UNIVERSIDADE SAGRADO CORAÇÃO

PAULO OTÁVIO SHINKAI BONINI

UM ESTUDO DOS MEIOS DE ATAQUE E PREVENÇÃO EM APLICAÇÕES WEB VIA SQL INJECTION

BAURU
2015

PAULO OTÁVIO SHINKAI BONINI

UM ESTUDO DOS MEIOS DE ATAQUE E PREVENÇÃO EM APLICAÇÕES WEB VIA SQL INJECTION

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas da Universidade do Sagrado Coração como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

BAURU
2015

Bonini, Paulo Otavio Shinkai

B7157e

Um estudo dos meios de ataque e prevenção em aplicações web via SQL Injection / Paulo Otavio Shinkai Bonini. -- 2015.

81f.: il.

Orientador: Prof. Dr. Elvio Gilberto da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. SQL Injection. 2. Aplicações web. 3. Ataques. I. Silva, Elvio Gilberto da. II. Título.

PAULO OTÁVIO SHINKAI BONINI

UM ESTUDO DOS MEIOS DE ATAQUE E PREVENÇÃO EM APLICAÇÕES WEB VIA SQL INJECTION

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas da Universidade do Sagrado Coração como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

Banca Examinadora:

_____
Prof. Dr. Elvio Gilberto da Silva.
Universidade do Sagrado Coração

_____
Prof. Esp. Alex Setolin Beirigo
Universidade do Sagrado Coração

_____
Prof. Me. Henrique Pachioni Martins
Universidade do Sagrado Coração

Bauru, 02 de dezembro de 2015.

RESUMO

A Informação tornou-se um ativo valioso e estratégico. A “era da informação”, caracterizada pela inclusão digital e disseminação da Internet, disponibilizou grandes volumes de informação. Não tardou para que a aparição de ameaças contra a confidencialidade, integridade e disponibilidade das informações gerassem desconfianças quanto ao modo como tais são armazenadas, transmitidas e processadas. Neste contexto esse trabalho apresenta uma técnica chamada “SQL Injection” que é utilizada para atacar bancos de dados através de aplicações voltadas para a Internet. Para isso propõe-se a implementação de ataques em aplicações web alvo utilizando as técnicas SQL Injection com a utilização de ferramentas como o SQLMap e o Havij, contribuindo, informando desenvolvedores e os usúarios de um dos meios de ataque que circulam pela internet.

Palavras-chave: SQL Injection, aplicações web, ataques.

ABSTRACT

The information became valuable and strategic. The “information age” characterized by the digital inclusion and spread of Internet provided large volumes of information. It didn’t take long to the apparition of threats against the confidentiality, integrity and availability of information generate distrust as to the manner these things are stored, transmitted and processed. In this context the work presents a technique called “SQL Injection” that is used to attack databases through applications focused on the Internet. For this proposes the attacks implementation on target web applications using the SQL Injection techniques using tools like SQLMap and the Havij contributing, informing developers and users of the one of the ways of attack that are circulating on the internet.

Keywords: SQL Injection, web applications, attacks.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A internet, desde seu início até os dias de hoje, obteve uma taxa de crescimento considerável. Uma organização voltada para a análise do tráfego da rede mundial de computadores constatou que, atualmente, aproximadamente 40% da população do planeta, possui conexão com a Internet. Uma porcentagem expressiva, levando em consideração que no ano de 1995, esse número era inferior a 1%. (INTERNET..., 2015).

A rede atualmente conta com inúmeros sistemas, serviços e informações, o que consequentemente eleva a porcentagem de falhas de seguranças contidas nesta. Diante das informações expostas, este meio de comunicação, que da mesma forma é utilizado para diversas outras atividades, requer certo nível de segurança.

De modo que, são as aplicações web as ferramentas responsáveis pela função de interação dos usuários a internet, acabam elas igualmente a rede, se tornando visadas e vulneráveis as quebras de segurança, por meios de ataques, invasões, entre outros, que em sua maioria buscam explorar os pontos vulneráveis da aplicação.

De acordo com Nakamura (2007), as informações devem se apresentar ao usuário de forma íntegra e confiável, para isso a segurança tem de estar ligada diretamente a manutenção das informações fornecidas aos cliente, mas o que ocorre, é o fato de uma grande maioria pessoas não compreenderem o conceito de segurança, deste modo, os devidos cuidados podem não ser tomados, o que pode ocasionar em imensas perdas, tanto financeiras como morais.

Com o uso da Internet por instituições coorporativas, governamentais e educacionais, a preocupação com a segurança das informações tem aumentado. Qualquer organização engajada em atividades que usem a rede mundial de computadores deve avaliar a segurança computacional associada a estas atividades. Os serviços com maior quantidade de ataques são os de publicidade, comércio eletrônico, informações confidenciais e acesso à rede. Devido a esta grande demanda de serviços ofertados, se faz necessário o estudo e divulgação de formas de se prevenir contra esses ataques.

Com base neste contexto, este projeto tem como proposta informar, e ao mesmo tempo, demonstrar à todos os usários e demais interessados, por meio da simulação de um ataque a aplicações Web utilizando o modo de invasão SQL

Injection, as possíveis causas que podem levar a este tipo de ataque, bem como, formas de prevenção.

1.2 OBJETIVOS

Apresenta-se abaixo o objetivo geral e os objetivos específicos da pesquisa.

1.2.1 Objetivo Geral

Estudar e testar os tipos de ataque à aplicações Web via SQL Injection.

1.2.2 Objetivos Específicos

a) efetuar um levantamento bibliográfico sobre tipos de ataque e formas de prevenção;
b) estudar sobre SQL Injection;
c) compreender como funciona o ataque à aplicações Web via SQL Injection;
d) simular uma invasão em uma aplicação Web via SQL Injection;
e) demonstrar as principais vulnerabilidades, bem como, apresentar de forma descritiva as práticas de segurança;
f) demonstrar o impacto e consequências causada pelo SQL Injection;
g) analisar as invasões estabelecendo um comparativo.

2 BANCOS DE DADOS

Antes da existência dos Bancos de Dados, por volta dos anos 60, os dados eram mantidos eventualmente em arquivos, de forma interativa a aplicação. Tal modo de armazenamento e recuperação de informações somente foi modificado após a chegada dos primeiros Sistemas Gerenciadores de Bancos de Dados comerciais. (BOSCARIOLI, 2006).

Como todos sabem nos dias de hoje, a tecnologia se tornou fundamental para todas as empresas, independente de seu porte, no qual os bancos de dados tem um

papel de grande importância, com a função de base para o armazenamento de informações. (GILLENSON, 2009).

Tratar sobre os banco de dados pode ser um assunto um pouco complexo, devido ao fato de existirem diversos conceitos referentes a esse ambiente. Segundo Gillenson (2009), para compreender totalmente os bancos de dados precisa-se entender algumas conceitos, como Repositório de dados, Dicionário de dados, Acesso a dados, Abstração de dados, entre outros.

Os BDs foram criados inicialmente para suprir a necessidade de informação de uma determinada organização, conjuntos ordenados deliberadamente e visando a lógica, estruturas inter-relacionadas, que utilizam o compartilhamento de dados como ferramenta, são características e definições que compõem, além de descrever de uma maneira simplista um BD. (GILLENSON, 2009).

Já de acordo com Damas (2007), os bancos de dados são basicamente uma coleção de dados estruturados, que de forma organizada, são armazenados de um modo pertinente por uma aplicação informatizada.

2.1 A LINGUAGEM SQL

2.1.1 História

De acordo com Celso, no começo da era dos computadores, sentiu-se a necessidade de armazenar dados de entrada processados pelo próprio, logo, o armazenamento e recuperação das informação tornam-se essencial na informática. (OLIVEIRA, 2006).

Edgar F. Codd pesquisador da IBM, na califórnia, em junho de 1970, publicou um trabalho, sobre o modelo relacional e banco de dados, que serviu como alicerce para o desenvolvimento da linguagem SEQUEL, conhecida atualmente como SQL. (OLIVEIRA, 2006).

Seu desenvolvimento foi continuado pela IBM, e sua primeira versão comercial somente foi lançada em 1979, deste modo, em 1986 e 1987, decorrente ao sucesso da linguagem, as entidades ANSI e ISO, veem o dever de padronizá-la, mantendo praticamente o mesmo modelo SQL original. (OLIVEIRA , 2006).

Não demorou muito, e dois anos depois, uma nova versão é emita, com expressivas modificações contudo sua sucessora chega em 1992, com uma versão

que define regras básica para os bancos relacionais, melhorando sua anterior. Sucedida em 99, com a entrada do SQL3 ou SQL-99, que possui foco no modelo de dados objeto-relacional, surgindo com novos tipos de dados, Boolean e LOB, assim como novos predicados e semânticas. (OLIVEIRA, 2006; LIMA, 2015?).

Já em 2003 com a vinda do SQL:2003 ou também conhecido como SQL:200n, novos dados como BIGINT similar ao smallint, MULTISET semelhante ao Array, instrução MERGE, aparecem caracterizando essa versão. [LIMA, 2015?].

Em 2006 e 2008, duas versões nascem, definindo de quais formas o SQL pode ser utilizado em união ao XML, delimitando modos de armazenar e importar informações XML em um banco de dados SQL. Além disso são definidos aos aplicativos de linguagem de programação convencional, estruturas e funções da linguagem SQL que podem ser chamadas pelos próprios. (ISO/IEC 9075-14, 2006; ISO/IEC 9075-3, 2008).

Atualmente a linguagem mais recente é denominada SQL:2011, lançada em dezembro 2011, tem como características, a exclusão com o MERGE, melhorias nas chamadas de funções, Pipelined DML, aprimoramento do comando SELECT para permitir a alteração de dados a partir do próprio, entre outras. (ZEMKE, 2012).

2.1.2 Conceito

Uma linguagem de computador utilizada em bancos de dados, composta pela DDL, um componente da linguagem que integra a definição dos dados, usada basicamente em declarações e exclusões, e pela DML, a manipulação de informações, utilizada em comandos de recuperação, como o SELECT, que será mostrado posteriormente. (GILLENSON, 2009).

Padronizada internacionalmente pelas instituições ISO e ANSI, possui fácil entendimento, faz uso de uma determinada sintaxe, um conjunto de regras baseados no inglês e na sintaxe do VBA, manipulando dados, realizando uma relação entre os mesmos, além, criar e alterar a aparência de objetos do banco de dados, como tabelas, a partir um componente do SQL, a DLL. (MICROSOFT, 2007).

De acordo com Taylor (2003), o Structured Query Language, é uma linguagem que pode ser utilizada de diversos modos, pelo modo de ser muito flexível, além de distinta das outras linguagens como C, COBOL, Java, Fortran, Pascal e Basic, por não ser procedural, de modo que pode-se processar conjuntos

de registros, ao invés de um por vez, possibilitando ao usuário manipular tipos complexos de dados.

Uma curiosidade sobre esta linguagem não ser procedural, é que de fato uma grande parte dos programadores, por estarem habituados as soluções procedural, fez-se com que em 2003, fosse lançado uma versão com funcionalidades procedural, como blocos BEGIN, instruções IF, funções e procedimentos. (TAYLOR, 2003).

Para o entendimento de um Ataque com a utilização do SQL-Injection, é preciso ter noções básicas sobre os comandos que podem ser utilizados na linguagem SQL, perante essa informação, foram selecionados e ordenados, algumas cláusulas e comandos, listados conforme mostrado na Figura 1.

Figura 1 - Comandos SQL.

CREATE	Comando usado para criar uma tabela.
ALTER	Comando usado para modificar uma coluna ou uma tabela existente.
DROP	Comando usado para excluir uma coluna ou tabela existente.
SELECT	Comando usado para recuperar dados de uma tabela.
UPDATE	Comando usado para modificar os dados de uma tabela.
INSERT	Comando usado para inserir novas linhas em uma tabela.
DELETE	Comando usado para excluir linhas de uma tabela.
FROM	Comando lista tabelas que possuem campos listados pelo SELECT.
WHERE	Restringe a lista exibida de acordo com os determinados critérios.
ORDER BY	Comando especifica como classificar os resultados.
UNION	Operador uni várias consultas semelhantes, retornando-as em uma.
LIKE	Operador para restringir lista, exibindo informações de acordo com os dados preenchidos.
HAVING	Comando parecido com o WHERE, especifica quais registro serão selecionados, após usar GROUP BY.
GROUP BY	Combina registros com informações idênticas.

Fonte: Elaborada pelo autor.

2.2 APLICAÇÕES WEB

A Internet, é um utensílio utilizado por todos os públicos, desde organizações privadas e governamentais até usuários privados, além de possuir diversas funções. Uma ferramenta usada como meio de comunicação, usada para comprar e vender praticamente tudo, entre outros, trata-se de uma rede global que uni milhões de computadores ao redor do planeta, no qual são compartilhados qualquer tipo de dados, imagens, documentos etc. (QIAN, 2010).

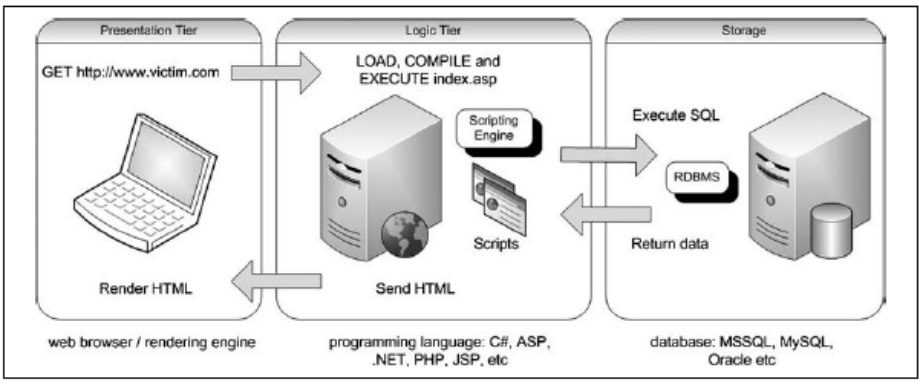
Seu desenvolvimento inicial foi feito pela agência ARPA em torno da década de 60, inicialmente chamado de ARPANET, seu principal objetivo, embora que a maioria das pessoas pensem que esta rede foi essencialmente criada para fins militares, seu maior intuito foi visando o compartilhamento das informações através dos computadores entre todos os centros da instituição de forma segura, reduzindo custos. (GATTO, 2009).

Tal rede passou por várias alterações durante os anos, até que em meados de 1989, Tim Berners-Lee, inventou a chamada World Wide Web, um sistema que revolucionou a internet, trabalhando em conjunto a própria. Em paralelo a este concebeu a linguagem HTML, utilizada para criar os documentos na Web, que podem ser localizados pela URL. [NUNES, 2015?; W3C, 2014?].

Para visualização dos documentos contidos na web, em 1990, Tim criou o primeiro navegador, WorldWideWeb. [W3C, 2015?]. O navegador é utilizado para acessar a URL, unindo um cliente e um servidor, enviando um pedido de solicitação ao servidor, retornando sua resposta para o cliente, de forma que estas ações conceituam uma aplicação Web. Para esta comunicação entre ambos existir, são seguidas especificação de um protocolo, chamado HTTP. (CLARKE, 2009).

Existem diversos métodos utilizados pelo protocolo HTTP, mas de forma que o objetivo é o Ataque SQL-Injection, carecemos somente dos métodos, GET, usado comumente em Links, transmite a informação na URL ao servidor, e o POST, muito utilizado em formulários, como de cadastros, encaminhando informação ao servidor de acordo com a URL de destino. (CLARKE, 2009). Conforme ilustra Figura 2.

Figura 2 - Cliente Servidor, Web e Banco de Dados.



Fonte: Clarke (2009).

2.3 LINGUAGEM HTML

HyperText Markup Language ou linguagem de marcação para hipertexto, o famoso HTML, como dito anteriormente foi inventado por Tim Berners-Lee, e desenvolvido com o propósito de criar e realizar a manutenção de documentos na web, nas chamadas páginas. De acordo com Silva (2011, p. 20), a definição exata desta linguagem, se resume em, "[…] todo o conteúdo inserido em um documento para a web e que tem como principal característica a possibilidade de se interligar a outros documentos da web […]".

Segundo Neves (2004), o HTML, trata-se de uma linguagem de formatação de texto, que permite o desenvolvimento de blocos de texto, no qual possuem marcas (tags) que indicam ao navegador a forma de como expor as informações contidas no documento.

Figura 3 - Cabeçalho e Corpo de um Documento HTML.

```
<html>
        <head>
                <title>Título do documento HTML</title>
        </head>

        <body>Conteúdo do documento HTML</body>

</html>
```

Fonte: Neves (2004).

Conforme a ilustra a Figura 3 pode-se observar um exemplo básico da estrutura da linguagem HTML, de forma que a primeira tag <html>, abrange todo

documento, no qual estão o <head> que normalmente é reservado para Título, e o <body> que remete a praticamente todo conteúdo.

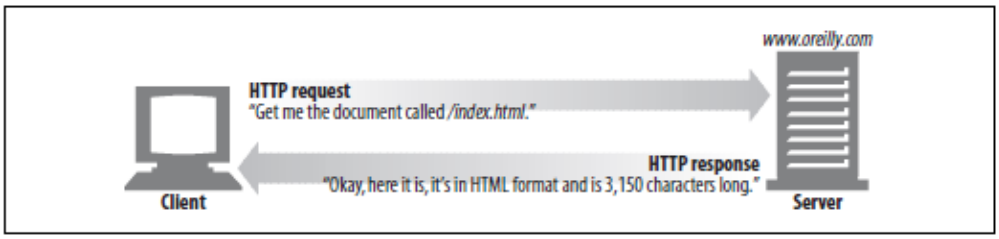
2.4 PROTOCOLO HTTP

O Hypertext Transfer Protocol é um protocolo que possui diversas utilidades e aplicações, mas atualmente seu uso está mais empregado para atender a realização da comunicação da Word Wide Web, entre os navegadores web e os servidores web. (GOURLEY, 2002).

Considerado o protocolo mais famoso de comunicação entre navegadores e servidores web, o HTTP de modo a visar a transmissão de dados de forma segura, utiliza protocolos confiáveis, o que garante que seus dados não serão alterados ou corrompidos em transição, além de garantir a integridade das informações ao usuário. (GOURLEY, 2002).

Os servidores Web, muitas vezes chamados de servidores HTTP, são recursos da Web que armazenam as informações da própria, de maneira que estas são solicitadas pelo usuário através dos navegadores, assim o servidor por sua vez retorna ao cliente os dados ou objetos com suas respectivas informações em HTTP, conforme ilustrado na Figura 4. (GOURLEY, 2002).

Figura 4 - Cliente solicitação ao Servidor Web.



Fonte: Gourley (2002).

2.4.1 Métodos

O protocolo HTTP possui diversos métodos, estes são utilizados para enviar diferentes tipos de solicitação, de forma que para cada mensagem de pedido HTTP, existi um método que informa ao servidor web qual tipo de ação será realizada. (GOURLEY, 2002).

2.4.1.1 Método GET

De acordo com Kennedy (2002) ambos os métodos, GET e POST, podem ser acionados por formulários na linguagem HTML. O método GET diferentemente do POST, se conecta ao servidor para processar os dados do formulário e enviá-los, de modo que são adicionados a URL, separando-os por interrogação (?). A Figura 5 é um exemplo, no qual é possível identificar o ponto de interrogação (?), que marca o fim do endereço e o início dos valores que serão passados ao servidor.

Figura 5 - Método GET

```
http://www.meusite.com.br/versecao.php?nome=Joao+Silva&celular=1223365
```

Fonte: Elaborada pelo autor.

O método GET, é utilizado para melhorar o desempenho normalmente quando os formulários possuem campos curtos, e para casos, no qual haja necessidade de chamar o servidor fora do domínio, pelo fato de ser possível passar parâmetros adicionados a URL. (KENNEDY, 2002).

2.4.1.2 Método POST

Ao referenciar segurança, como a maioria das ferramentas da web, o método POST possui suas falhas, mas pelo fato de poder utilizar a criptografia, de forma que quando a segurança for visada, este se torna mais aconselhável diante do método anterior. (KENNEDY, 2002).

As informações nesse método são passadas separadamente a conexão feita ao servidor, são enviadas isoladas do cabeçalho da solicitação entre navegador e o servidor web, diferindo-se do método GET. [USP, 2015?].

2.5 SEGURANÇA

Para o professor Filho (2004, p. 1), a segurança da informação "[…] compreende um conjunto de medidas que visam proteger e preservar informações e sistemas de informações […]".

Segundo Litchfield (2005), para um uma melhor segurança em sistemas, deve-se primeiramente entender como funcionam seus ataques, para assim analisar os método de prevenção adequados.

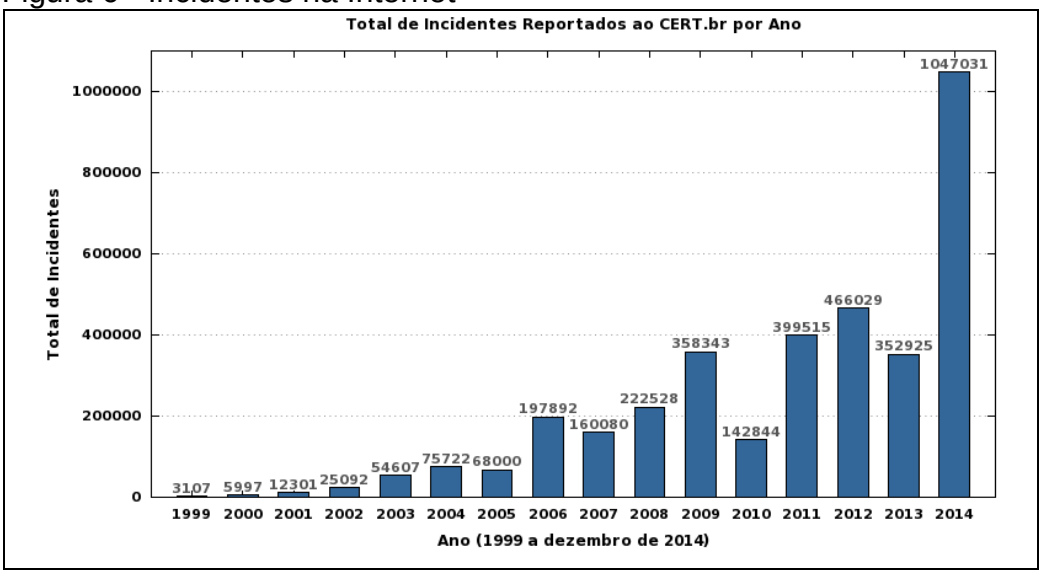
Para o compreendimento do tema segurança da informação de acordo com Iepsen [2015?], existem três aspectos a serem levados em consideração, a confidencialidade, que está ligada diretamente as permissões do sistema em autorizar e não autorizar determinados usuários a acessar os dados, a integridade, aspecto que garante a autenticidade dos mesmos, não permitindo que sejam alterados sem permissão, e a disponibilidade, que está relacionado a quantidade de vezes em que o sistema realizou as ações solicitadas, sem erros ou falhas em relação ao número de vezes que o sistema foi encaminhado a realizar as determinadas ações.

A quebra de segurança dos sistemas, pode ser relacionada ao fato da existência de vulnerabilidades no próprios. Atacantes exploram estas, de forma a comprometer todo o sistema. (SILVA, 2012).

Empresas e organizações devem sempre estar atualizadas, para suas devidas proteções, afim de utilizar a segurança da informação de maneira estratégica, com intuito de garantir a integridades de seus dados, além de inconsequentemente aumentar a produtividade ao possuir um ambiente mais organizado. (SANTO, 2010).

