorama, viabilizando a definição precisa de regras para o uso seguro de APIs criptográficas em código Java. Esta linguagem conferiu a capacidade de estabelecer padrões mais rigorosos para a implementação de práticas seguras de criptografia. Entretanto, questionamentos relativos à eficácia das ferramentas existentes e à precisão de seus alertas têm sido objeto de considerável escrutínio.

Neste contexto, o presente estudo empreende uma análise qualitativa abrangente da detecção de vulnerabilidades em APIs criptográficas, valendo-se das ferramentas CogniCrypt e CryptoGuard. Inicialmente, submetemos issues e gists em projetos de código aberto, visando investigar a percepção dos desenvolvedores quanto aos alertas gerados por essas ferramentas. A análise preliminar revelou que, em grande parte das ocorrências, os alertas indicavam possíveis falhas em bibliotecas externas, o que ressalta a necessidade premente de um mecanismo capaz de distinguir tais origens.

Durante a condução deste estudo, deparamo-nos com a complexidade inerente à análise de código obfuscado, o que ressaltou a importância de considerar uma gama diversificada de contextos de implementação ao avaliar a eficácia das ferramentas de detecção de vulnerabilidades.

Adicionalmente, observou-se que as ferramentas CogniCrypt e CryptoGuard, embora inestimáveis em sua capacidade de identificar potenciais vulnerabilidades, carecem da aptidão para discernir a origem do alerta, seja ela proveniente de bibliotecas nativas ou externas. Tal limitação poderia potencialmente acarretar em falsos positivos ou negligenciar alertas de importância vital advindos de bibliotecas de fundamento.

Para superar esse desafio, lançamos mão do estudo intitulado "Automated Third-Party Library Detection for Android Applications: Are We There Yet?". A partir dessa fonte, propomos uma solução inovadora ao integrar o resultado do LibScout ao contexto do CryptoGuard e CogniCrypt. Esta abordagem possibilitou não apenas a detecção de potenciais vulnerabilidades, mas também a identificação precisa de correspondências associadas a bibliotecas externas. Desse modo, concebeu-se uma flag adicional, denominada "external_library", destinada a sinalizar a presença de uma biblioteca externa quando uma correspondência era identificada.

Todavia, é imperativo mencionar as dificuldades enfrentadas na utilização do LibScout. A ferramenta, por vezes, demonstrou limitações ao definir clusters com diferentes níveis de granularidade, resultando na omissão de bibliotecas conhecidas como não-nativas nos resultados. Ademais, o conjunto de dados mais recente disponível para a utilização data de julho de 2019, o que pode potencialmente impactar a abrangência das correspondências identificadas.

Esta adição não apenas aprimorou a precisão na identificação de vulnerabilidades, mas também abriu novas possibilidades. Agora, somos capazes de prover sugestões diretamente aos desenvolvedores das bibliotecas em questão, permitindo uma intervenção mais direta e eficaz na resolução de potenciais vulnerabilidades, em contraposição à necessidade de os desenvolvedores que as utilizam se encarregarem dessa tarefa.

Este trabalho representa um passo significativo na promoção da segurança em aplicações baseadas em Java, visando salvaguardar de maneira mais abrangente os dados sensíveis e sistemas críticos contra potenciais ameaças cibernéticas. Por meio dessa abordagem qualitativa e da integração de ferramentas de detecção, busca-se fortalecer as práticas de segurança na implementação de APIs criptográficas, contribuindo para um ecossistema digital mais resiliente e protegido.

Referências

Apêndice A

Fichamento de Artigo Científico

UnB Departamento de Ciência da Computação 2017/1

Fichamento de Artigo Científico

Prof. Guilherme N. Ramos

Um fichamento reúne elementos relevantes do conteúdo, apresentando a estrutura do texto, e deve seguir a seqüência do pensamento do autor, destacando suas ideias, argumentos, justificativas, exemplos, fatos, etc.

1 Artigo Científico

Geralmente, um *artigo científico* é escrito com a seguinte estrutura (buscando responder algumas questões):

I. Introdução

- Qual o contexto do problema? (O que? Onde? Quando?)
- Qual a principal questão ou problema colocado? (Por quê? Como? Qual?)
- Qual o objetivo visado? O que se pretende constatar ou demonstrar? (investigar, analisar, refletir, contribuir,...)

II. Referencial Teórico

- Quais são os autores/teorias/conceitos que já estudaram os principais assuntos abordados e que sustentam ao texto?
- Quais os resultados mais recentes relacionados a eles?

III. Metodologia/Desenvolvimento

- Quais os procedimentos metodológicos adotados? (natureza do trabalho: empírico, teórico, histórico) – (coleta de dados: questionário, entrevista, levantamento bibliográfico).
- Como a pesquisa foi desenvolvida? Quais as principais relações entre teoria e prática?
- Havendo artefato proposto, ele está disponível para utilização e/ou modificação?