No Brasil como em qualquer outro país, no que se refere segurança da informação, é alvo de diversos tipos de ataques, invasões, entre outros. Segundo o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (2015), um grupo destinado a Resposta a Incidentes de Segurança para a Internet brasileira, faz divulgação dos índices encontrados no país, que no decorrer dos anos sempre apresentaram determinadas variações, o qual acompanharam o desenvolvimento da Internet, mas conforme demonstra a Figura 6, verifica-se uma considerável e preocupante taxa de crescimento destes no ano de 2014.

Figura 6 - Incidentes na Internet



Fonte: CERT (2015).

2.6 ATAQUES

Atualmente existem diversas técnicas de ataques relacionados aos computadores, no quais de acordo com Bezerra [2015?], são crime virtuais, que todos usuários estão expostos. Neste trabalho será visado uma técnica de ataque voltada para a utilização das aplicações web, denominada SQL Injection. Este ataque possui como base a utilização de uma técnica chamada Code Injection, diante disso será abordado de maneira simplista outras duas forma que utilizam do mesmo recurso, afim de auxiliar na compreensão da técnica.

2.6.1 Code Injection

Code Injection ou Código de Injeção são os tipos de ataque que utilizam da injeção de um código em uma determinada aplicação, no qual normalmente é explorado as falhas de segurança, referentes ao manuseio dos dados, como a validação dos dados de entrada e saída. (OSWAP, 2013a).

De acordo com Athanasopoulos (2009), o ataque é feito através de uma determinada injeção de código realizada pelo lado do cliente em alguma aplicação web, ou seja, pode-se realizar a invasão introduzindo um comando malicioso em algum formulário comum, em um determinado site, afim de mudar seu curso de execução.

Como este tipo de ataque é tradicionalmente considerado uma da principais ameaças, logo, existem várias técnicas de ataque que utilizam como base este método de injeção, algumas delas seriam, SQL Injection, Cross-site Scripting (XSS), LDAP injection, entre outros.

2.6.2 Cross-Site Scripting (XSS)

O Cross-Site Scripting é uma categoria de ataque do tipo injeção, no qual são injetados na aplicação web scripts maliciosos afim de obter informações, cookies, ou até mesmo remodelar as informações contidas na página HTML. O método de ataque XSS, em sua maioria sempre aproveitam vulnerabilidades generalizadas e, podem ser divididos em 3 diferentes vertentes. (OSWAP, 2013b).

A primeira vertente, Stored XSS, ocorre basicamente quando o atacante consegue armazenar informações no servidor da aplicação web, a segunda é Reflected XSS, é definida quando após a tentativa de injeção, imediatamente uma mensagem de qualquer tipo é retornada, e por último o tipo DOM Based XSS, acontece quando é enviado ao usuário uma URL maliciosa. (OSWAP, 2013b).

2.6.3 LDAP Injection

O LDAP Injection ou Lightweight Directory Access Protocol Injection, como nome já referencia, é um tipo de ataque que faz uso como base do Code Injection. Um ataque que visa aplicações web que possuem declarações LDAP, afim de modificá-las, através de um proxy local, de forma a permitir a implementação de códigos maliciosos. As técnicas de exploração avançadas do SQL-Injection também podem ser aplicadas nesse modelo. (OSWAP, 2015).

2.6.4 SQL Injection

Segundo Clarke (2009), este método de ataque que é um dos mais devastadores no que se refere negócios, pelo simples fato de poder acessar praticamente todo seu banco de dados, provavelmente deve existir desde a primeira conexão entre as aplicações Web e BDs da linguagem SQL, mas somente meados de 1998 um artigo chamado “NT Web Technology Vulnerabilities”, escrito por Rain

Forest Puppy, abordou essa técnica revelando-a para o mundo. Rain um pouco depois em 2000, divulgou outro trabalho, entitulado "How I hacked PacketStorm", no qual alertava de que maneira o SQL Injection era usado em websites. Com base em suas descobertas, pesquisadores ainda continuam trabalhando, desenvolvendo e aprimorando técnicas a partir desta.

De acordo com Iepsen [2015?], muitas das aplicações web disponíveis possuem como modo de armazenamento os bancos de dados, que em sua maioria fazem uso da linguagem SQL para manutenção, inclusão, alteração, consultas, entre outros. Diante desse fato, determinados usuários com certo conhecimento na área, utilizam das peculiaridades da linguagem e vulnerabilidades do sistema, para atacar a aplicação alvo, com o uso da técnica denominada SQL Injection, no qual ocorrido com sucesso, danos irreparáveis podem ocorrer.

O SQL Injection é considerado uma vulnerabilidade, no qual se da ao usuário malicioso a oportunidade de influenciar sobre determinado banco de dados. Um tipo de ataque em que se faz a injeção de código SQL em uma aplicação web, através dos parâmetros de entrada do usuário, que são enviados ao servidor de bancos de dados SQL, para análise e execução. (CLARKE, 2009).

A principal forma para utilização da injeção do SQL de acordo com Clarke (2009) é através do inserção direta, conhecida como a manipulação de dados ou comandos, esta inseri diretamente o código malicioso em um campo da aplicação que está ligado ao armazenamento no banco de dados, de maneira a ser executado imediatamente. (SILVA, 2012).

Esta técnica exploratória maliciosa de falhas que utiliza o Code Injection, do mesmo modo que pode injetar um código para ser executado imediatamente, pode implementar um comando em alguma área apropriada do aplicativo, para ser executado posteriormente em um momento oportuno por outra atividade.

Para um melhor entendimento do ataque, na Figura 7, é possível observar uma consulta que será feita em um banco de dados SQL ligado a uma aplicação web, que está a espera de uma resposta do servidor web para que o comando seja executado.

Figura 7 - Consulta esperando valor.

```
SELECT * FROM PRODUTOS WHERE
PRECO<="&Request.QueryString("Preco");
```

Fonte: Elaborada pelo autor.

Na URL exibida na Figura 8, é apresentado um caminho, no qual o servidor web tem a função de localizá-lo, e executar em seu banco de dados o determinado comando SQL, visto na Figura 7, diante dos parâmetros passados.

Figura 8 - URL normal.

```
http://www.meusite.com.br/produtos.php?val=200
```

Fonte: Elaborada pelo autor.

Neste caso, produtos, que é uma tabela no BD, receberá em seu campo preço, o valor 200, retornando todo conteúdo de sua tabela que possuem preços menores ou iguais a 200.

Figura 9 - URL alterada.

```
http://www.meusite.com.br/produtos.php?val=200’ OR ‘1’ = ‘1
```

Fonte: Elaborada pelo autor.

Já na Figura 9, pode-se perceber que a URL foi alterada, no qual foram acrescentados alguns caracteres. Esta URL, possui o mesmo intuito, retornar os produtos do banco de dados, mas devido a adição dos códigos, ela retornará todos os produtos da tabela independente de seus preços. Ao adicionar o comando OR, a consulta foi alterada como pode ser visto na Figura 10, de forma que preços sejam menores ou igual a 200, ou 1 igual a 1, que nada mais é que uma verdade, logo todo registro que se encontrar, será uma verdade, retornando sempre todos.

Figura 10 - Consulta alterada.

```
SELECT * FROM PRODUTOS WHERE PRECO <=  '200' OR '1' = '1' ;
```

Fonte: Elaborada pelo autor.

Existem diversos outros modos do atacante obter informações do sistema, um deles é através do comando UNION, mostrado na Figura 11.

Figura 11 - Consulta nomes e senhas.

```
http://www.meusite.com.br/produtos.php?val=200;UNION SELECT
NOME, SENHA FROM USUARIOS
```

Fonte: Elaborada pelo autor.

Além do usuário malicioso, possuir o poder de deletar os dados já armazenados nas tabelas, exibido na Figura 12.

Figura 12 - Deleta tabela produtos.

```
http://www.meusite.com.br/produtos.php?val=200;DROP TABLE PRODUTOS
```

Fonte: Elaborada pelo autor.

Nestes simples exemplos consegue-se notar como o atacante pode manipular a seu favor a instrução SQL, explorando a vulnerabilidade do sistema. Segundo Clarke (2009), mesmo com a grande quantidade de maneiras existentes para a exploração das vulnerabilidades para a utilização da injeção, o sucesso do ataque é diretamente ligado as falhas dos sistemas e a habilidades do atacante.

2.6.4.1 SQL Injection baseado em erros

O SQL Injection baseado em erros normalmente é utilizado por usuários com um nível de conhecimento maior e quando não se possui nenhuma informação a respeito da aplicação. Segundo Halfond (2006), este ataque é considerado o antecessor do ataque principal, pois tem como objetivo identificar informações relevantes.

Esta técnica consiste na geração de erros, explorando a vulnerabilidade dos sistemas, de forma que os servidores em sua maioria, retornam muitos erros descritivos, que incialmente tinham o intuito de facilitar os programadores a encontrar erros em seus aplicativos. Diante disto o usuário malicioso, força a geração de erros do sistema, usando a injeção para causar erros de conversão de tipo, sintaxe, ou lógicos no BD, para obter informações da estrutura do banco de dados. (HALFOND, 2006).

Figura 13 - Campo usuário e senha.

Nome: Senha:

Fonte: Elaborada pelo autor.

Para melhor compreensão, na Figura 13, existem dois campos, como em aplicações web. Um atacante não sabendo um usuário e senha válidos, afim de descobrir informações, irá neste exemplo usar o comando having e mais uma sequência de caracteres, no primeiro campo, ilustrado na Figura 14.

Figura 14 - Utilização do having.

Nome: ' having 1=1 -- Senha: bláblá

Fonte: Elaborada pelo autor.

Neste momento os parâmetros irão para o banco de dados, como mostra a Figura 15.

Figura 15 - Consulta having

SELECT * FROM USUARIOS WHERE NOME = ' HAVING 1=1 -- SENHA = BLÁBLÁ

Fonte: Elaborada pelo autor.

A linguagem SQL de BDs, interpreta o "--", como início de comentário, logo tudo aquilo após esses caracteres será desconsiderado. A instrução permanece até o comando having, e gera um erro ilustrado na Figura 16.

Figura 16 - Erro Having

Microsoft OLE DB Provider for ODBC Drivers error ‘80040e14′ [Microsoft][ODBC SQL Server Driver][SQL Server]Column ‘usuarios.CODUSUA’ is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause. /arquivo.asp, line 94

Fonte: Elaborada pelo autor.

Diante deste erro o usuário com certo conhecimento, já consegue identificar uma tabela chamada USUARIOS, e o primeiro campo dela, que no caso é o CODUSUA.

Figura 17 - Utilização do Group by

```
Nome: ' group by usuarios.codusua having 1=1-- Senha:bláblá
```

Fonte: Elaborada pelo autor.

Ao utilizar o comando group by como mostrado na Figura 17, outro erro ocorre.

Figura 18 - Erro Group by

```
Microsoft OLE DB Provider for ODBC Drivers error ‘80040e14′ [Microsoft][ODBC SQL Server Driver][SQL Server]Column ‘usuarios.idade’ is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause. /arquivo.asp, line 94
```

Fonte: Elaborada pelo autor.

Ao analisar esse novo erro, o usuário consegue identificar outro campo da tabela chamado idade. Desta forma ao fazer isto sucessivamente, o atacante encontrará todos os campos da tabela.

Figura 19 - Comando Sum

```
Usuário: ' union select sum(codusua) from usuarios -- Senha :
```

Fonte: Elaborada pelo autor.

Na Figura 19, é usado o comando sum, que serve para soma, com o intuito de descobrir o tipo de dado da coluna codusua.

Figura 20 - Erro Sum

```
Microsoft OLE DB Provider for ODBC Drivers error ‘80040e07′ [Microsoft][ODBC SQL Server Driver][SQL Server]The sum or average aggregate operation cannot take a varchar data type as an argument. /arquivo.asp, line 94
```

Fonte: Elaborada pelo autor.

O servidor ao retornar este erro mostrado na Figura 20, informa o tipo de dado do campo codusua, um varchar. Assim como para descobrir os campos, pode-se utilizar deste comando para descobrir todos os tipos de dados de todos os campos da tabela usuarios. Desta forma, o atacante ao possuir todos as informações da tabela, pode inserir facilmente um registro, com o comando insert.

Os desenvolvedores das aplicações web, ao se conscientizarem sobre o SQL Injection baseado em erros, começaram a desativar as mensagens, outros, as tornaram genéricas, tudo com o objetivo de extinguir essa ameaça, mas em decorrência a estas ações, atacantes acabaram desenvolvendo outra técnica chamada Blind SQL Injection. (CLARKE, 2009).

2.6.4.2 Blind SQL Injection

Este tipo de ataque normalmente é recomendado, quando uma aplicação web está configurada para apresentar mensagens de erro genéricas. O Blind Injeção SQL é um tipo de ataque de injeção SQL que trabalha com o retorno do servidor, as solicitações de perguntas, no qual de acordo com as respostas são definida sua autenticidade. (OSWAP, 2013c).

Segundo Clarke (2009), este método que pode ser conhecido como ataque cego, é uma ferramenta que esta no conhecimento de todo atacante. Este faz a verificação se a injeção é possível através do retorno de saída do aplicativo web, sua principal função é a extração de informações do BDs e não necessita das mensagens de erros. (CLARKE, 2009).

Figura 21 - URL normal

```
http://www.meusite.com.br/Noticia.jsp?pressReleaseID=5
```

Fonte: Elaborada pelo autor.

Na Figura 21, é ilustrado uma URL de um site de notícias fictício, no qual está na quinta reportagem, de acordo com o pressReleaseID=5.

Figura 22 - URL teste Blind

```
www.meusite.com.br/Noticia.jsp?pressReleaseID=5  AND USER_NAME() = 'DBO'
```

Fonte: Elaborada pelo autor.

A notícia é exibida normalmente, mas ao adicionar USER_NAME (), uma função do servidor SQL, que retorna o nome do atual do usuário, e atualizar a página como na Figura 22, é feita uma pergunta ao servidor se o usuário é o DBO, se a reposta no caso for verdadeira, a página abrirá normalmente, caso a resposta não seja verdadeira, a consulta falhará e nenhuma notícia será mostrada.

Figura 23 - URL recuperação do nome de uma tabela

```
http://www.meusite.com.br/Noticia.jsp?pressReleaseID=5 AND
ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE
xtype='U'), 1, 1))) > 109
```

Fonte: Elaborada pelo autor.

Na Figura 23, é ilustrado uma pergunta um pouco mais complexa, que tem como meta, identificar o nome de uma tabela do bancos de dados, descobrindo um caracter por vez. A substring() junto com o select, irá retornar o primeiro caractere do resultado da consulta, já a função lower(), serve para converter o caracter retornado para minúsculo, e o comando ascii(), é utilizado para converter para o alfabeto ASCII o valor retornado. Assim quando esta URL for executada, e trazer a quinta notícia como resposta, o usuário malicioso saberá que a primeira letra da tabela está entre "N" e "Z", pois na tabela ASCII o número 109, representa a letra "m". Ao insistir neste comando alterando a comparação, o atacante conseguirá identificar mesmo que de uma forma trabalhosa o nome desta tabela, de modo a poder utilizar essa informação para adquirir outras, e assim conseguir invadir a aplicação.

2.6.4.3 Controle do Sistema Operacional a partir da Invasão

Após um ataque bem sucedido a uma aplicação web, é possível executar algumas estratégias que possam levar ao acesso remoto do sistema. No SQL Server, um sistema de gerenciamento de banco de dados, quando se possui os privilégios de administrador, o qual não é difícil, é possível a partir da stored procedure xp_cmdshell, realizar uma conexão com o sistema operacional, executando comandos deste, que serão respondidos juntos ao retorno. (CLARKE, 2009).

Como a xp_cmdshell, existem diversos outros tipos de stored procedure e comandos no SQL Server, que possuem o papel de realizar essa conexão com o sistema operacional, visando um objetivo diferente, como ilustrado na Figura 24.

Figura 24 - Comandos operacionais.

Bulk Copy Program (BCP)	Utilizado para criação de arquivos no sistema.
XP_CMDSHELL	Utilizado execução de comandos no sistemas, para criação de arquivos e facilitar seu upload.
BULK INSERT	Utilizado para leitura dos arquivos no sistema.
SP_OACREATE	Utilizado para escrever arquivos no sistema.

Fonte: Elaborada pelo autor.

2.7 MÉTODOS DE PROTEÇÃO CONTRA O SQL INJECTION

Este método de ataque que usa injeção, se demonstrou um grande risco para os usuários, diante disso, desenvolvedores e programadores, desenvolveram algumas formas para se proteger deste. Contudo os métodos de proteção se embasam em boas práticas de implementação, visando a segurança das aplicações.

Como já mostrado em suas técnicas de ataque, um dos problemas primários, que tornam as aplicações web vulnerável ao SQL Injection, são os comandos e códigos, facilmente amarrados aos caracteres que são enviados ao servidor. Para este problema as linguagens de programações modernas, criaram as instruções parametrizadas, que trabalham com espaços reservados e variáveis de ligação, de forma a não utilizar diretamente as informações vindas do usuário. No entanto quando uma aplicação web, utiliza o SQL dinâmico, criando funções ou utilizando uma Stored Procedure (procedimento de armazenamento) em tempo de execução, a injeção ainda poderá ocorrer. (CLARKE, 2009).

Segundo Kost (2004), a forma mais efetiva de proteção contra esta técnica de ataque é o uso de variáveis de ligação, além de melhorar o desempenho da aplicação. Deve se utilizar as variáveis de ligações em todas as instruções possíveis, independente de quando ou onde elas forem executadas, e nunca amarrar diretamente uma instrução a um parâmetro recebido.

Stored Procedure são programas, armazenados em banco de dados, que são utilizados para acessar o próprio. Uma Store Procedure pode ser escrita em diferente linguagens de acordo com o banco, além de possibilitar a configuração de acesso base aos BDs. Seu uso pode diminuir o impacto da técnica de Ataque SQL Injection em aplicações. (CLARKE, 2009).

Outro ponto de grande importância que se deve realçar quando abordar os método de proteção, é a validação dos dados de entrada do usuário, trata-se de um teste feito pela aplicação com o intuito de verificar se a entrada está de acordo com um padrão pré estabelecido. Esta validação é divida em duas vertentes, a lista branca, conhecida como validação positiva, e a lista negra, conhecida como validação negativa. (CLARKE, 2009).

A lista branca, é considerada a mais confiável entre as duas, entretanto pode ser mais complexa de aplicá-la em alguns ambientes. Este tipo de validação consiste basicamente em verificar uma variedade de possibilidades, por exemplo, ao preencher um campo refente a um número de cartão de crédito, para este tipo de método, será analisado se a entrada realmente é somente numérica, se a entrada possui valor positivo como tem que ser, se o valor possui entre 13 e 16 números, se o número inserido está de acordo com o cálculo para emissão de cartões, entre outros. Já a lista negra, tem como objetivo bloquear a entrada de caracteres ou uma sequência deles, que são considerados uma ameaça. Esta forma de proteção não é muito segurada, e é considerada ruim, pois sobrecarrega o sistema percorrendo uma grande quantidade de caracteres para validação, além de muito difícil mantê-la atualizada, devido a grande quantidade de técnicas que surgem rapidamente. (CLARKE, 2009).

Os padrões e funções personalizadas disponibilizados pelos bancos de dados, que podem alterar senhas, criar usuários, são igualmente vulneráveis a exploração, no qual sua utilização pode facilitar o determinado ataque, desta forma é aconselhável restringi-los aos que forem realmente necessários. (KOST, 2004).

De acordo com OSWAP (2007), como ações de proteção para ataques de injeção, deve-se emitir mensagens de erros menos detalhadas possível, afim de serem mais genéricas, além de utilizar as API’s (Application Programming Interfaces), funcionalidades dos BDs que já implementam de forma segura a entrada dos parâmetros, e a realização da codificação dos dados de entrada para a representação interna da aplicação, antes de qualquer tipo de validação seja feita.

2.8 FERRAMENTAS UTILIZADAS

Para realização deste trabalho, foram utilizadas algumas ferramentas que serão brevemente conceituadas.

2.8.1 SQLMAP

SqlMap é uma ferramenta código aberto muito conhecida, que automatiza o processo de detectar e explorar as falhas de injeção SQL. Ela possui um poderoso sistema de detecção e uma grande quantidade de recursos que podem até permitir ao utilizador, acessar o sistema de arquivo do alvo e aplicar comandos em seu sistema operacional. Na figura 25, é ilustrado o prompt de comando com a utilização da ferramenta relatada. (SQLMap...,2015).

Figura 25 - Ferramenta SQLMap.

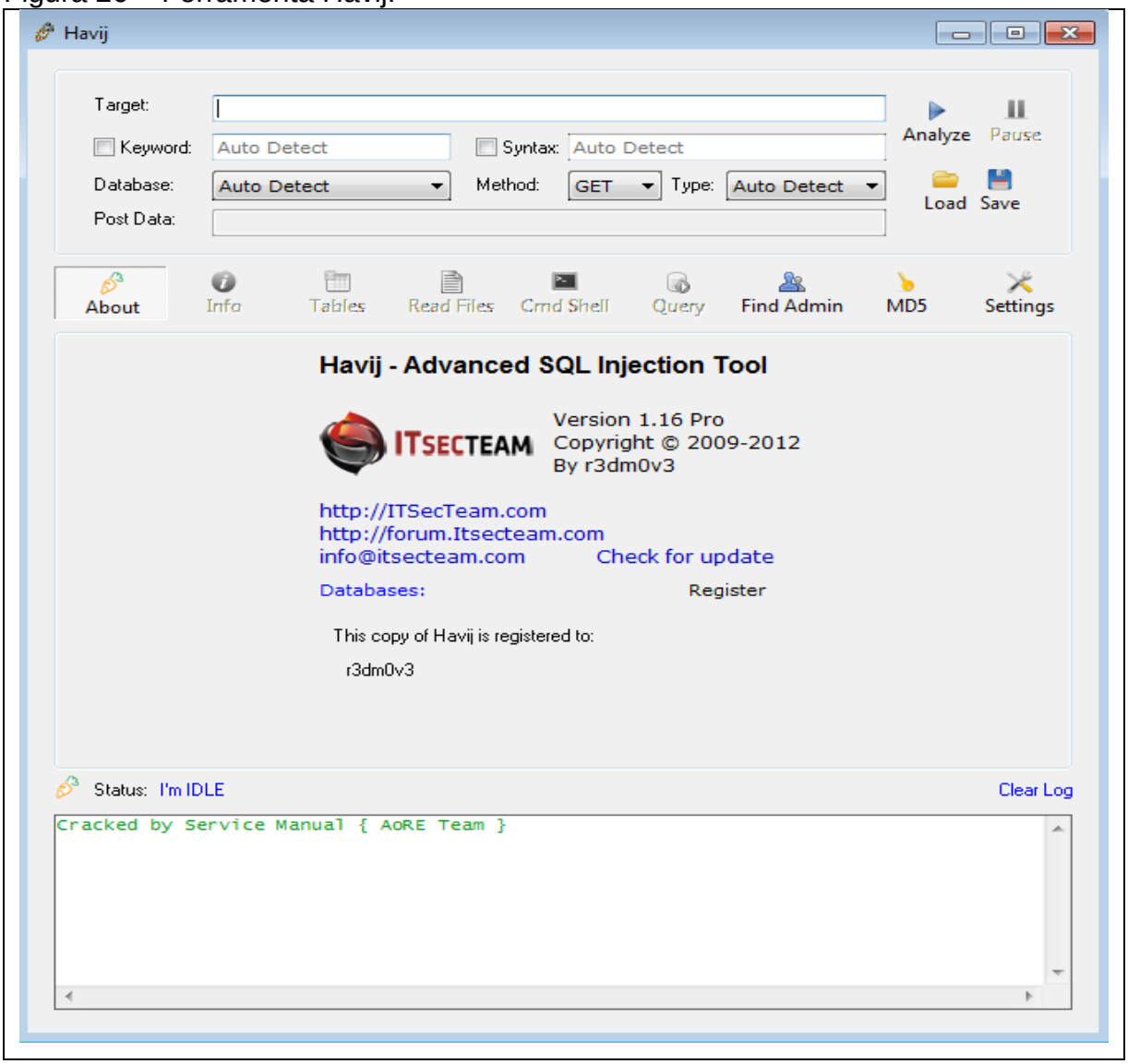
Fonte: SQLMAP (2015).

2.8.2 Havij

O Havij assim como o SQLMap é uma ferramenta automatizada de injeção SQL, que explora vulnerabilidades de uma aplicação web. Desenvolvida com o propósito de deixar páginas da web mais seguras, analisando se existe ou não, a

possibilidade de invasão. Na Figura 26, é ilustrado a interface do programa. (WIKIPÉDIA, 2015).

Figura 26 – Ferramenta Havij.

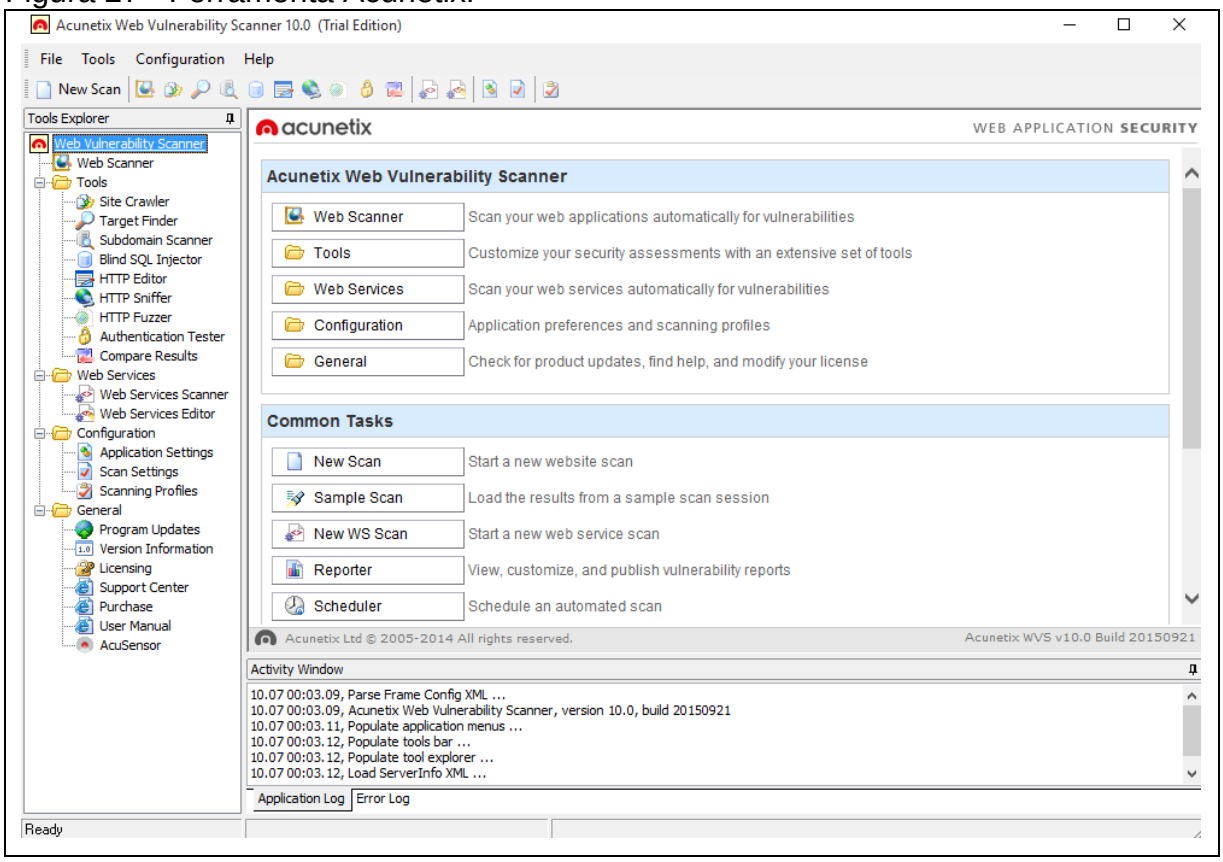


Fonte: Elaborada pelo autor.

2.8.3 Acunetix Web Vulnerability Scanner 10.0

Essa ferramenta possui como foco analisar aplicações web e detectar vulnerabilidades. Desenvolvida pela empresa Acunetix, foi pioneira na área de segurança de aplicações web, na análise do site e detecção de vulnerabilidades. Na Figura 27, é ilustrado a interface do programa. (ACUNETIX, 2015).

Figura 27 - Ferramenta Acunetix.



Fonte: Elaborada pelo autor.

2.8.4 Python versão 2.7.5

Compilador Python 2.7.5, possui o objetivo de compilar a linguagem de programação denominada python. Uma linguagem de programação orientada a objetos, que possui código aberto, desenvolvida pela Python Software Foundation, uma organização que não possui fins lucrativos. (PYTHON, 2015).

3 TRABALHOS CORRELATOS

Neste capítulo serão apresentados alguns trabalhos pesquisados que fazem relação com o tema deste trabalho, o JsafeFilter e o Cross-Site Scripting. Existem vários outros projetos que utilizam do estudo destes meios de ataque e prevenção, porém, abaixo situa-se os dois que obtiveram maior relevância dentre os encontrados.

- O trabalho realizado por Magalhães [2010?], tem como objetivo apresentar alternativas de segurança para anular o SQL Injection, com a utilização do JsafeFilter. Um método de detecção e prevenção de ataques em aplicações web. Tal detecção é realizada através da manipulação de campos de entrada de dados, pois a ferramenta retira a responsabilidade do tratamento daqueles campos por meio do escopo das aplicações web, fazendo com que seja caracterizada com um filtro de requisições HTTP.
- O Cross-Site Scripting (XXS) é um dos meios de ataque à aplicações web, causado por scripts maliciosos pelo lado cliente ou servidor, onde os dados de entrada do usuário não são validados de forma adequada, permitindo o roubo de informações confidenciais e sessões do usuário e compromentendo o navegador web cliente e a integridade do sistema em execução, igualmente ao SQL Injection. Desta forma o trabalho realizado por Nunan [2010?], visa classificar quais páginas web são vulneráveis a este tipo de ataque.

4 METODOLOGIA

De acordo com Gil (1991), uma pesquisa bibliográfica normalmente é feita quando não se possui informações suficientes sobre o assunto. Esse tipo de pesquisa, trata-se de um procedimento racional e sistemático, que possui como o objetivo principal responder as dúvidas dos problemas referente ao tema escolhido, além de ampliar a visão do investigador sobre o mesmo. Esta é desenvolvida e embasada principalmente em livros e artigos científicos,

Levando em consideração estas informações este trabalho foi desenvolvido em duas partes, compostas pela investigação dos assuntos abordados nos tópicos, e a realização do ataque em uma aplicação web com a utilização da técnica SQL Injection.

No primeiro momento foi desenvolvida, a revisão da literatura, no qual foi realizada uma pesquisa bibliográfica, de modo a visar a busca de conhecimento referente ao tema, afim de levantar informações necessárias para sua documentação e posteriormente seu desenvolvimento. Nesta revisão literária foram levantadas informações sobre alguns tipo de ataque que utilizam do método de injeção, mas de modo a sempre enfatizar a técnica SQL Injection, além de alguns assuntos como aplicação web, linguagem sql, bancos de dados, afim de auxiliar no compreendimento total do objetivo que este trabalho tem em foco.

Após a realização da primeira etapa, foi dado início a segunda parte deste trabalho, que possui como meta, demonstrar de maneira prática o que foi estudado na pesquisa bibliográfica, de forma a ser realizado um ataque a uma aplicação web via SQL Injection.

Para realização desta demonstração, primeiramente foi realizado um estudo, com o objetivo de verificar em qual aplicação web seria efetuada a invasão, de forma a utilizar aplicações web disponibilizadas na internet, que foram desenvolvidas com o propósito de servirem como testes para ataques, ou se seria viável a criação de um site extremamente simples com as vulnerabilidades necessárias.

Ao término desse estudo, foi concluído que o ideal seria a utilização de sites já prontos e próprios para esse fim, além da utilização de ferramentas. Ferramentas estas, com o objetivo de auxiliar nos ataques, pois mesmo com todo estudo feito, é preciso de muito conhecimento profundo sobre tal, de forma que, também seja muito complicado e complexo atacar uma aplicação web.

Diante disso, foram escolhidos três sites que possuem vulnerabilidades para fins de estudo para serem atacados, de modo que todas as aplicações seriam exploradas em duas vulnerabilidades para fim de comparação.

Primeiramente foi utilizado em cada site, http://testphp.vulnweb.com/, http://testasp.vulnweb.com/, e http://testaspnet.vulnweb.com/, uma ferramenta bastante intuitiva chamada Acunetix Web Vulnerability Scanner 10.0, que possui o objetivo de escanear o site alvo afim de encontrar qualquer tipo de vulnerabilidade sobre o mesmo.

Assim ao escanear todos os sites escolhidos e definir as vulnerabilidades que seriam atacadas, foram utilizadas duas ferramentas, chamadas SQLMap e o Havij. Ferramentas que estão no “mercado” e são bastante conhecidas nesse meio. Esses dois recursos possuem como principal foco, automatizar o processo de invasão a sites vulneráveis ao SQL Injection. Desta forma ambos foram utilizados para atacar os três sites em suas duas vulnerabilidades alvo.

Ao fim de todos os ataques foi construído uma tabela comparativa entre as ferramentas com os resultados obtidos, de modo que as informações encontradas e o tempo que foi cronometrado manualmente, foram os critérios avaliados. Demonstrando que além de servirem como “arma”, podem e são utilizadas como auxílio na proteção de diversos sites, de forma a contribuir e informar os desenvolvedores e os usúarios, de um dos muitos meios de ataque prejudiciais que circulam pela Internet.

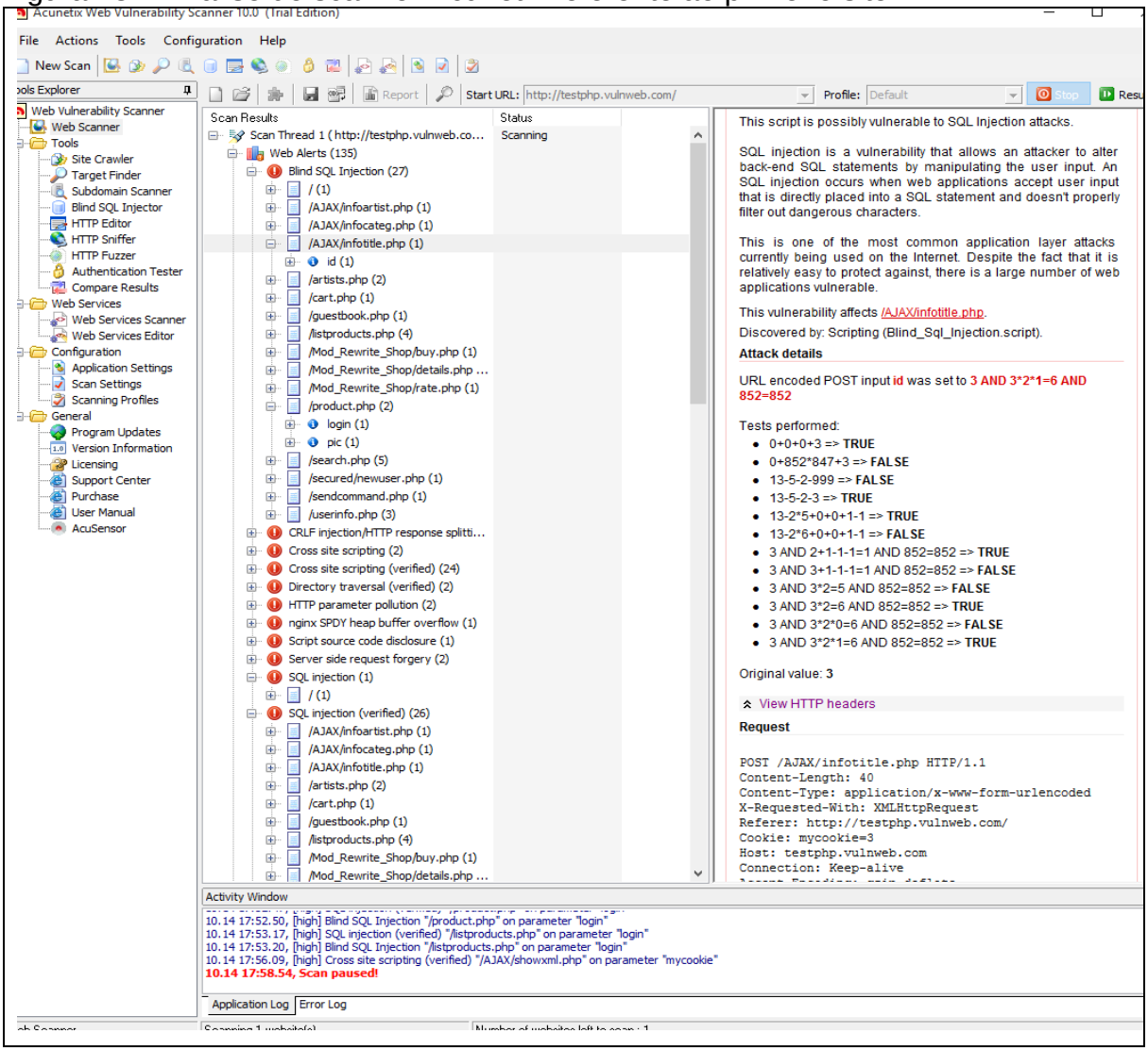
5 RESULTADOS

Nesta etapa do trabalho, assim como descrito na metodologia, foram efetuados alguns ataques com a utilização de certos softwares nas determinadas aplicações web, obtendo certos resultados e conclusões.

Primeiramente, foi utilizado o programa Acunetix Web Vulnerability Scanner 10.0, com a inteção de encontrar vulnerabilidades nos respectivos sites escolhidos. Este software não só encontra vulnerabilidades para o ataque SQL Injection, mas como para diversos outros tipos de invasões.

O site testphp.vulnweb.com, foi o primeiro a ser escaneado pelo scanner acunetix, sua análise pode ser observada na Figura 28.

Figura 28 – Análise do scanner Acunetix referente ao primeiro site.



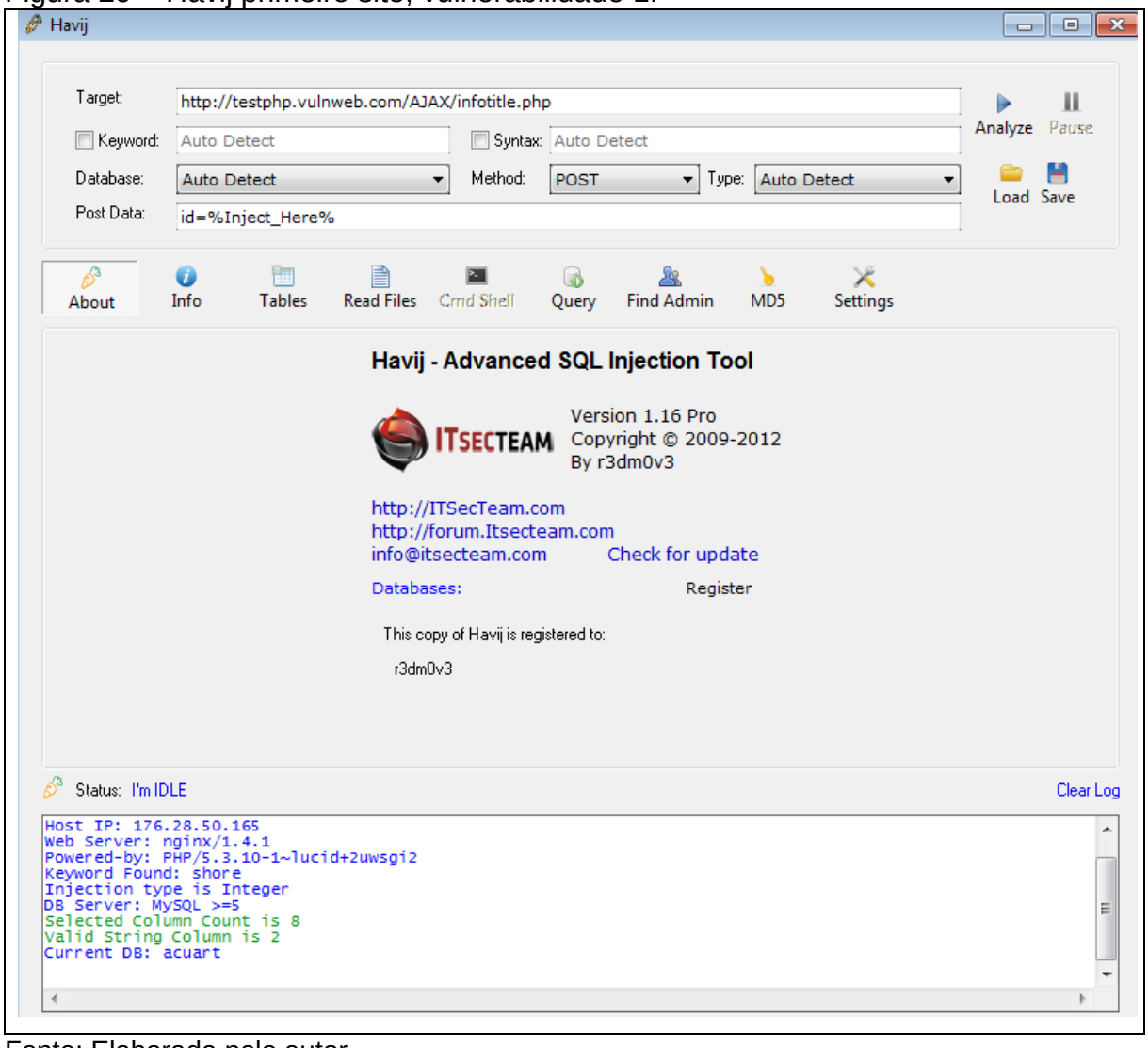
Fonte: Elaborada pelo autor.

O processo de busca por vulnerabilidades feita por este programa não é muito rápido, por esse motivo o scan foi pausado como pode-se observar pois no respectivo site já havia se encontrado muitas vulnerabilidades para o SQL Injection.

Realizada a análise, foram escolhidas duas vulnerabilidades para serem exploradas, buscando por serem de tipos de métodos diferentes para uma futura comparação. Deste modo foram optados pelas duas urls demonstradas na Figura 28, /AJAX/infotitle.php?id e a /product.php?pic, sendo uma do método POST e uma do método GET respectivamente.

Com as vulnerabilidades definidas, o passo seguinte foi dar início ao primeiro ataque com a utilização do software Havij.

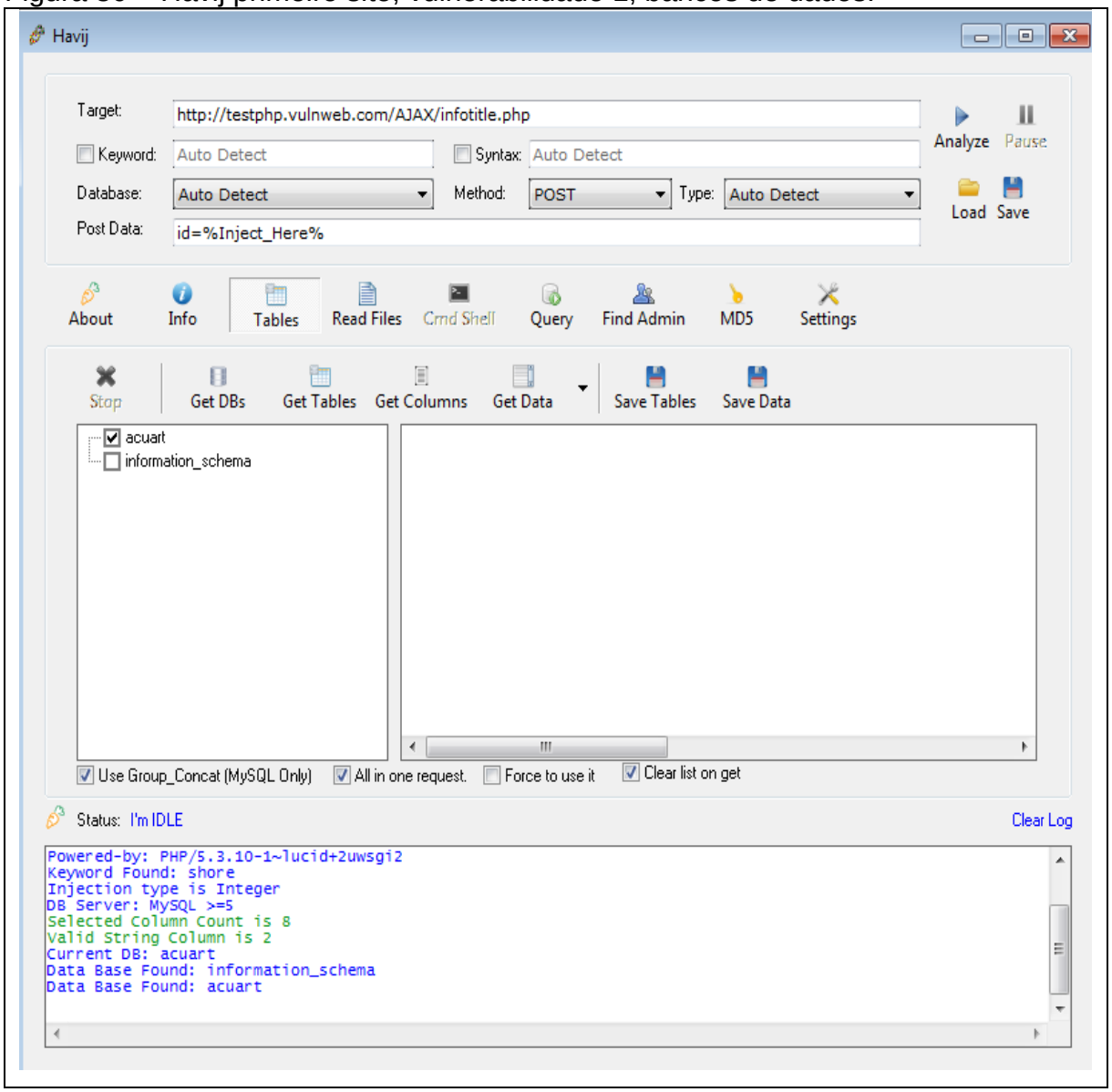
Figura 29 – Havij primeiro site, vulnerabilidade 1.



Fonte: Elaborada pelo autor.

No campo Target do Havij foi colocado a url vulnerável do site, como é ilustrado na Figura 29, e separadamente foi posto a variável que será injetada na área Post Data. Desta forma o programa, a partir desses dados conseguiu encontrar o banco de dados acuart e descobrir que o sistema de gerenciamento de banco de dados é um MySQL.