IV. Resultados

- Houve validação (por meio de experimentação)? Como foi feita?
- Os resultados obtidos são corretos/válidos?

V. Conclusões

- Qual o problema atacado?
- Quais os resultados obtidos para os objetivos propostos?
- Quais conclusões podem ser tiradas destes resultados?
- Quais as limitações da metodologia utilizada?
- Quais as possibilidades de trabalhos futuros para o problema?

2 Fichamento

Neste contexto, um fichamento deve conter a seguinte estrutura:

1. ***Identificação do aluno:*** indicação precisa de quem é o autor do fichamento.
2. ***Identificação do texto:*** indicação precisa de quem são os autores do texto analisado e dos detalhes do documento, de modo que se possa buscá-lo para uma leitura completa.
3. ***Pontos-chave:*** noções mais relevantes do texto analisado. *Proposta* (o que é apresentado?), *mérito* (por que é relevante?), *validação* (como verificar a utilidade?), e *perspectivas* (o que pode ser melhorado?).
4. ***Palavras-chave:*** expressões que identificam o assunto abordado.
5. ***Sinopse do texto:*** resumo *com suas palavras*. Deve ser mais detalhado que um *abstract*, geralmente apresentando pelo menos um parágrafo por seção do texto original. No caso de inclusão de trechos, o texto deve ser identificado entre "aspas" e concatenado através de suas próprias palavras.
6. ***Análise crítica:*** posicionar-se em relação as seguintes questões: pertinência do assunto; forma como foi abordado; comparação com outras abordagens do mesmo assunto (caso conheça). Junto ao *resumo*, é a parte mais interessante para o leitor, pois apresenta uma avaliação do conteúdo apresentado.

2.1 Exemplo

1. ***Identificação do aluno:***Alan Mathison Turing, 00/000000
2. ***Identificação do texto:***Guilherme N. Ramos, Yutaka Hatakeyama, Fangyan Dong, and Kaoru Hirota, Hyperbox clustering with Ant Colony Optimization (HACO) method and its application to medical risk profile recognition, Applied Soft Computing, Vol. 9, Issue 2, pp 632-640, 2009. (doi:10.1016/j.asoc.2008.09.004)
3. ***Pontos-chave:***

   Proposta: HACO - método para aglomeração de dados utilizando hipercaixas com posicionamento otimizado via algoritmo de colônia de formigas.

   Mérito: apresenta uma nova forma de fazer agrupamentos considerando a topologia do espaço de dados e fornecendo resultados intuitivos e facilmente utilizáveis.

   Validação: comparação com algoritmos conhecidos em testes com dados padrões e com dados de infecção viral para diagnóstico auxiliado por computador.

   Perspectivas: adequação das dimensões das hipercaixas, diminuição de parâmetros.

4. ***Palavras-chave:***colônia de formigas, hipercaixa, otimização, reconhecimento de padrões.

5. ***Sinopse do texto:*** A *Colônica de Formigas* (ACO) é um método de otimização que pode ser utilizado para agrupar dados. *Hyperbox clustering with Ant Colony Optimization* (HACO) é um método de agrupamento que utiliza ACO tentar posicionar hipercaixas no espaço de forma a agrupar a maior quantidade de dados possível, e ainda gera uma forma simples de classificar novos dados.

   ACO é baseado no comportamento de formigas reais, que otimizam o caminho percorrido entre o alimento e o formigueiro. Hipercaixas definem de forma muito simples uma região em um espaço n-dimensional, combinadas para definir regiões de topologia complexa, e utilizadas como um classificador de forma trivial.

   HACO busca encontrar uma partição de dados, efetivamente definindo grupos. Primeiro, aplica ACO para tentar posicionar hipercaixas de forma que estas contenham a maior quantidade possível de dados. A seguir, se não há conhecimento prévio da quantidade de classes, considera-se que as hipercaixas que se sobrepõem representam uma mesma classe de dados, e [grupos de] hipercaixas distintas representam classes diferentes. Caso o número de classes seja conhecido, HACO aplica o algoritmo *Nearest-neighbor* (NN) para definir a quantidade correta de grupos. Uma consequência de se usar hipercaixas é que o resultado do agrupamento define também um classificador: se um novo dado está dentro de uma hipercaixa, sua classe será a mesma da definida por esta hipercaixa.

   Os resultados experimentais de HACO fora, comparados a três algoritmos que têm o mesmo fim: testado em NN, *Fuzzy C-Means* (FCM), e o próprio ACO (com uma abordagem diferente para agrupamento). O primeiro teste foi em conjuntos de dados sintéticos, e serviu como prova de conceito, oferecendo diversas informações sobre o comportamento do método em função de certas configurações. Um segundo experimento foi realizado com dados reais de pacientes para agrupá-los em "saudáveis" e "não saudáveis", e HACO obteve o melhor resultado dentre os algoritmos testados. A análise da estrutura do classificador gerado possibilita descobrir informações relativas às características das classes, indicando um "perfil de risco" para os pacientes.