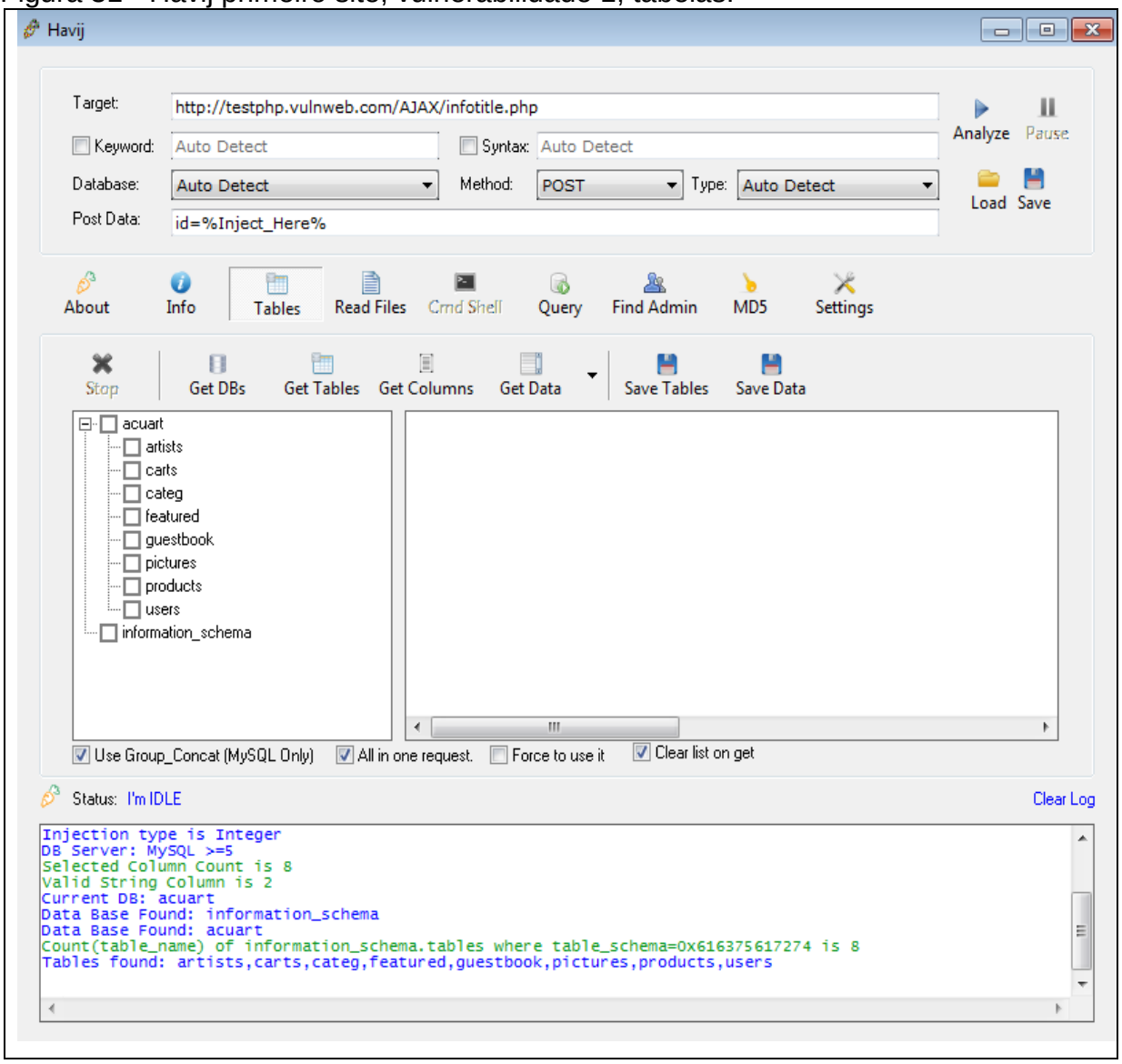
Figura 30 – Havij primeiro site, vulnerabilidade 1, bancos de dados.



Fonte: Elaborada pelo autor.

Procurando por novos bancos de dados o software encontrou mais um, chamado information_schema, como é ilustrado na Figura 30.

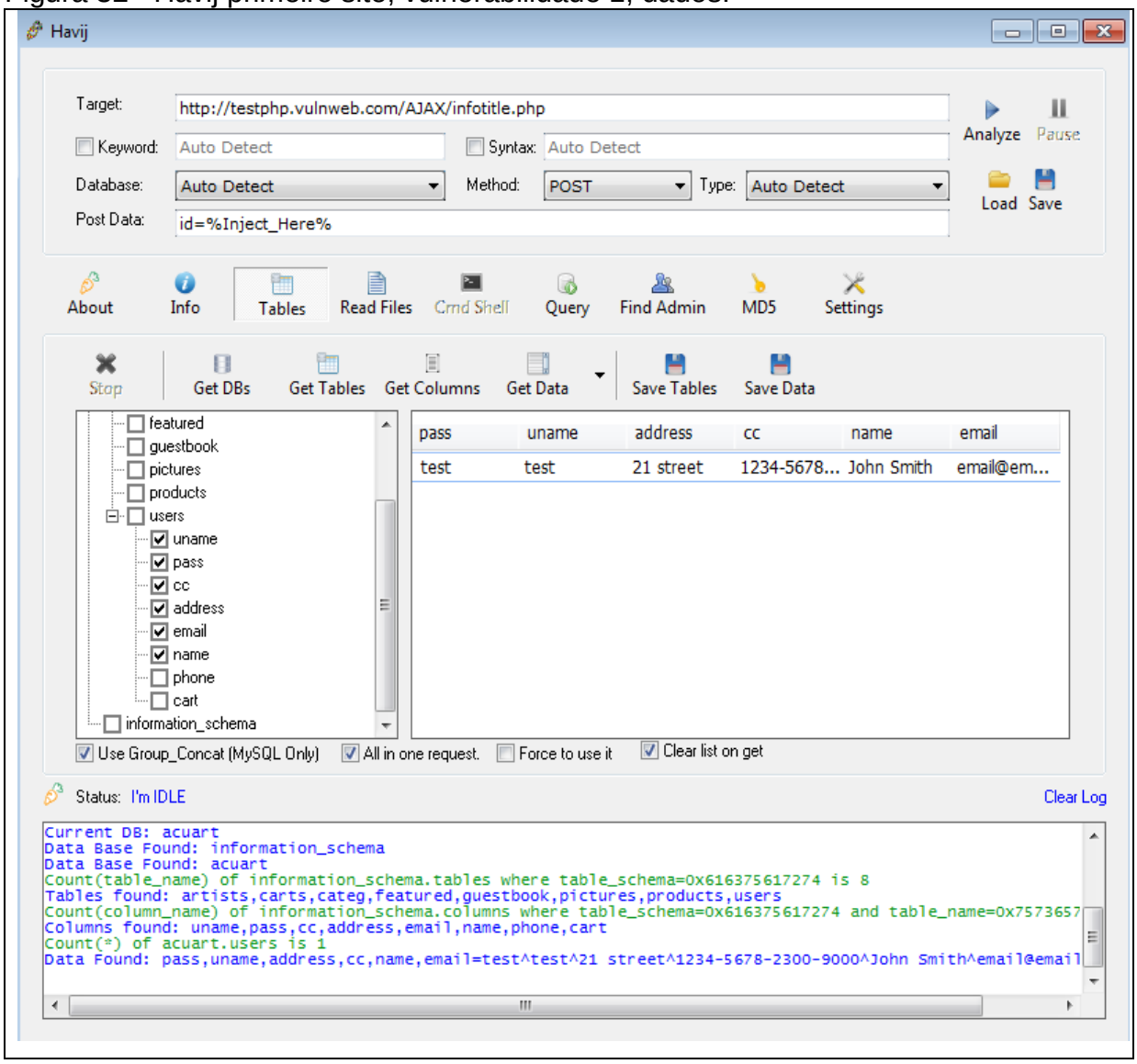
Figura 31 - Havij primeiro site, vulnerabilidade 1, tabelas.



Fonte: Elaborada pelo autor.

Ao explorar o primeiro banco de dados encontrado, o acuart, foram localizadas oito tabelas, que podem ser vistas na Figura 31. Diante disto foi escolhida a tabela users, que muito provavelmente contenha informações dos usúarios para ser expandida, obtendo oitos colunas. Assim ao regastar os dados contidos em algumas das colunas achadas foi retornado uma linha de dados com algumas informações, como pode-se obsevar na Figura 32.

Figura 32– Havij primeiro site, vulnerabilidade 1, dados.



Fonte: Elaborada pelo autor.

Esse mesmo ataque foi realizado com a utilização da outra ferramenta, o SQLMap. Esse recurso depois de instalado é utilizado a partir do prompt de comando, à base de comandos. Desta forma o primeiro comando executado com o objetivo de encontrar os bancos de dados é ilustrado no início da Figura 33.

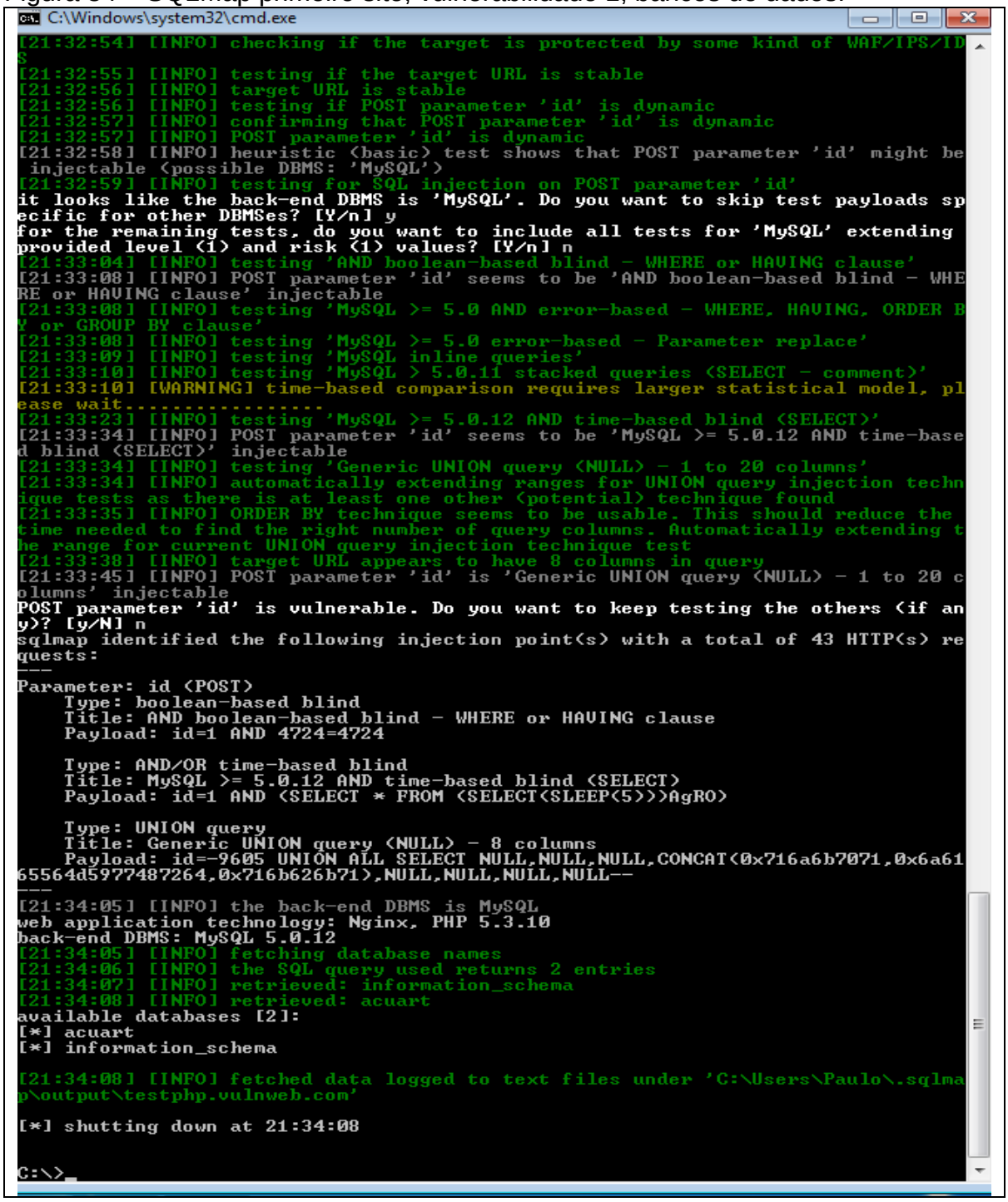
Figura 33 - SQLMap primeiro site, vulnerabilidade 1.

Fonte: Elaborada pelo autor.

Durante a execução do comando, o software faz algumas perguntas após sugerir que o modelo de gerenciamento de banco de dados é o MySQL, questionando se é necessário continuar com testes específicos para outros bancos de dados para garantir o tipo de BD, além de questionar se é válido aplicar todos testes para o BD MySQL. Para as duas perguntas foram respondidos sim e não respectivamente, pois o intuito desta invasão é aplicar um ataque simples, para fins

comparativos. De modo que o programa em seguida as respostas, continua o ataque, e ao chegar ao final dele, o recurso confirma que o parâmetro id que foi passado é vulnerável e questiona, se queremos continuar com testes de outros parâmetros, e novamente foi respondido não.

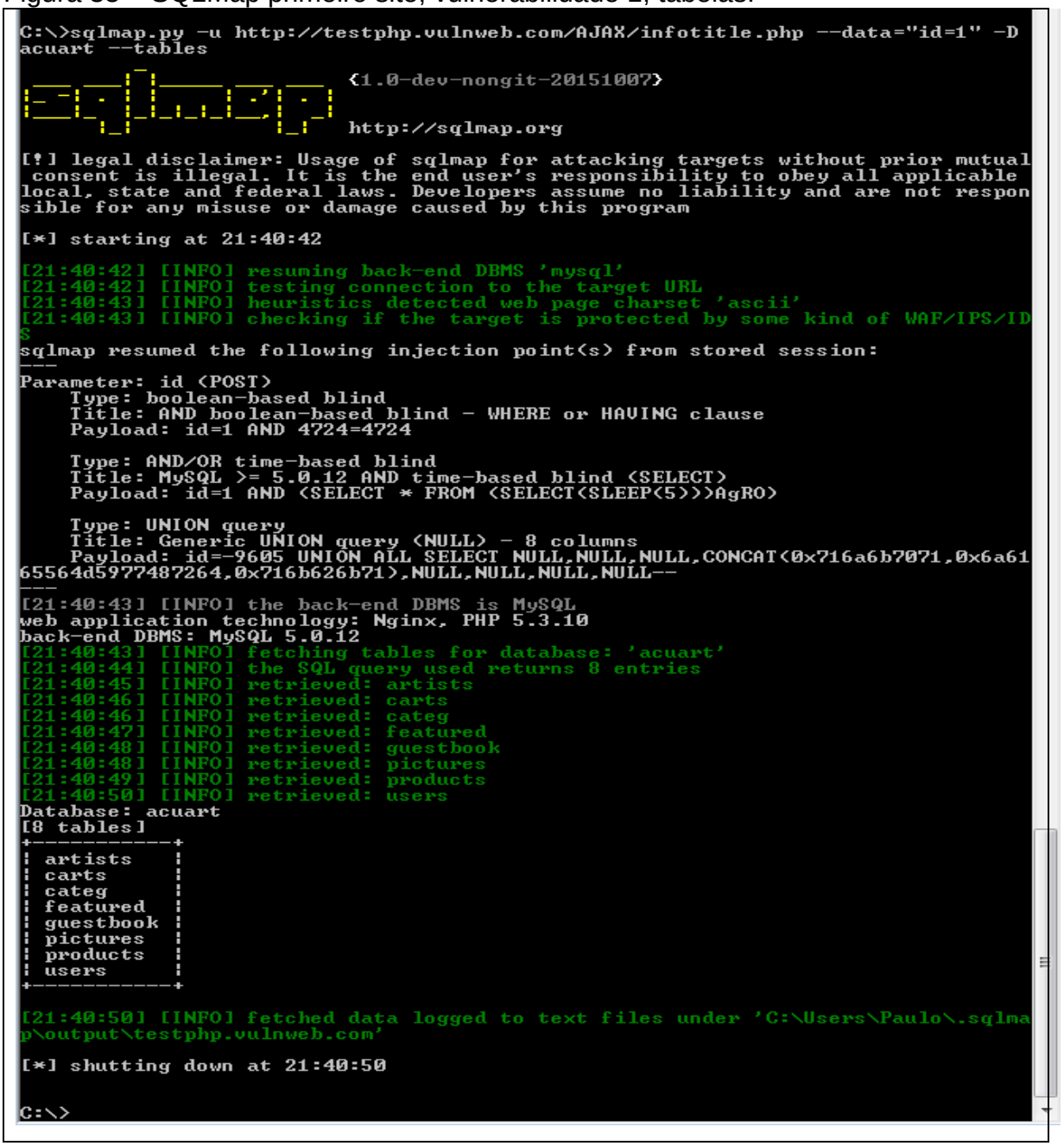
Figura 34 – SQLMap primeiro site, vulnerabilidade 1, bancos de dados.



Fonte: Elaborada pelo autor.

Após a conclusão da análise, o SQLMap, encontra igualmente ao Havji, dois bancos de dados, acuart e information_schema, ilustrados na Figura 34.

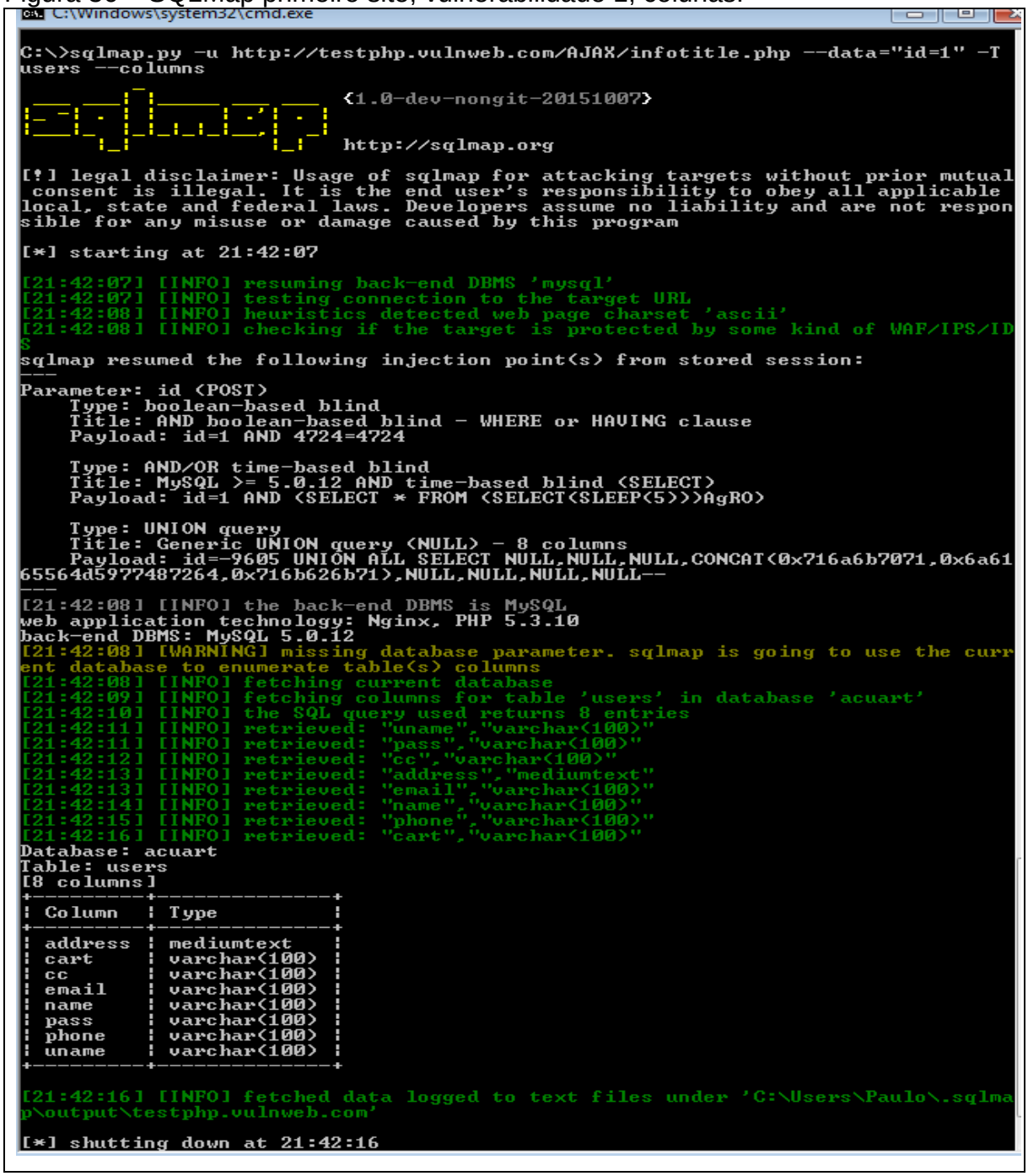
Figura 35 – SQLMap primeiro site, vulnerabilidade 1, tabelas.



Fonte: Elaborada pelo autor.

Já na Figura 35 é ilustrado em seu início o segundo comando aplicado, no qual tem a intenção de encontrar as tabelas do banco de dados descoberto no primeiro ataque, o acuart. Assim ao final deste ataque são encontradas oito tabelas, as mesmas encontradas pelo programa Havij.

Figura 36 – SQLMap primeiro site, vulnerabilidade 1, colunas.



Fonte: Elaborada pelo autor.

De modo que para encontrar as colunas e seus respectivos dados foram executados mais dois comandos como pode ser visto nas Figuras 36 e 37, retornando 8 colunas e uma linha de dados, igualmente ao ataque executado pelo recurso Havij.

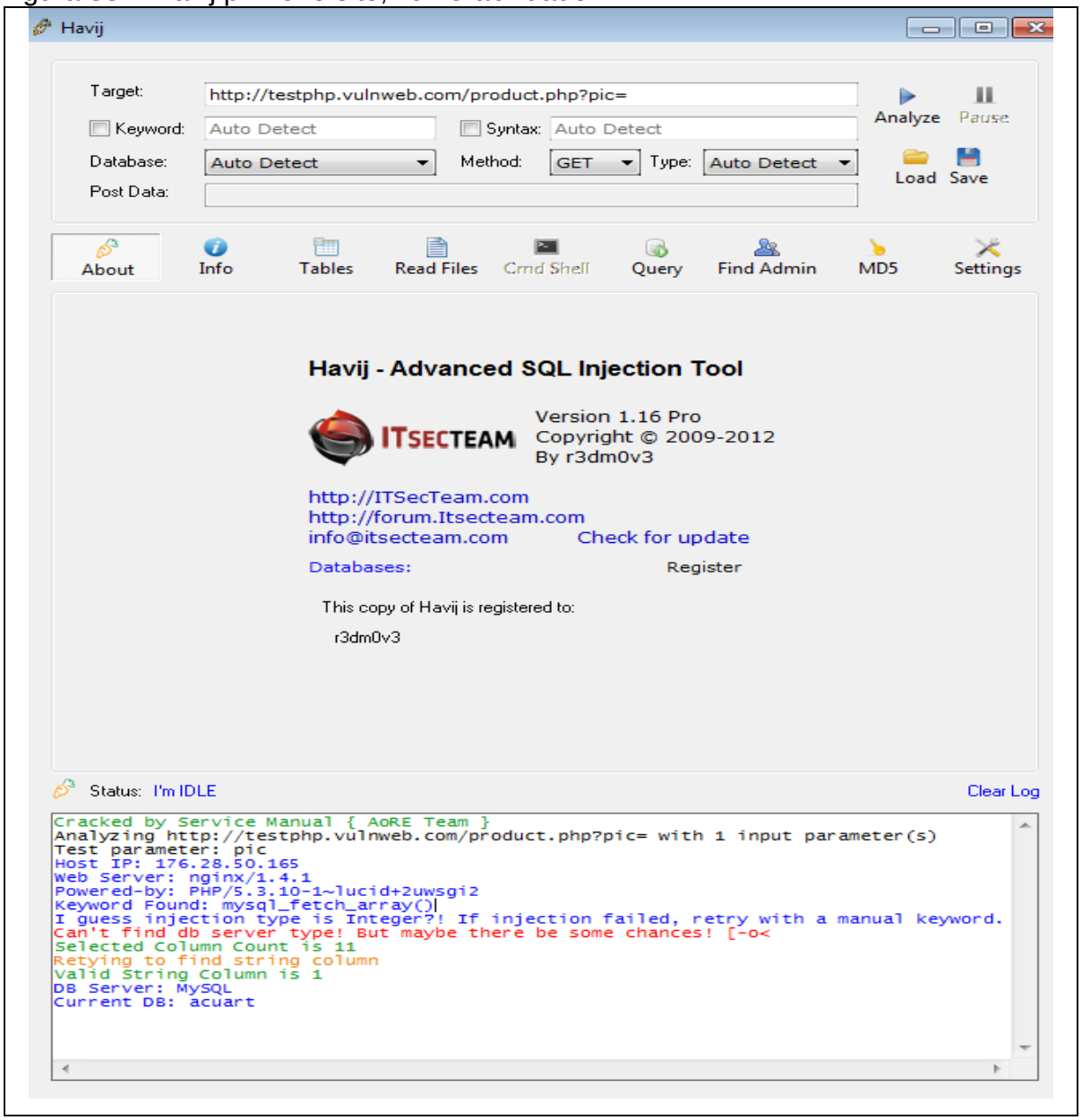
Figura 37 – SQLMap primeiro site, vulnerabilidade 1, dados.

Fonte: Elaborada pelo autor.

Ao encerrar os ataque com essa url, foi dado início ao próximo ataque com as ferramentas Havij e SQLMap, utilizando a vulnerabilidade encontrada pela análise do Acunetix na url /product.php?pic deste mesmo site.

Diferentemente do ataque anterior realizado pelo Havij, neste foi colocado o parâmetro vulnerável, no que no caso é a variável pic, juntamente a url na área Target do software, devido a este ser do método GET.

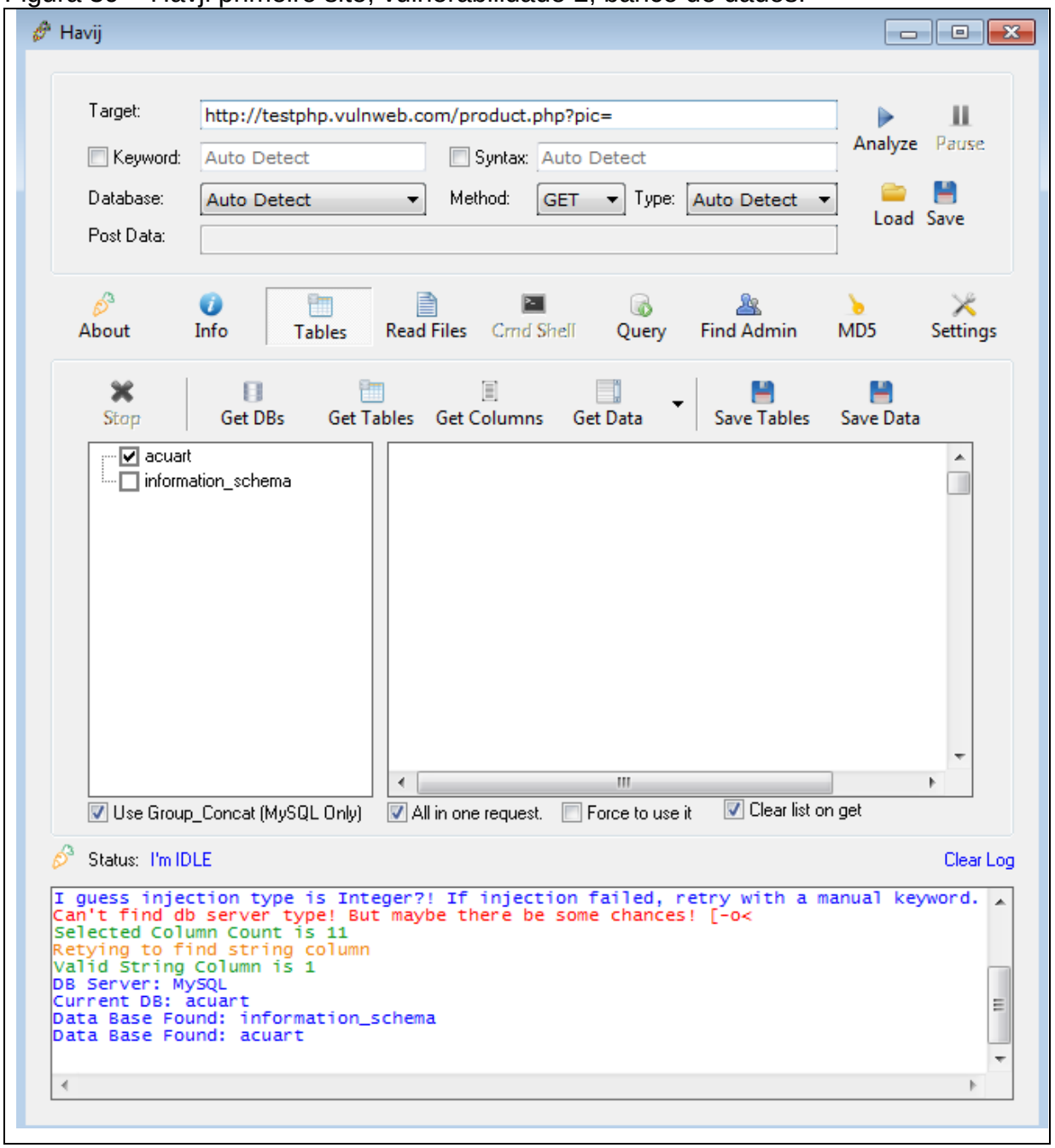
Figura 38 – Havij primeiro site, vulnerabilidade 2.



Fonte: Elaborada pelo autor.

Inicialmente este ataque encontrou o banco de dados acuart e o tipo de SGBD, MySQL, como é ilustrado na Figura 38. De modo que ao procurar por novos BDs igualmente feito anteriormente, foi encontrado novamente o banco, information_schema, ilustrado na Figura 39.

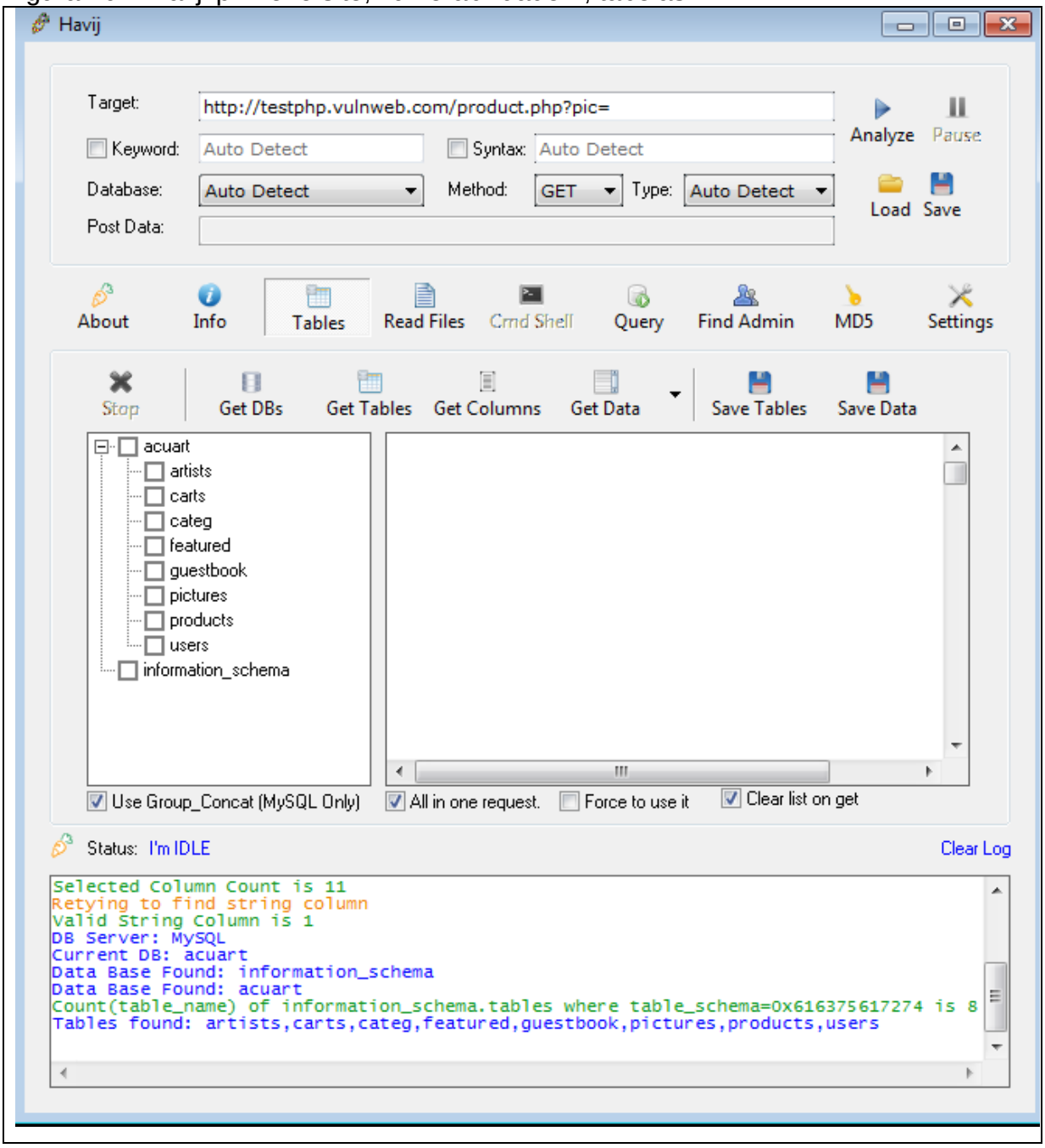
Figura 39 – Havji primeiro site, vulnerabilidade 2, banco de dados.



Fonte: Elaborada pelo autor.

Para manter um padrão, após identificar os BDs, o banco acuart foi novamente explorado, de forma que, foram encontradas 8 tabelas neste, que no caso são as mesmas encontradas anteriormente nos ataques. As tabelas obtidas são ilustradas na Figura 40.

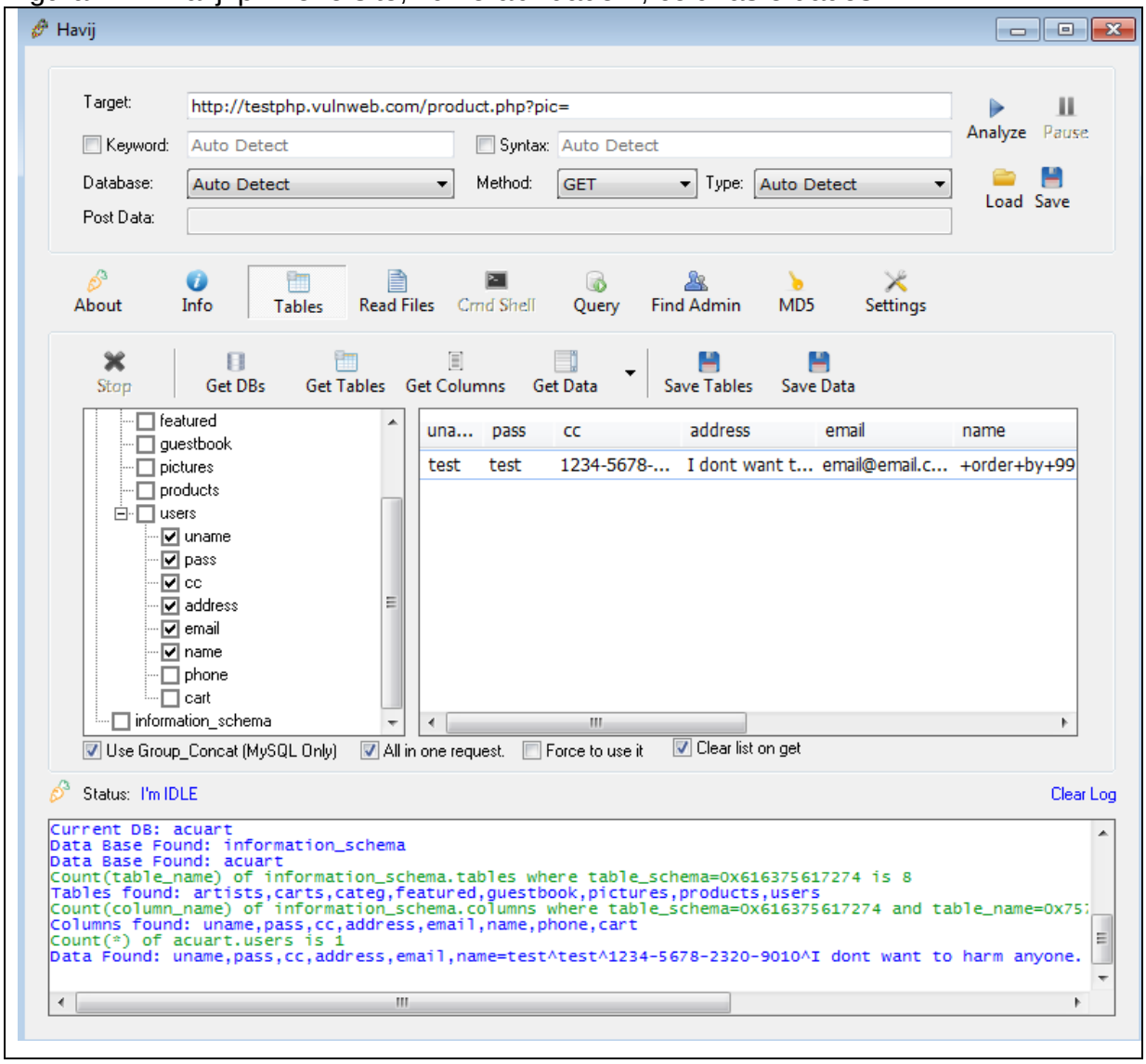
Figura 40 – Havji primeiro site, vulnerabilidade 2, tabelas.



Fonte: Elaborada pelo autor.

Desta forma o último passo deste ataque, mantendo um padrão, foi explorar a tabela users, com o objetivo de descobrir suas respectivas colunas e seus dados. Assim ao explorar tal foi obtido novamente 8 colunas, e uma linha de dados, que são ilustrados na Figura 41. O único diferencial do que foi encontrado na primeira url vulnerável testada, do método Post, deste mesmo site, são as informações encontradas nas colunas .

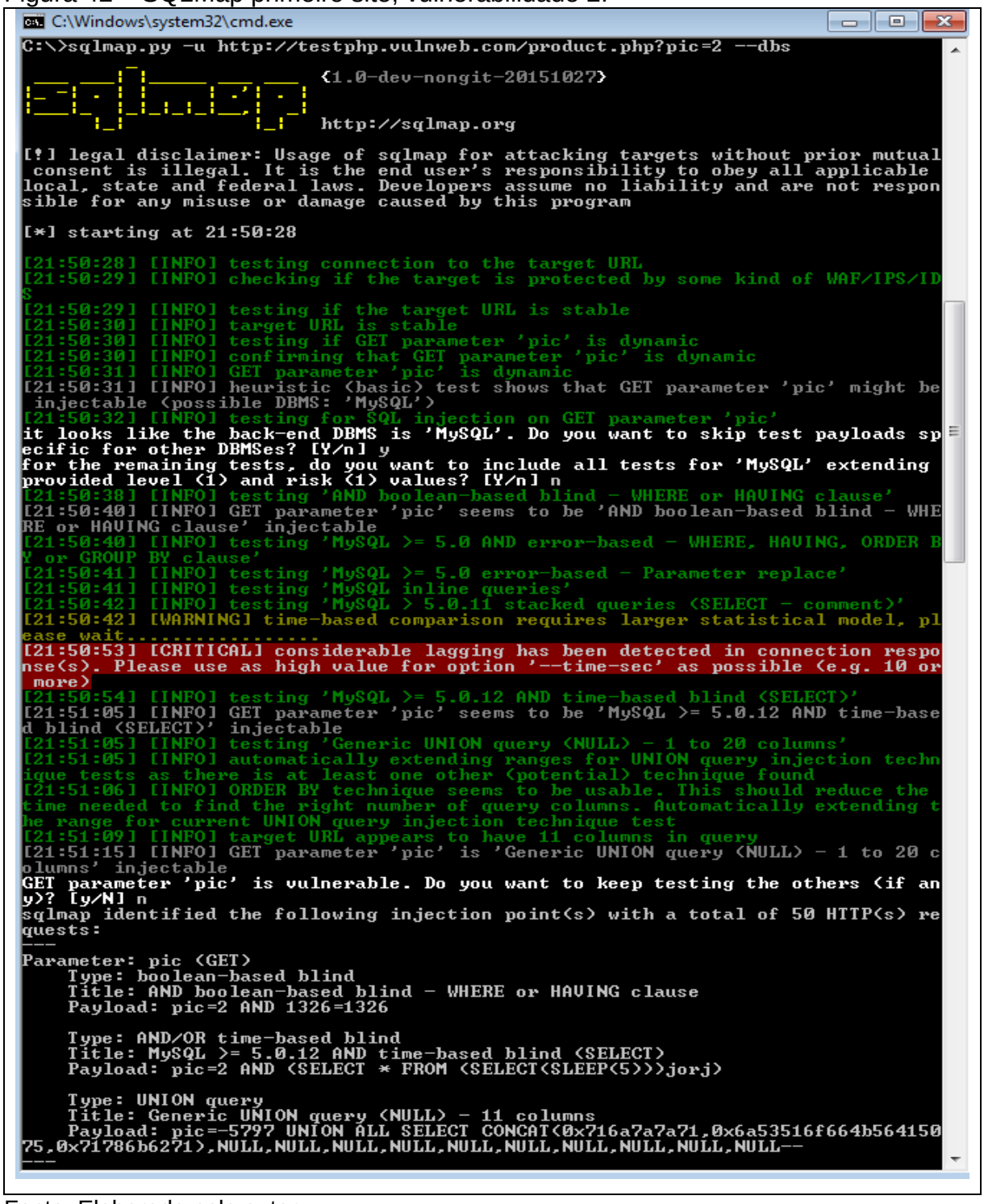
Figura 41 – Havji primeiro site, vulnerabilidade 2, colunas e dados.



Fonte: Elaborada pelo autor.

Ao continuar o processo de ataques, agora com o SQLMap, foi aplicado o primeiro comando utilizando a mesma url vulnerável que acabou de ser utilizada no Havij. O comando busca por bancos de dados fazendo as mesmas três perguntas feita anteriormente durante a primeira análise realizada com o recurso, pergutando se é necessário continuar com testes específicos para outros bancos de dados para garantir o tipo de sistema de gerenciamento de BDs encontrado até o momento, que no caso é o MySQL, questiona se é válido aplicar todos testes para o BD MySQL, e por últmo ao chegar ao final do ataque pergunta se queremos continuar com testes de outros parâmetros. As respostas para as três questões foram as mesmas utilizadas da primeira vez.

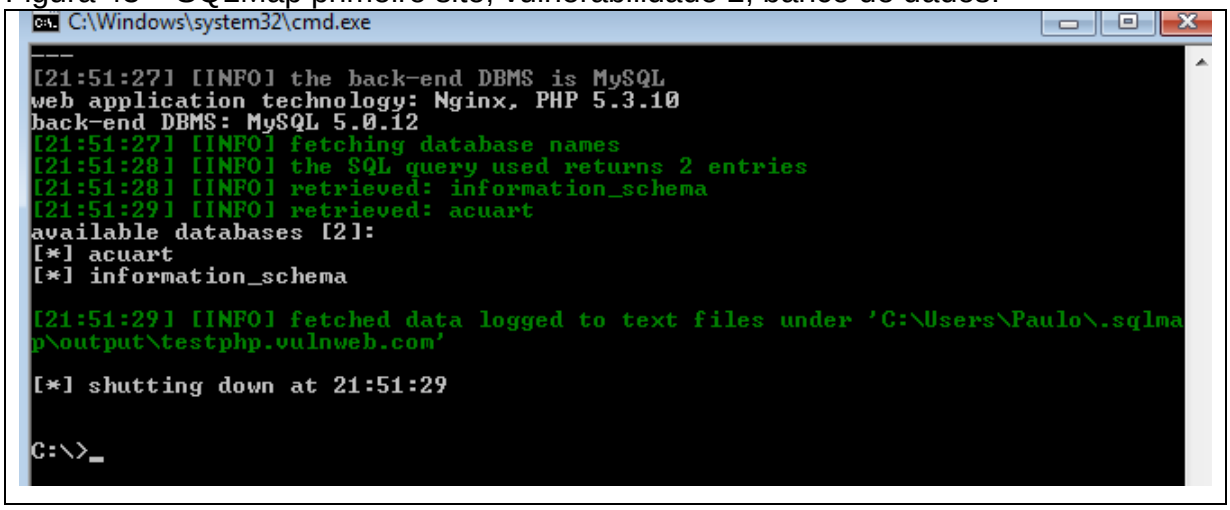
Figura 42 – SQLMap primeiro site, vulnerabilidade 2.



Fonte: Elaborada pelo autor.

Nas Figuras 42 e 43, pode-se observar a partir do comando executado, os testes feitos pela ferramenta, assim como os bancos de dados obtidos.

Figura 43 – SQLMap primeiro site, vulnerabilidade 2, banco de dados.



Fonte: Elaborada pelo autor.