   Foi apresentado o método HACO para agrupar dados, utilizando a meta-heurística ACO e hipercaixas, que possibilita a extração de informações inerentes a estrutura dos dados. HACO foi validado com experimentos, e demonstrou grande potencial. Os resultados são muito influenciados pela configuração dos parâmetros, que será investigada.

6. ***Análise crítica:*** ~~Este é o melhor artigo de todos os tempos.~~ O artigo apresenta uma forma inovadora de agrupar dados, de forma não-supervisionada (embora possa aproveitar informações se houver). O resultado pode ainda ser utilizado como classificador de novos dados, e - o mais interessante - analisado para descobrir informações sobre as classes. Além disso, explora as vantagens de cada elemento que compõe o método, obtendo melhores resultados e diminuindo o custo computacional. A aplicação em um caso real, cujos resultados podem ser utilizados para auxiliar o diagnóstico de pacientes, dá mais destaque ao trabalho.

   O problema de agrupamento de dados é muito pertinente e, em tempos de excesso de dados, a possibilidade de análise intuitiva da estrutura e descoberta de conhecimento é bastante interessante. Além disso, a solução proposta é de uso geral, oferecendo mais possibilidades de uso.

   Os experimentos realizados foram coerentes e suficientes para demonstrar o que foi afirmado. Entretanto, o método só foi comparado a outros algoritmos simples, seria interessante uma comparação com algoritmos mais avançados, bem como específicos para aplicação. A comparação também foi em uma única aplicação específica, seria melhor que houvesse mais testes com outros dados para conclusões melhor embasadas. Além disso, é preciso uma análise mais profunda quanto às configurações de HACO, que influenciam muito o resultado.

Anexo I

Documentação Original UnB-CIC (parcial)

```
% -*- mode: LaTeX; coding: utf-8; -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% File    : unb-cic.cls (LaTeX2e class file)
%% Authors : Flávio Maico Vaz da Costa
%%               (based on previous versions by José Carlos L. Ralha)
%% Version : 0.96
%% Updates : 0.5  [??/11/2004] - Initial release. don't remember the day.
%%         : 0.75 [04/04/2005] - Fixed font problems, UnB logo
%%                               resolution, keywords and palavras-chave
%%                               hyphenation and generation problems,
%%                               and a few other problems.
%%         : 0.8  [08/01/2006] - Corrigido o problema causado por
%%                               bancas com quatro membros. O quarto
%%                               membro agora é OPCIONAL.
%%                               Foi criado um novo comando chamado
%%                               bibliografia. Esse comando tem dois
%%                               argumentos onde o primeiro especifica
%%                               o nome do arquivo de referencias
%%                               bibliograficas e o segundo argumento
%%                               especifica o formato. Como efeito
%%                               colateral, as referências aparecem no
%%                               sumário.
%%         : 0.9 [02/03/2008]  - Reformulação total, com nova estrutura
%%                               de opções, comandos e ambientes, adequação
%%                               do logo da UnB às normas da universidade,
%%                               inúmeras melhorias tipográficas,
```

```
%%                              aprimoramento da integração com hyperref,
%%                              melhor tratamento de erros nos comandos,
%%                              documentação e limpeza do código da classe.
%%          : 0.91 [10/05/2008] - Suporte ao XeLaTeX, aprimorado suporte para
%%                              glossaries.sty, novos comandos \capa, \CDU
%%                              e \subtitle, ajustes de margem para opções
%%                              hyperref/impressao.
%%          : 0.92 [26/05/2008] - Melhora do ambiente {definition}, suporte
%%                              a hypcap, novos comandos \fontelogo e
%%                              \slashedzero, suporte [10pt, 11pt, 12pt].
%%                              Corrigido bug de seções de apêndice quando
%%                              usando \hypersetup{bookmarksnumbered=true}.
%%          : 0.93 [09/06/2008] - Correção na contagem de páginas, valores
%%                              load e config para opção hyperref, comandos
%%                              \ifhyperref e \SetTableFigures, melhor
%%                              formatação do quadrado CIP.
%%          : 0.94 [17/04/2014] - Inclusão da opção mpca.
%%          : 0.95 [06/06/2014] - Remoção da opção "mpca", inclusão das opções
%%                              "doutorado", "ppginf", e "ppca" para identificar
%%                              o programa de pós-graduação. Troca do teste
%%                              @mestrado por @posgraduacao.
%%          : 0.96 [24/06/2014] - Ajuste do nome do curso/nome do programa.
%%
```