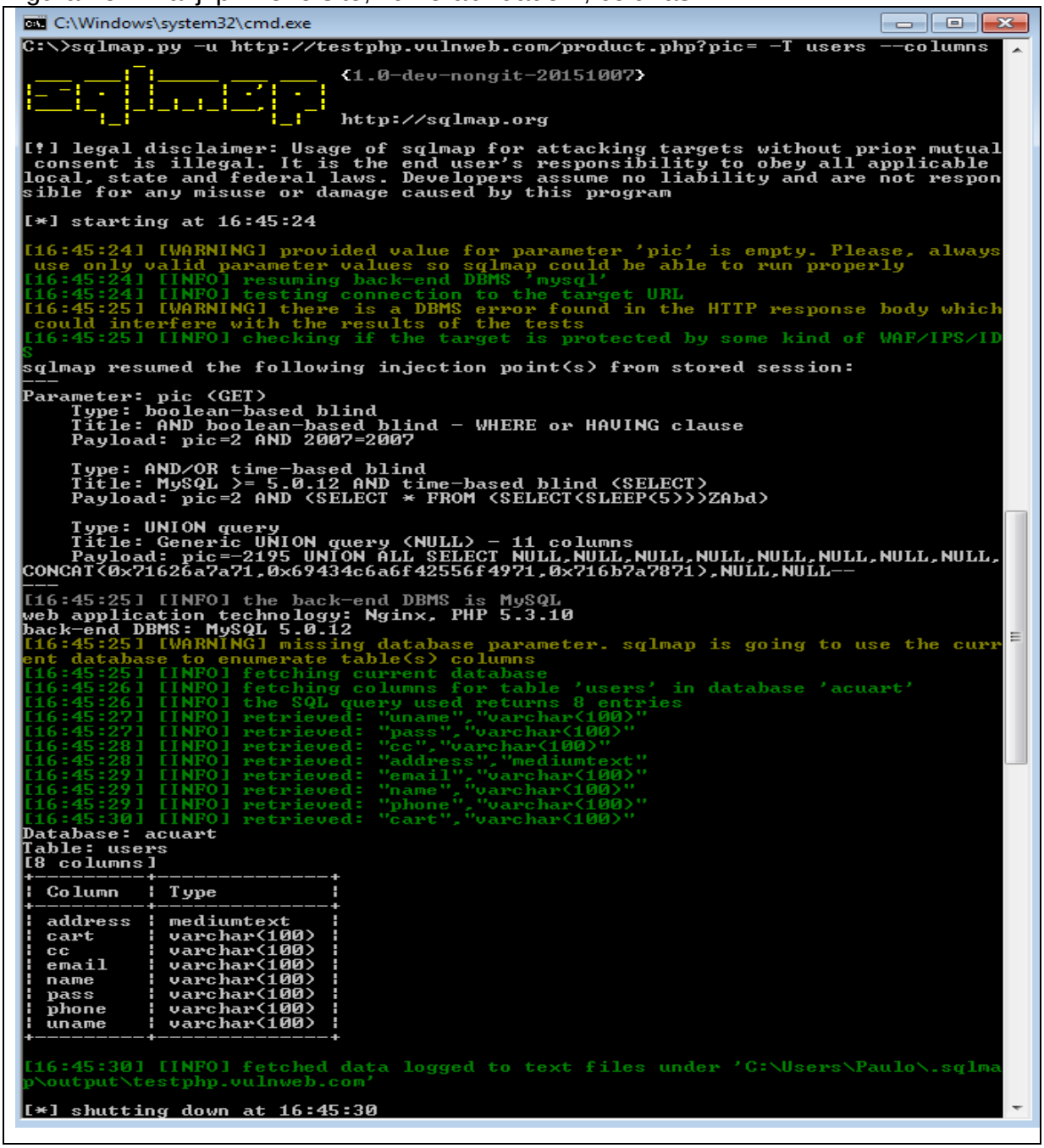
Descobrindo os bancos de dados e seguindo o padrão, foi executado mais um comando explorando o banco de dados acuart, com o intuito de conseguir acesso as suas tabelas. O ataque, retorna como resposta as mesmas oito tabelas encontradas pelo Havij, o que é ilustrado na Figura 44.

Figura 44 – SQLMap primeiro site, vulnerabilidade 2, tabelas.

Fonte: Elaborada pelo autor.

Já na Figura 45, prosseguindo com a invasão é feito novamente outro ataque, só que desta vez visando encontrar as colunas da tabela users, e assim são obtidas oito colunas iguais as encontradas pelo Havij.

Figura 45 – Havji primeiro site, vulnerabilidade 2, colunas.



Fonte: Elaborada pelo autor.

Por último é realizado um novo comando com o intuito de encontrar as informações contidas nas colunas da tabela users, como foi feito anteriormente realizado com a ferramenta Havij. Desta forma ao fim do ataque e após execução de vários testes realizado pelo SQLMap, é retornado uma linha de dados, que no caso são iguais ao obtidos pelo recurso anterior, como mostrado na Figura 46.

Figura 46 – Havji primeiro site, vulnerabilidade 2, dados.

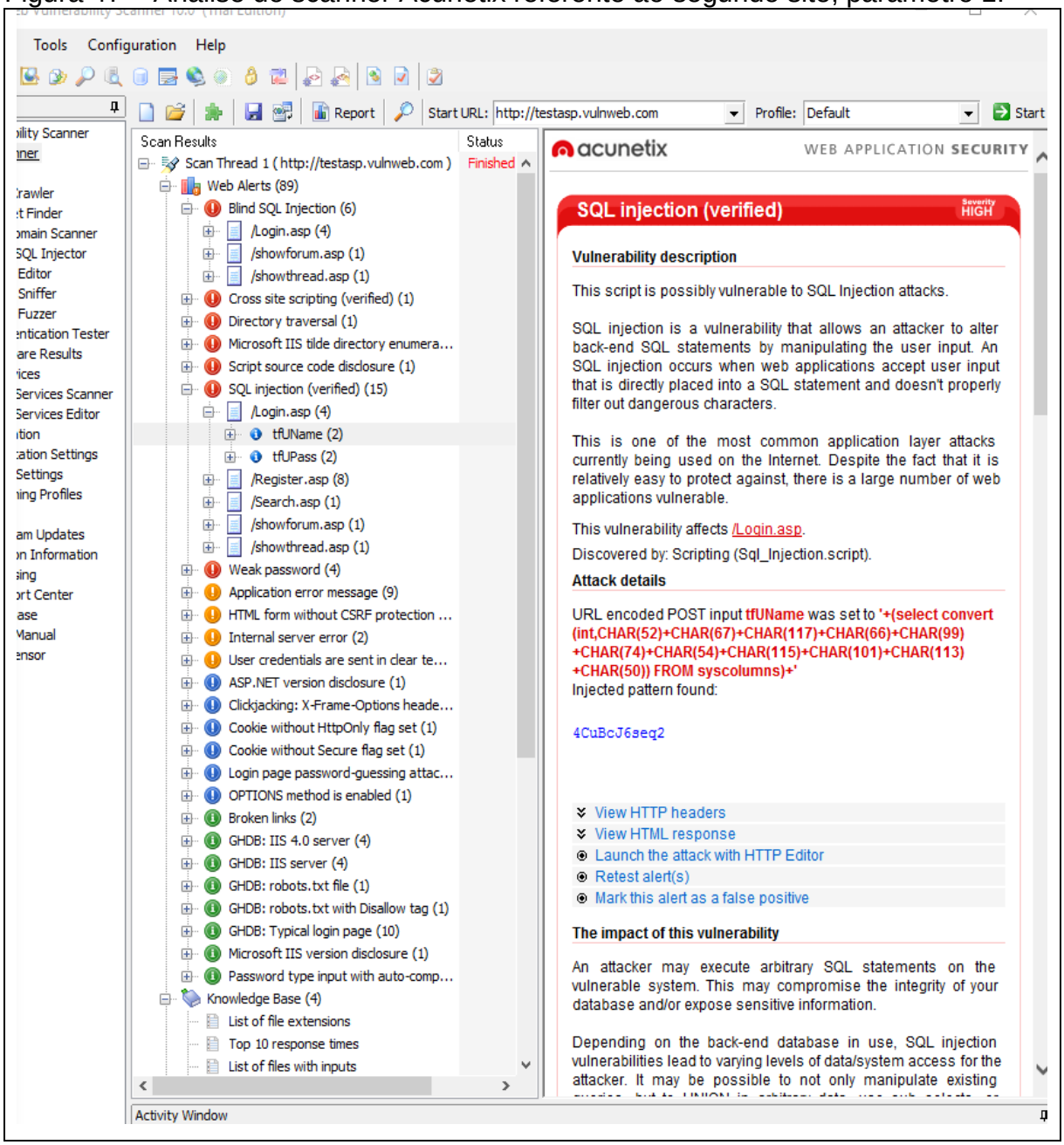
Fonte: Elaborada pelo autor.

Encerrando essa bateria de ataques realizados nas duas url do site http://testphp.vulnweb.com/, foi dado início ao mesmo procedimento mas no respectivo site, http://testasp.vulnweb.com/, e assim como anteriormente, esse foi analisado pelo scanner Acunetix para obtenção de suas vulnerabilidades.

O scanner por sua vez retornou diversas vulnerabilidades, no qual foram escolhidas aleatoriamente duas. A primeira vulnerabilidade escolhida foi a /Login.asp

com o parâmetro tfUName, que no caso é do método POST, como pode ser visto na Figura 47.

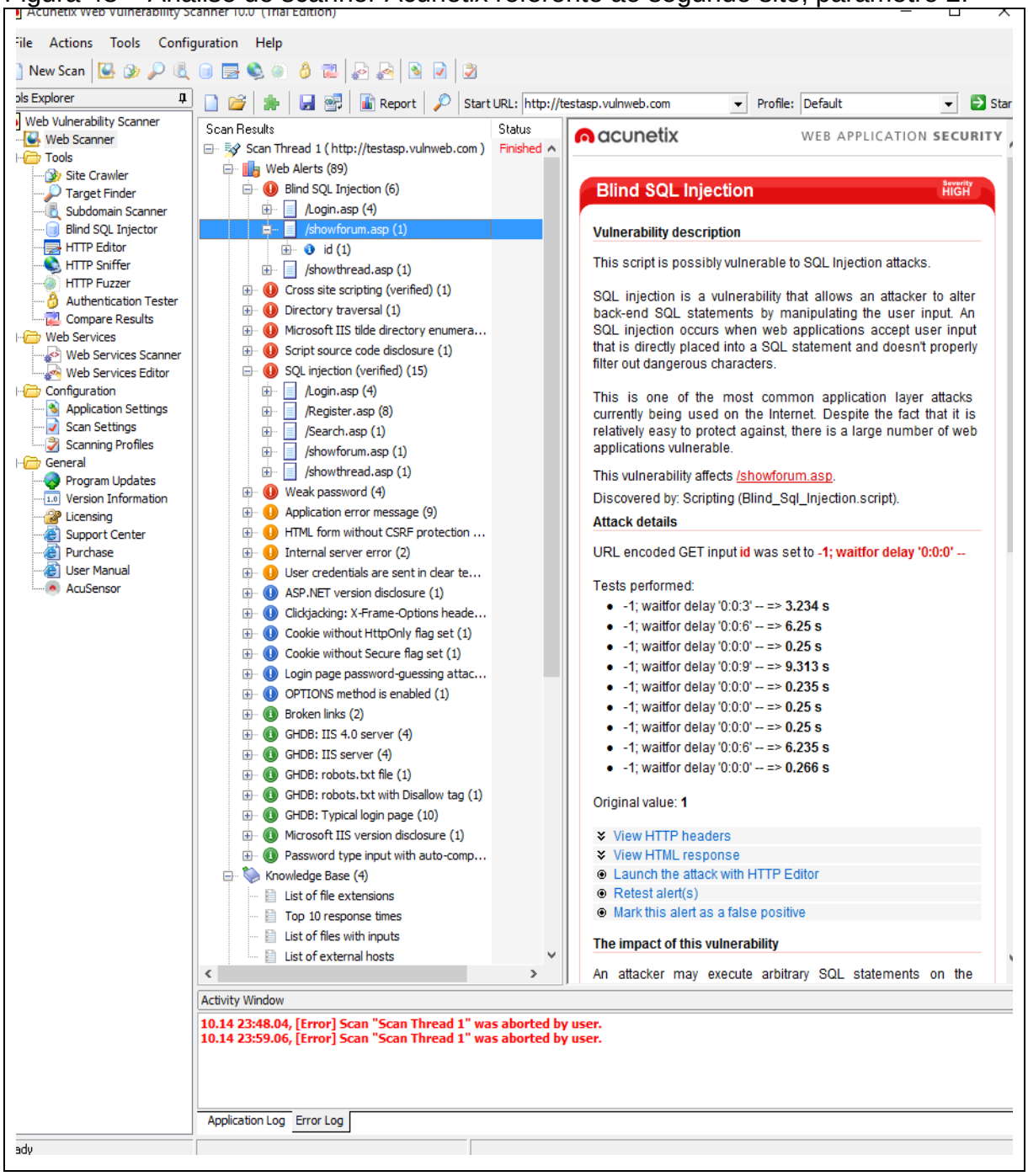
Figura 47 – Análise do scanner Acunetix referente ao segundo site, parâmetro 1.



Fonte: Elaborada pelo autor.

Já a segunda escolhida, pertencente ao métogo GET, foi a vulnerabilidade /showforum.asp que possui como parâmetro o id, que pode se observada na Figura 48.

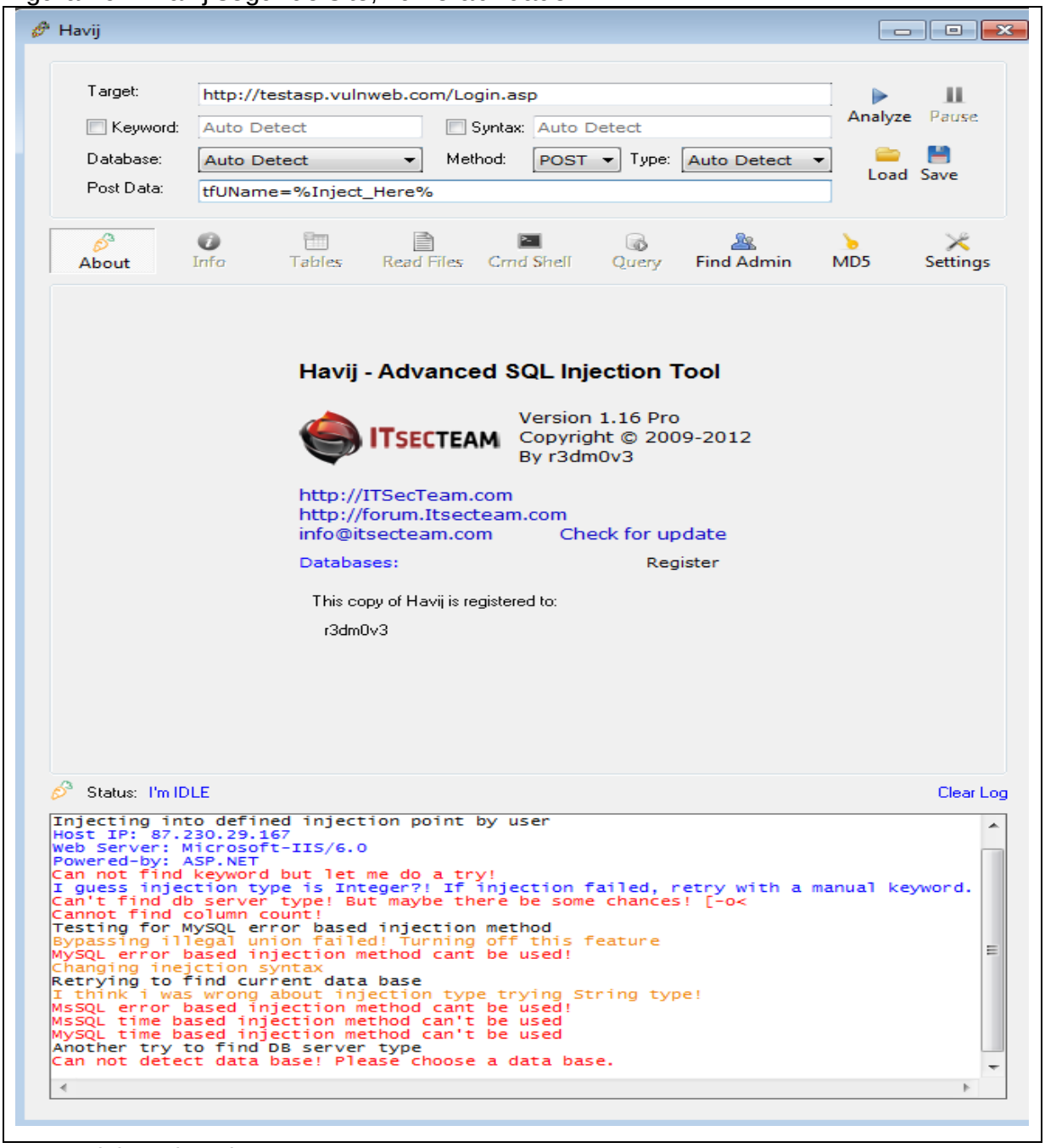
Figura 48 – Análise do scanner Acunetix referente ao segundo site, parâmetro 2.



Fonte: Elaborada pelo autor.

Após definidas as vulnerabilidades do site, que serão exploradas, o próximo passo foi iniciado. Começando pela ferramenta Havij, e do mesmo modo que anteriormente foram colocadas as informações em seus respectivos lugares para efetuar o ataque, mas como é possível ver na Figura 49, depois de alguns testes, o recurso não retorna nenhuma informação sobre o banco de dados do site alvo, e conclui que não é possível detectar nenhum BDs.

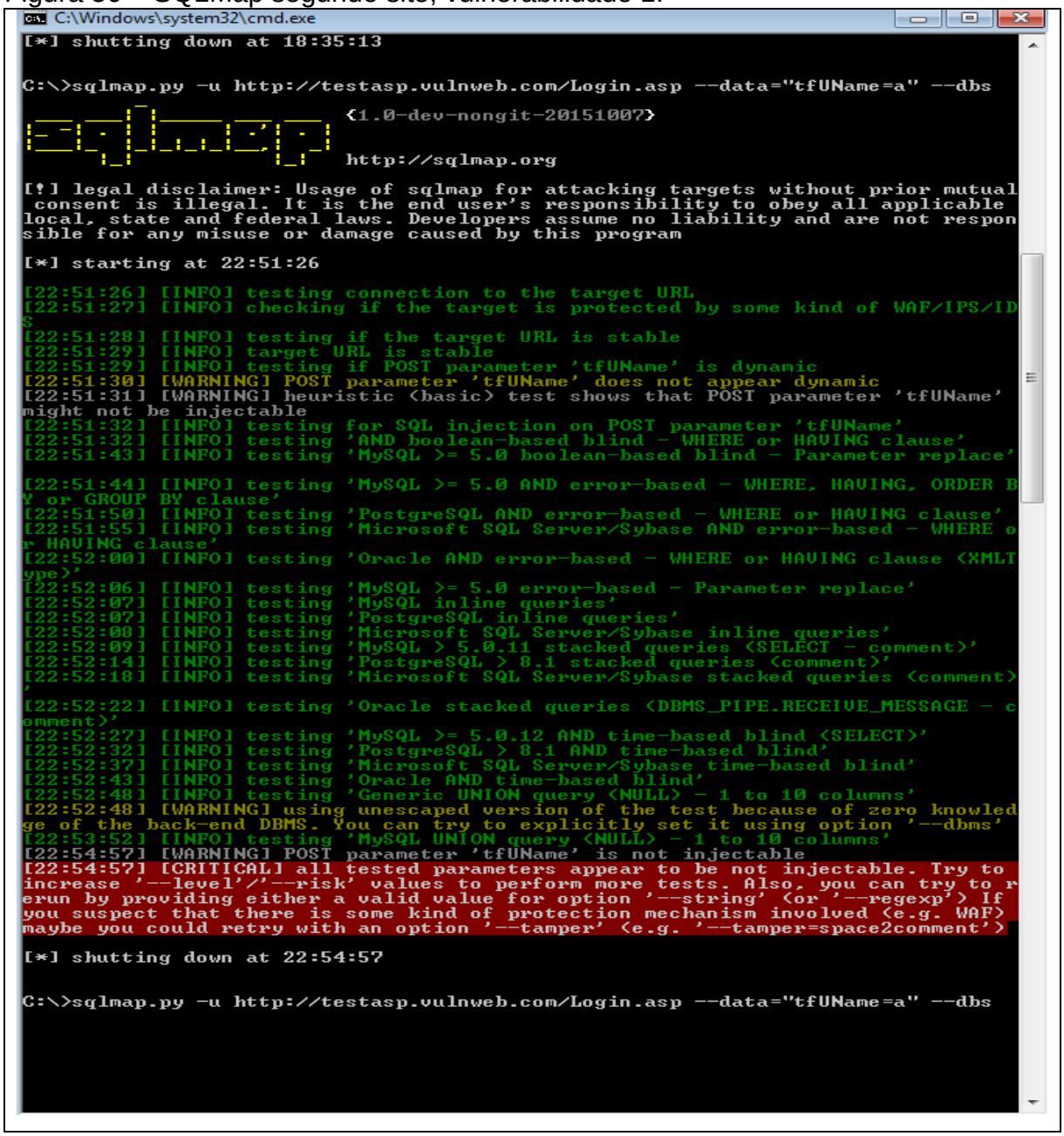
Figura 49 – Havij segundo site, vulnerabilidade 1.



Fonte: Elaborada pelo autor.

Diante da tentativa de ataque sem êxito feita utilizando o recurso Havij, o mesmo ataque foi realizado mas com a ferramenta SQLMap, e igualmente ao Havij, o SQLMap não consegue identificar nenhuma informação sobre o banco de dados alvo, mas diferentemente da ferramenta anterior, ilustrado na Figura 50, este recomenda refazer o ataque incrementando alguns recursos e dados a mais no comando, possibilitando talvez um ataque efetivo.

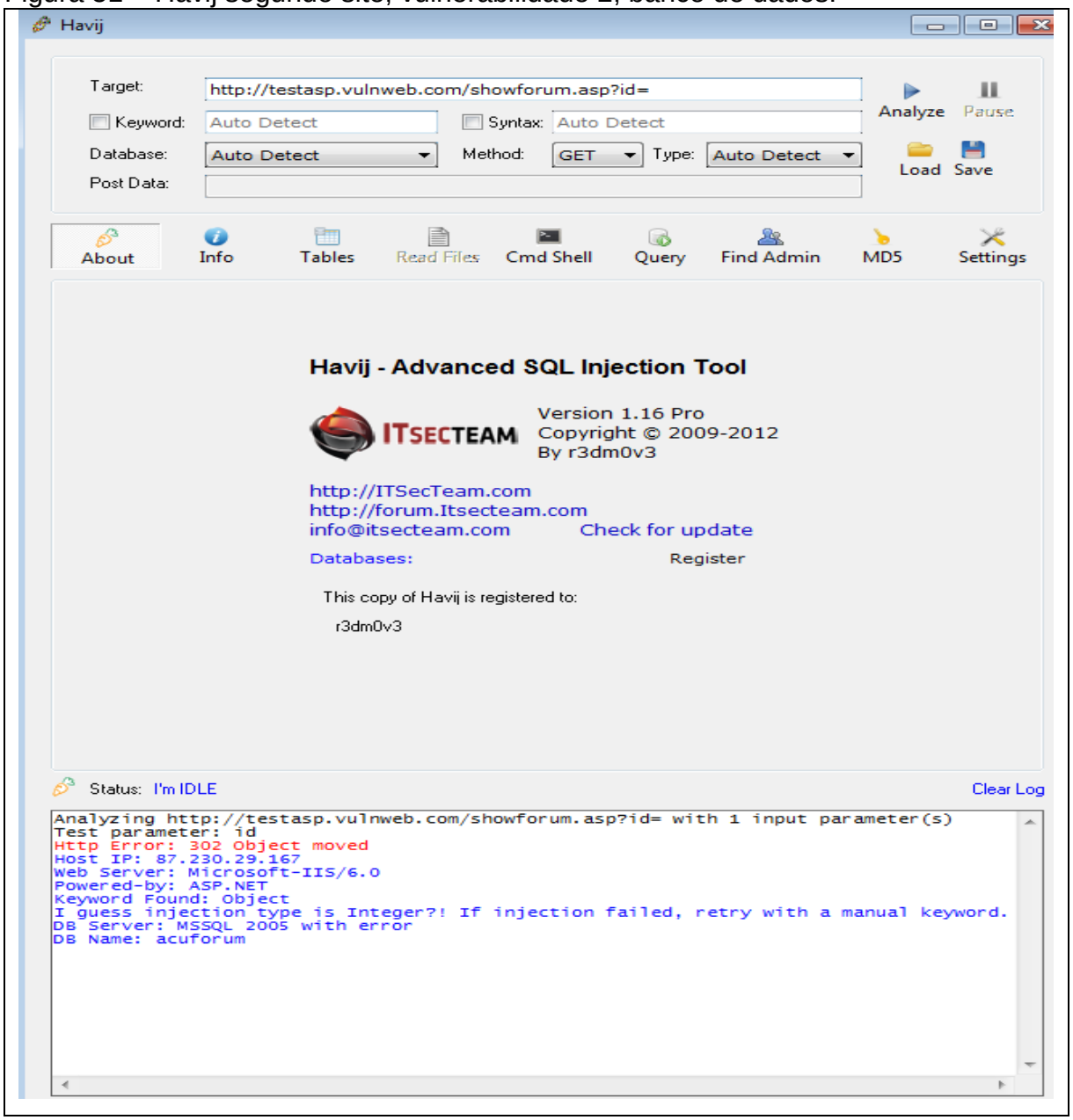
Figura 50 – SQLMap segundo site, vulnerabilidade 1.



Fonte: Elaborada pelo autor.

Desta forma foi dada continuidade ao processo, realizando um ataque com o recurso Havij, mas explorando a segunda vulnerabilidade deste site, /showforum.asp?id, como é ilustrado na Figura 51. Nesta foi encontrado o banco de dados acuforum.

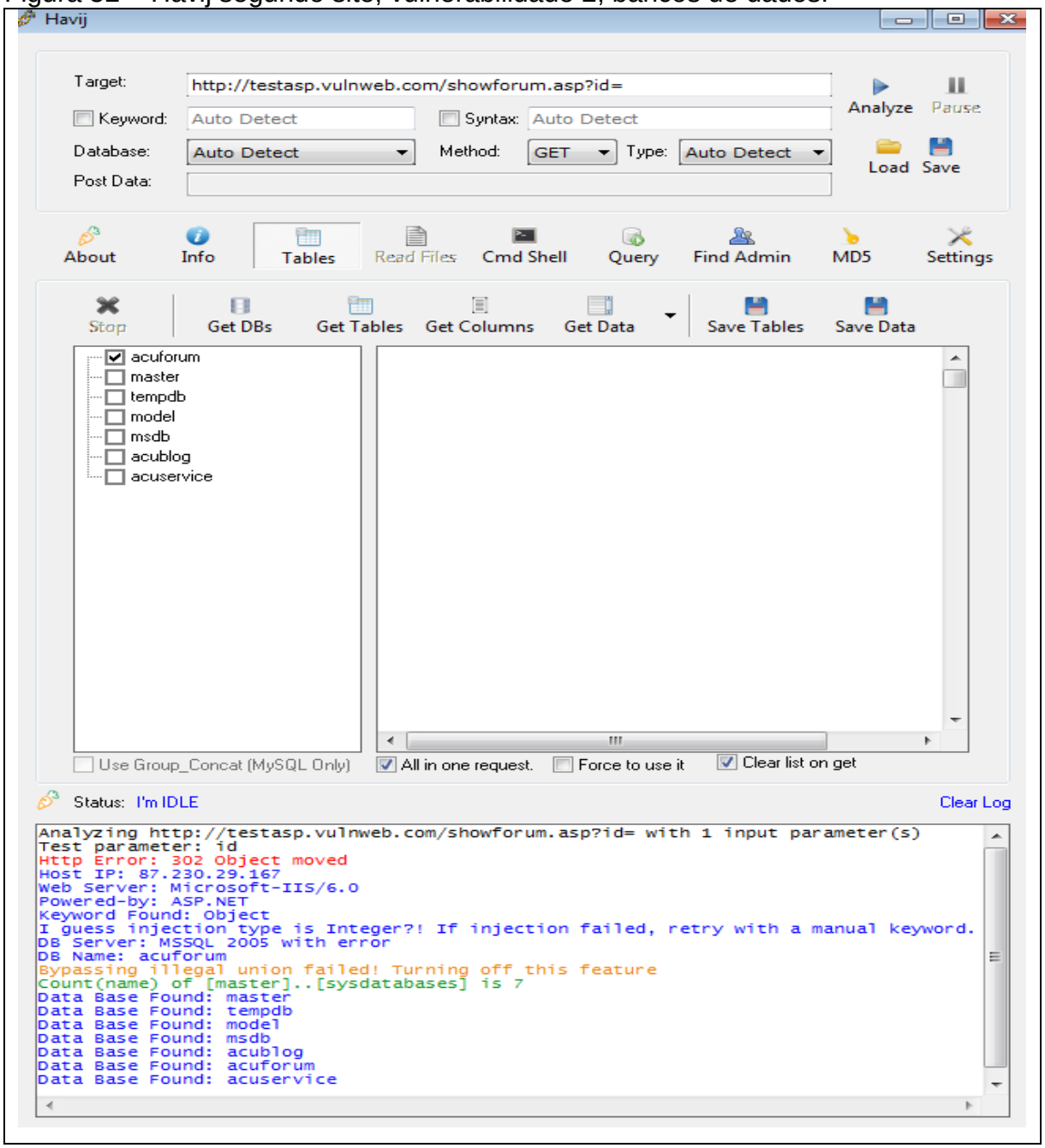
Figura 51 – Havij segundo site, vulnerabilidade 2, banco de dados.



Fonte: Elaborada pelo autor.

Prosseguindo com uma nova busca a procura de BDs, o software encontrou mais seis, master, tempdb, model, msdb, acublog, e acuservice, que são ilustrados na Figura 52.

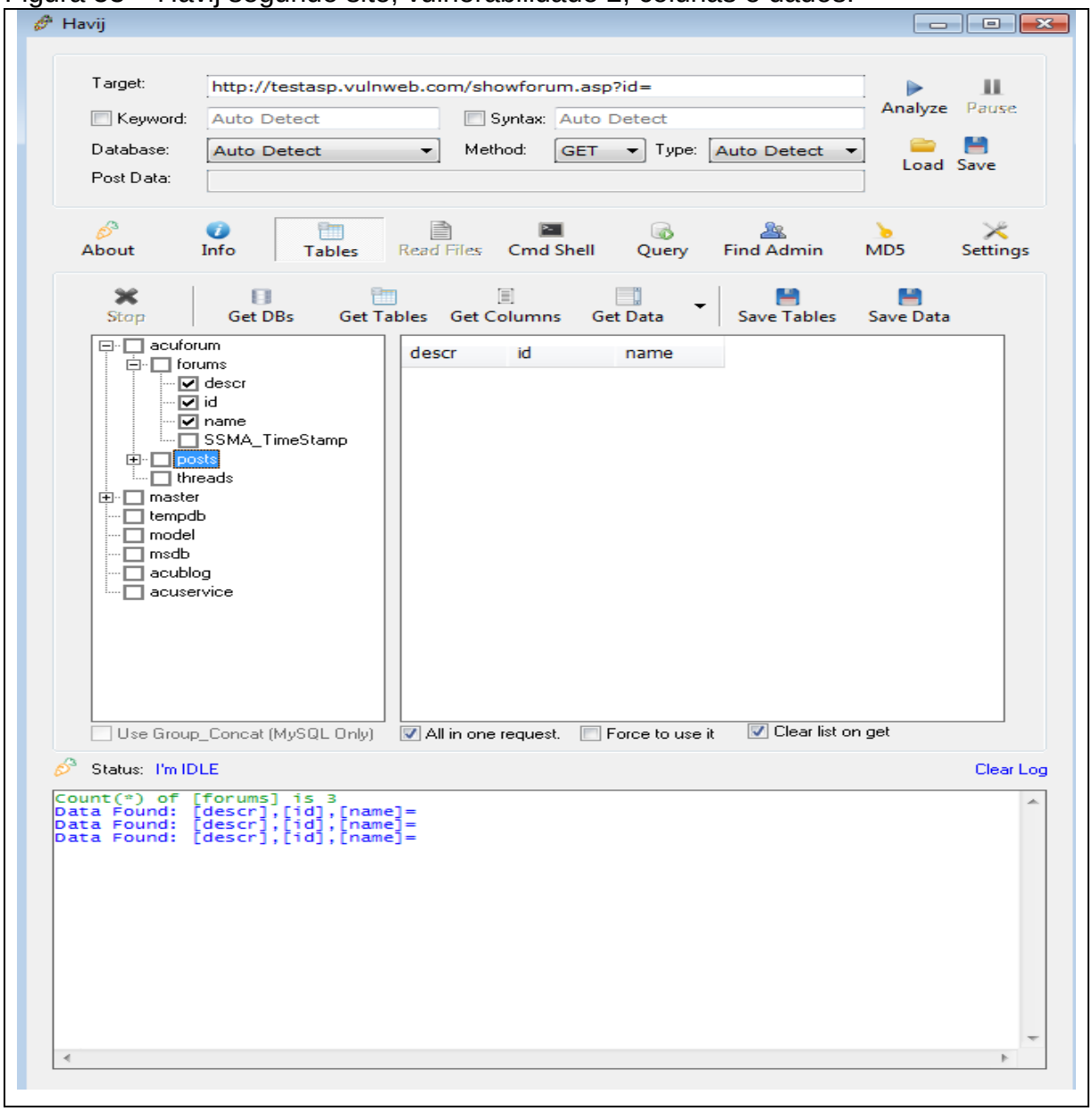
Figura 52 – Havij segundo site, vulnerabilidade 2, bancos de dados.



Fonte: Elaborada pelo autor.

Já na Figura 53 é ilustrado o resultado de mais dois ataques, no qual o primeiro teve intuito de explorar o BD acuforum, afim de encontrar tabelas, que no caso foram três, e o segundo ataque, teve objetivo de encontrar dados em uma destas tabelas, aleatoriamente foi escolhida a tabela forums e nenhum dados a ferramenta Havij retornou.

Figura 53 – Havij segundo site, vulnerabilidade 2, colunas e dados.



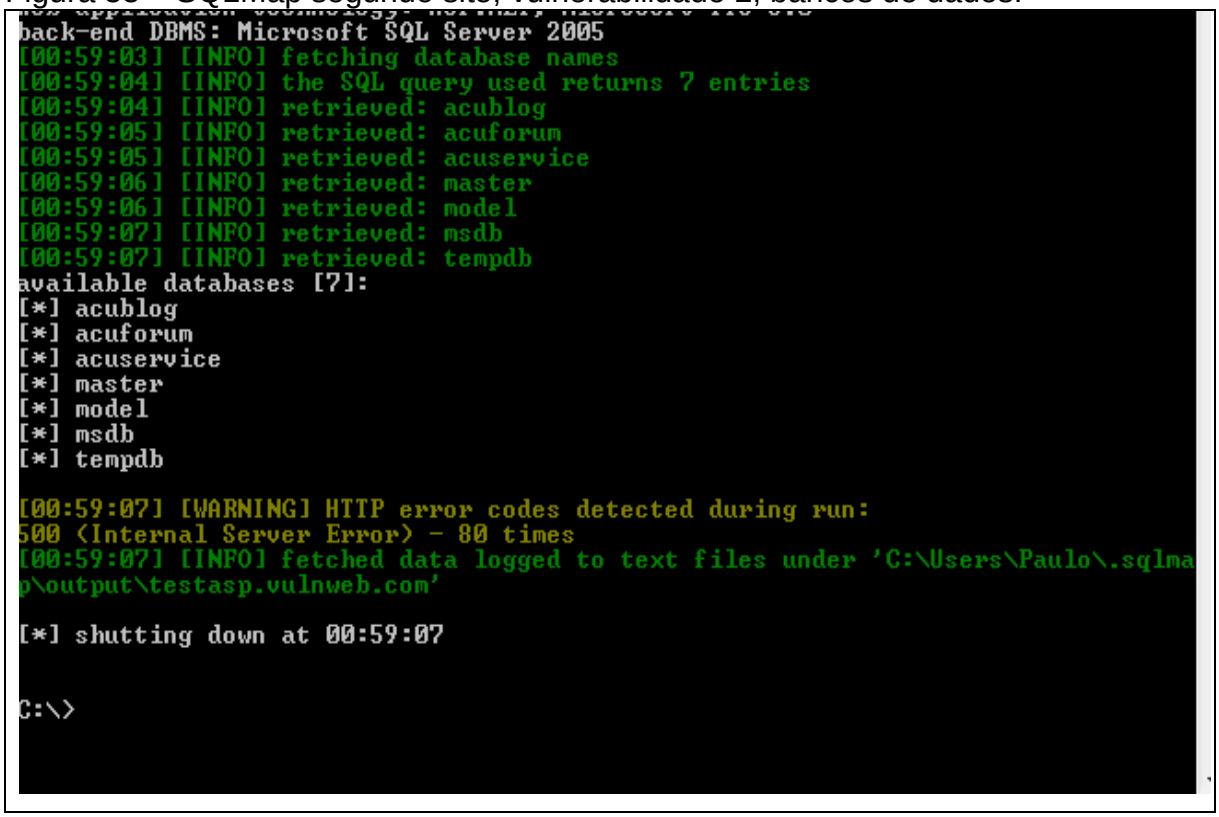
Fonte: Elaborada pelo autor.

Realizando o mesmo ataque com a ferramenta SQLMap, os resultados foram um pouco diferentes. No primeiro momento ao efetuar o ataque visando encontrar os bancos de dados, a ferramenta fez as mesmas três perguntas que já feitas anteriormente em outros ataques, e encontrou os mesmos 7 BDs, além do tipo de sistema de gerenciamento, Microsoft SQL Server 2005, como é ilustrado nas Figuras 54 e 55.

Figura 54 – SQLMap segundo site, vulnerabilidade 2.

Fonte: Elaborada pelo autor.

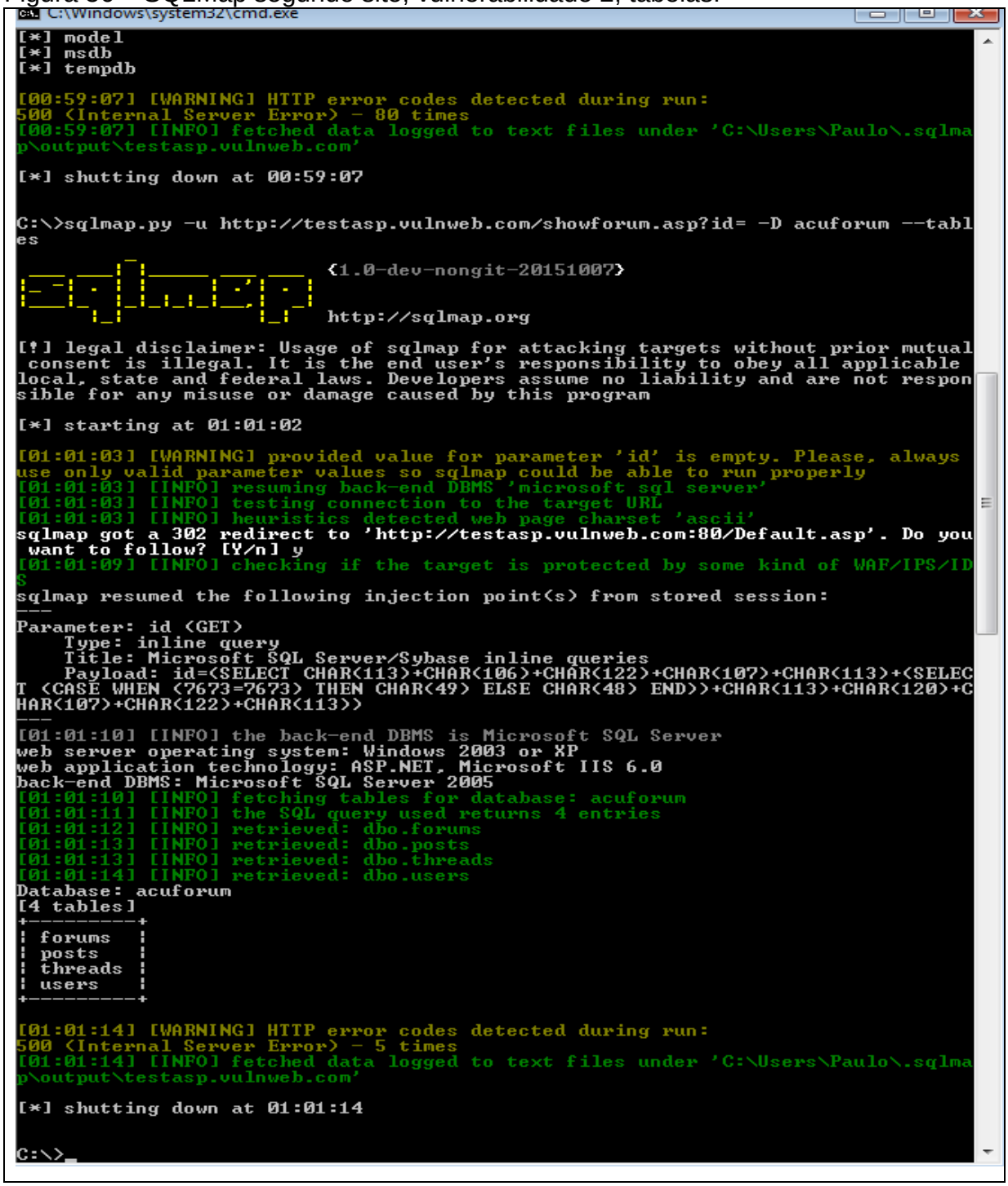
Figura 55 – SQLMap segundo site, vulnerabilidade 2, bancos de dados.



Fonte: Elaborada pelo autor.

Ao continuar com a bateria de ataques, e explorando o BD forums igualmente feito com o Havij, foram encontradas 4 tabelas, forums, posts, threads, e users. De forma que com essa ferramenta, o diferencial foi a tabela users, que não foi encontrada no recurso Havij, como pode ser visto na Figura 56.

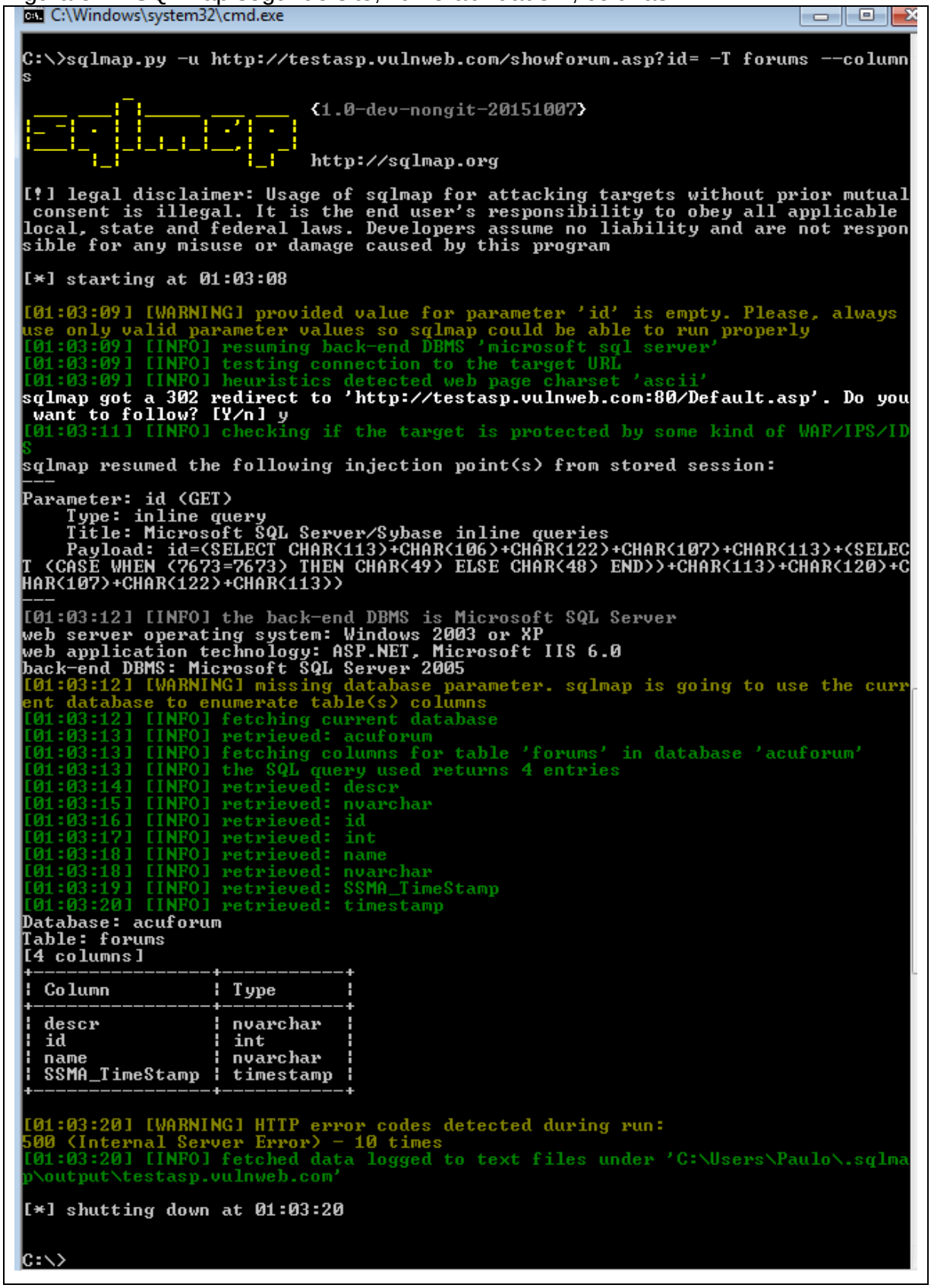
Figura 56 – SQLMap segundo site, vulnerabilidade 2, tabelas.



Fonte: Elaborada pelo autor.

Na Figura 57 é ilustrado, um ataque com o objetivo de descobrir as colunas da tabela forums. Neste são encontrados quatro, iguais as descobertas pelo Havij, descr, id, name, SSMA_TimeStamp.

Figura 57 – SQLMap segundo site, vulnerabilidade 2, colunas.



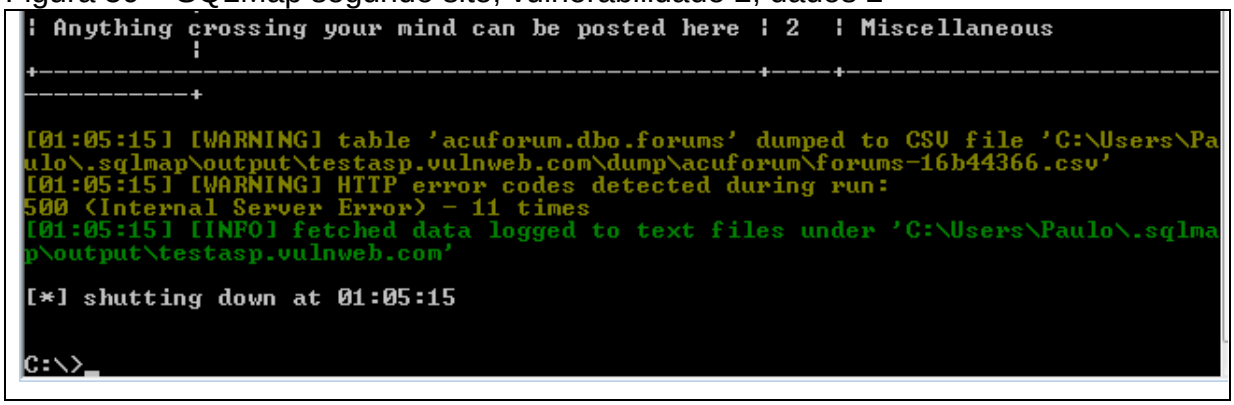
Fonte: Elaborada pelo autor.

De modo a explorar as colunas encontradas em um novo ataque, é executado um comando como pode ser visto na Figura 58 e 59, de modo que ao final dele foram encontradas 3 linhas de dados, o que não aconteceu com a ferramenta Havij, no qual não foram encontrados nenhuma informação.

Figura 58 – SQLMap segundo site, vulnerabilidade 2, dados

Fonte: Elaborada pelo autor.

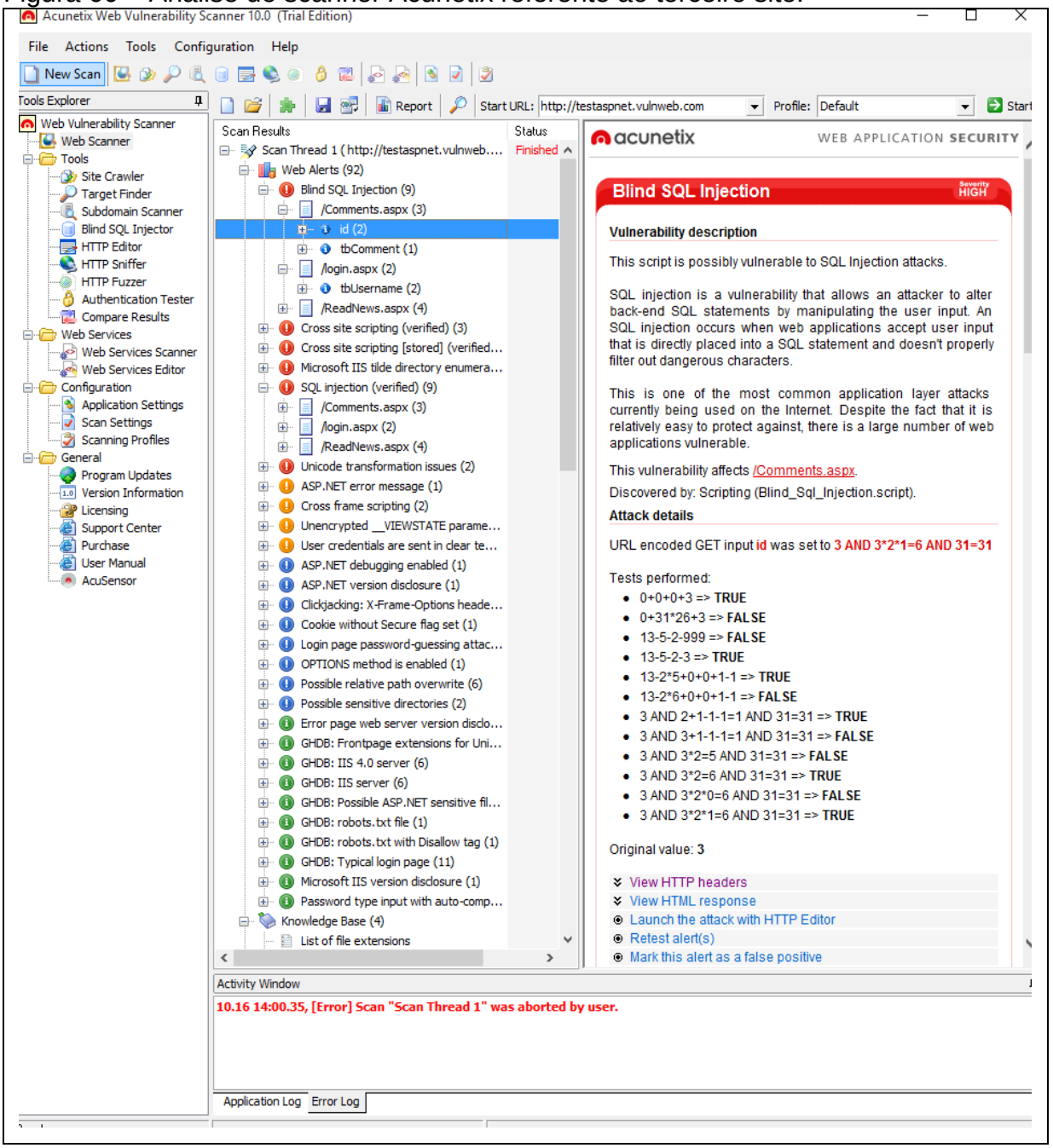
Figura 59 – SQLMap segundo site, vulnerabilidade 2, dados 2



Fonte: Elaborada pelo autor.

Para finalizar o processo, o último site escolhido foi atacado, http://testaspnet.vulnweb.com/. Este igualmente os outros foi explorado em suas duas vulnerabilidades escolhidas aleatoriamente, /comments.aspx?id e /login.aspx?tbUsername, no qual são respectivamente dos métodos GET e POST, como pode ser visto na Figura 60.

Figura 60 – Análise do scanner Acunetix referente ao terceiro site.

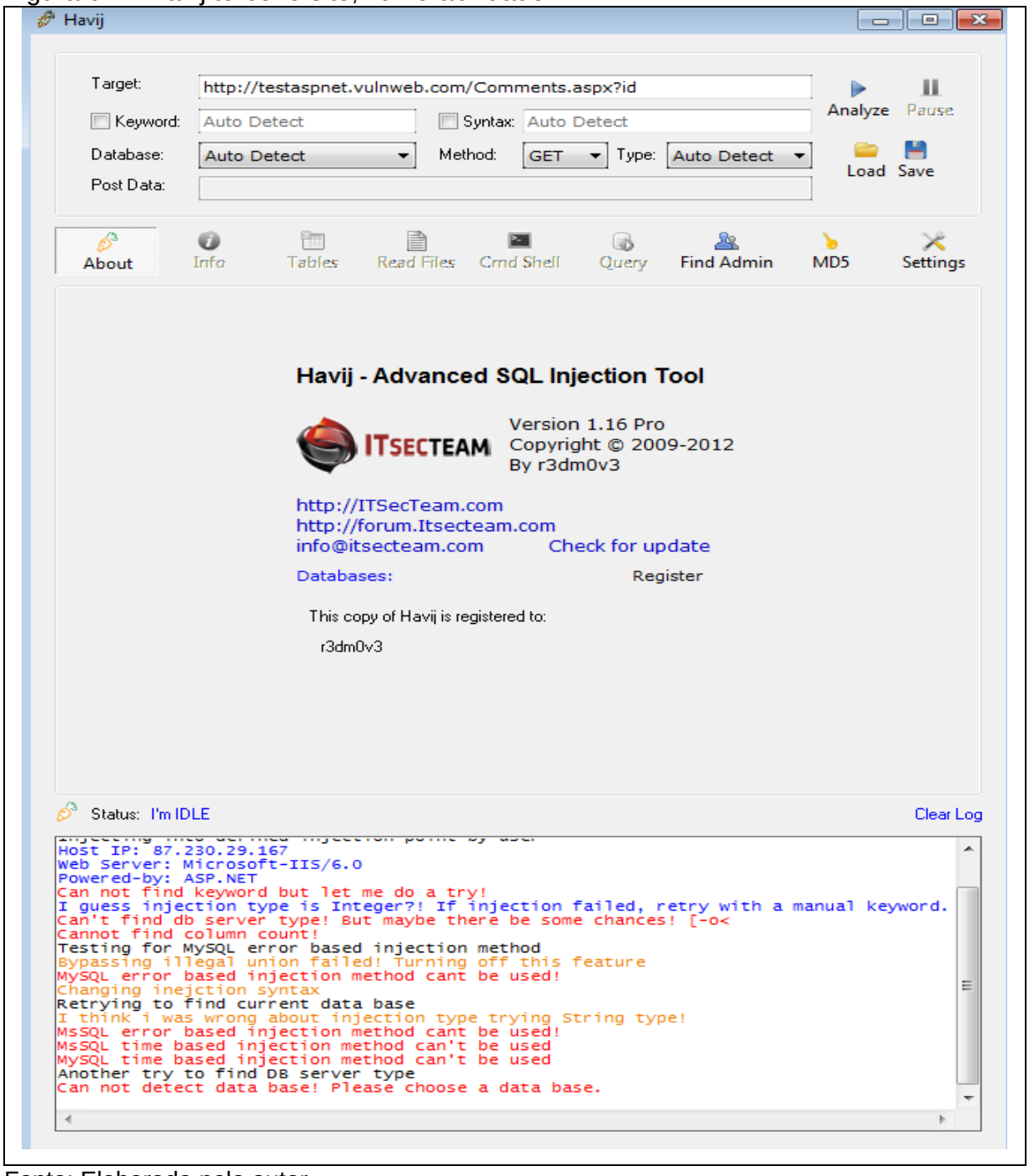


Fonte: Elaborada pelo autor.

A vulnerabilidade /Comments.aspx?id foi a primeira a ser testada, com a utilização da ferramenta Havij.

No recurso foram colocadas as informações em seus devidos lugares, e iniciado o ataque, de modo que ao chegar em seu término, o Havij informou que não foi possível detectar o banco de dados, não fornecendo nenhuma informação referente a tal, o que é ilustrado na Figura 61.

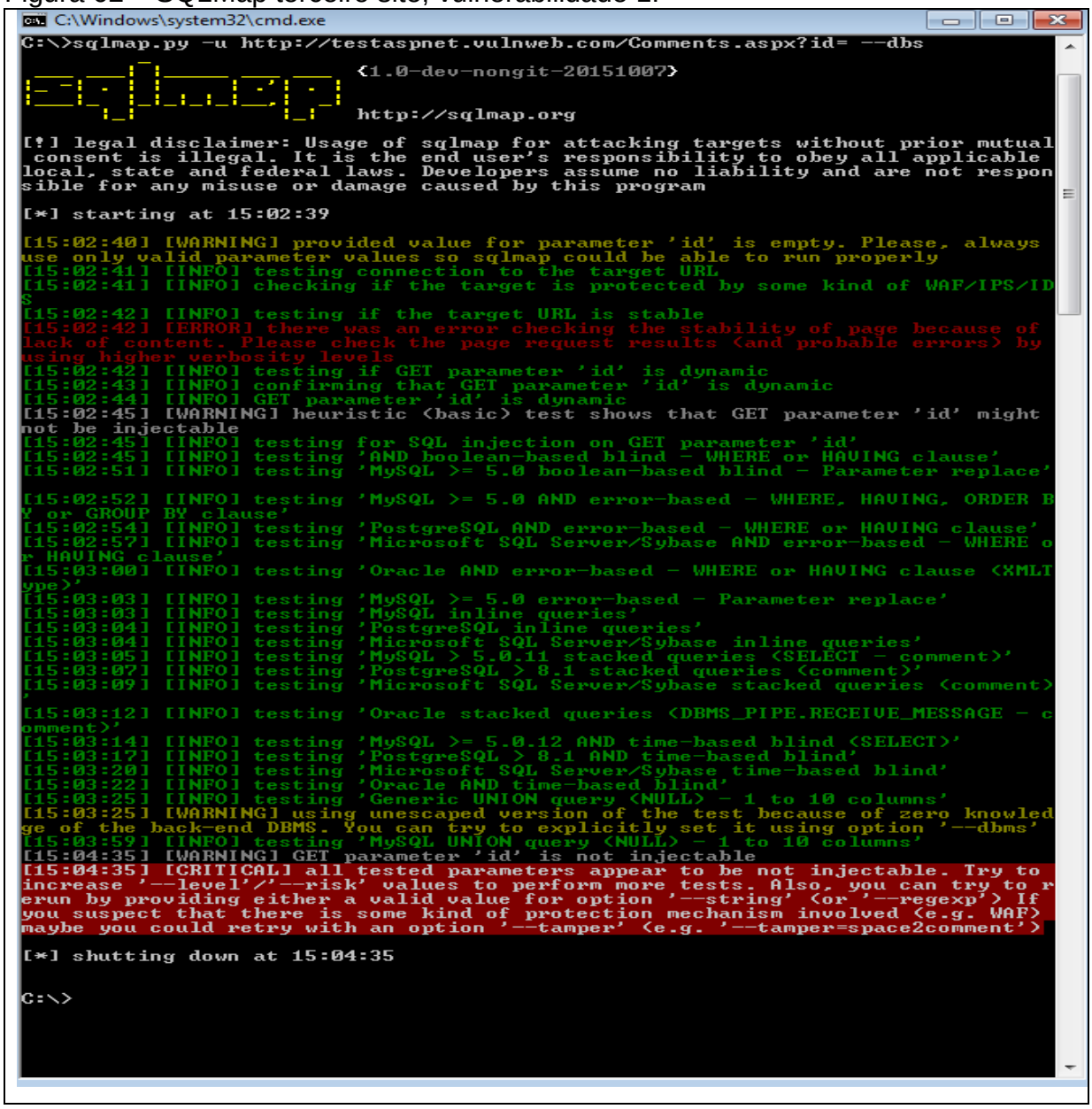
Figura 61 – Havij terceiro site, vulnerabilidade 1.



Fonte: Elaborada pelo autor.

De forma que ao elaborar o mesmo ataque, com o intuito de descobrir o BDs, com a ferramenta SQLMap, o resultado foi o mesmo, como pode ser visto na Figura 62, mas como anteriormente ela menciona a possibilidade de incrementar e melhorar o comando executado com outras técnicas possibilitando talvez um ataque bem sucedido posteriormente.

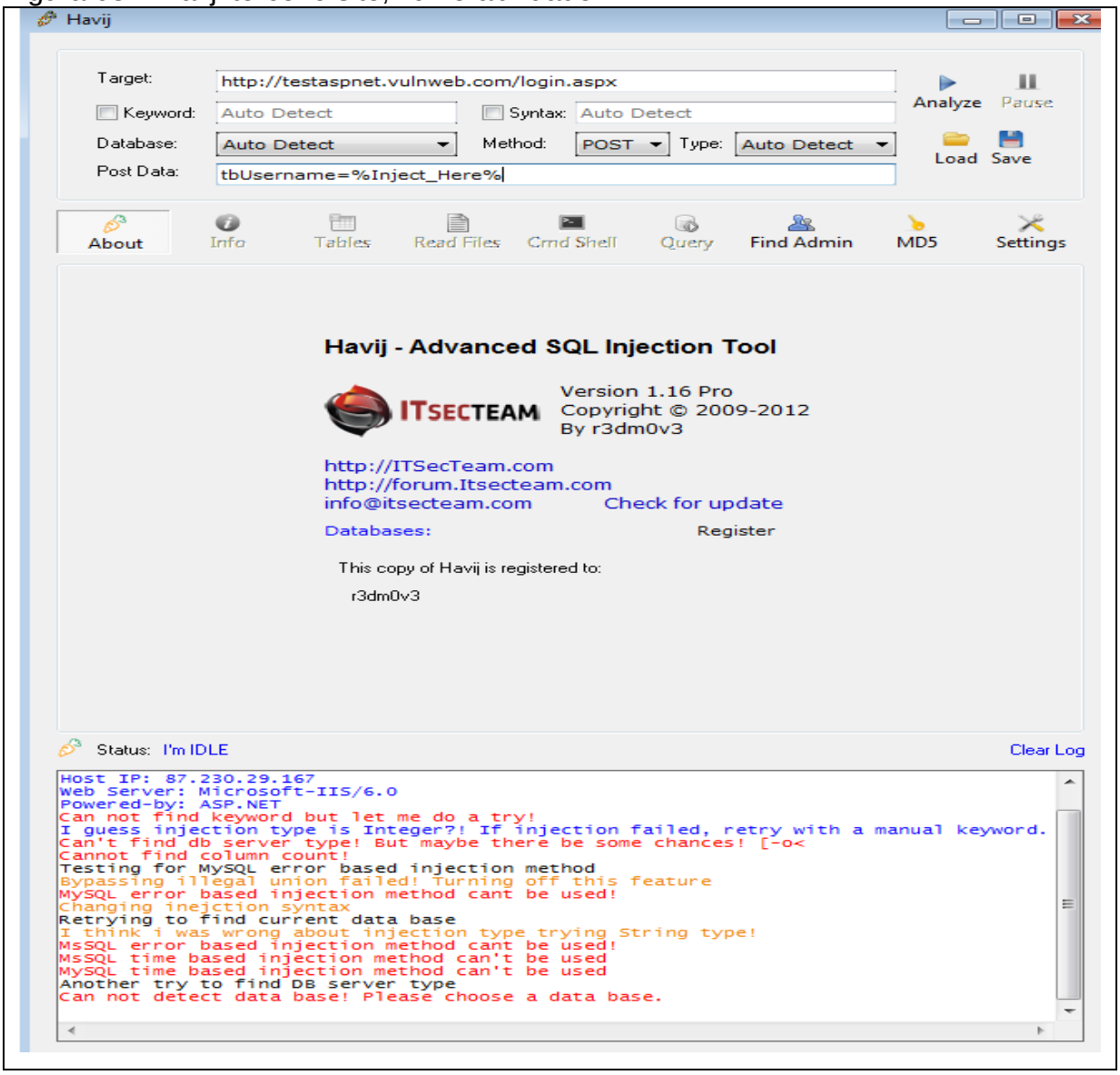
Figura 62 – SQLMap terceiro site, vulnerabilidade 1.



Fonte: Elaborada pelo autor.

Por vez foi explorado a última vulnerabilidade deste site, /login.apx?tbusername, do método POST. Como padrão foi iniciado o ataque no programa Havij, que realizou certos testes, e por último identicamente ao ataque feito na vulnerabilidade anterior, a ferramenta não conseguiu identificar informações sobre o banco de dados, como pode ser visto na Figura 63.

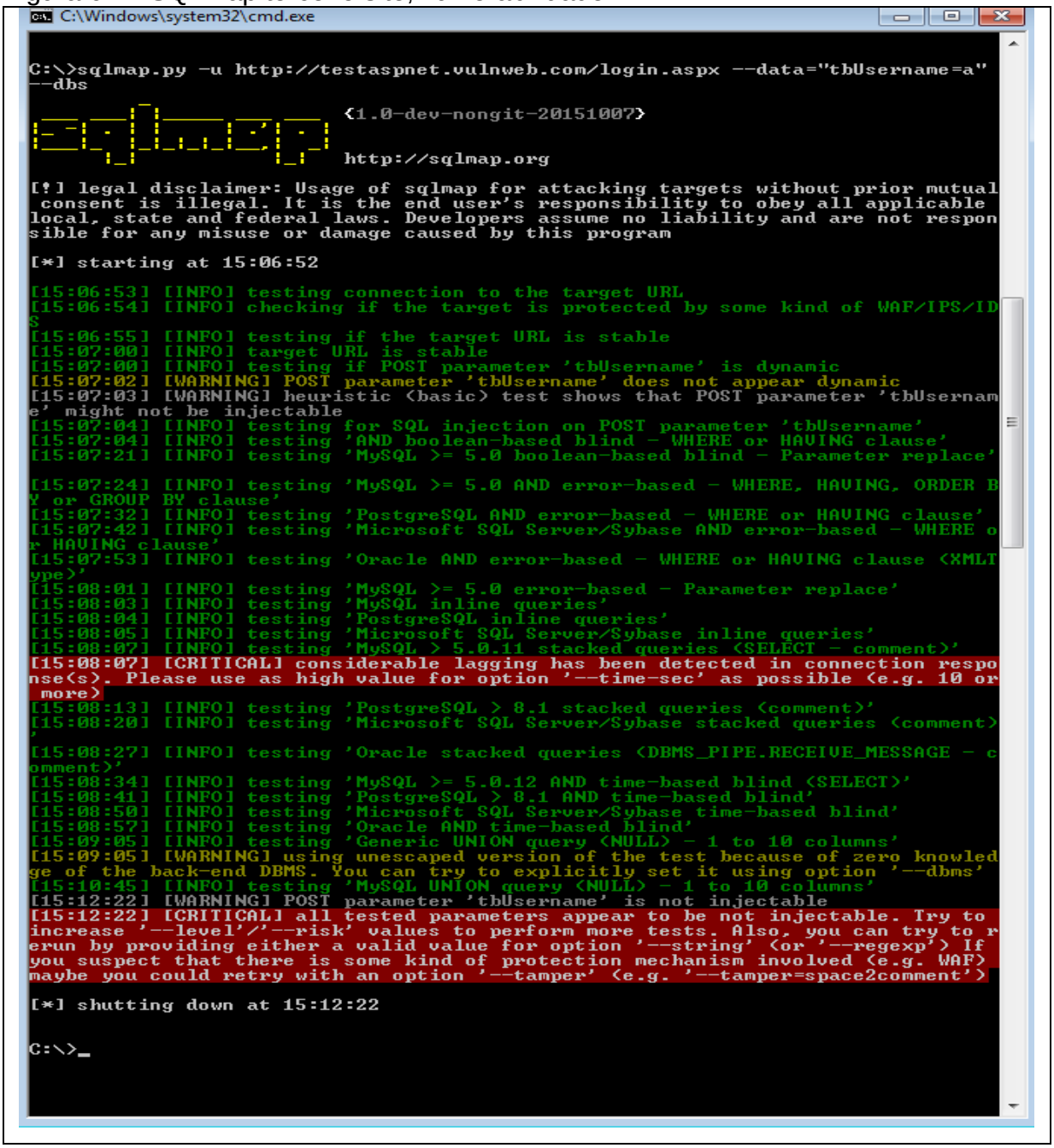
Figura 63 – Havji terceiro site, vulnerabilidade 2.



Fonte: Elaborada pelo autor.

Já no ataque realizado com a ferramenta SQL, o resultado não foi diferente, ao executar o comando, que possuia o objetivo de encontrar o BDs, o recurso também fez uma quantidade significativa de testes mas não obteve êxito em tais tentativas, como ilustrado na Figura 64.

Figura 64 – SQLMap terceiro site, vulnerabilidade 2.



Fonte: Elaborada pelo autor.

Assim ao fim deste ataque, foi encerrado as tentativas de invasão nestes 3 sites que foram escolhidos, nos quais foi possível observar algumas diferenças entre as duas ferramentas e seus ataques.

Para uma visualização mais evidente sobre as informações obtidas através dos ataques, a Figura 65 ilustra uma tabela desenvolvida a partir dos dados fornecidos pelas tentativas de invasão.

Figura 65 – Tabela de Resultados.

		Método	Ferramenta	SGBD	BDs	Tabelas	Colunas	Dados	Tempo
SITE 1	Vul 1	GET	SQLMap	MySQL	2	8	8	1 linha	74s
			Havij	MySQL	2	8	8	1 linha	23s
	Vul 2	POST	SQLMap	MySQL	2	8	8	1 linha	94s
			Havij	MySQL	2	8	8	1 linha	22s
SITE 2	Vul 1	POST	SQLMap	-	-	-	-	-	211s
			Havij	-	-	-	-	-	88s
	Vul 2	GET	SQLMap	Microsoft SQL Server	7	4	4	3 linhas	125s
			Havij	-	7	3	4	-	20s
SITE 3	Vul 1	GET	SQLMap	-	-	-	-	-	116s
			Havij	-	-	-	-	-	86s
	Vul 2	POST	SQLMap	-	-	-	-	-	330s
			Havij	-	-	-	-	-	138s

Fonte: Elaborada pelo autor.

Na Figura 65, os nomes dos sites e das vulnerabilidades foram enumerados para auxiliar na estrutura e entendimento da tabela, logo o site 1 corresponde ao testphp.vulnweb.com, site 2 ao testasp.vulnweb.com e site 3 ao site testaspnet.vulnweb.com. Já as vul correspondem as vulnerabilidades de cada site, no qual a vul 1 e 2 do site 1 são referentes respectivamente ao product.php?pic e infotitle.php?id, vul 1 e 2 do site 2, pertencem respectivamente a login.asp?tfUName e a showforum.asp?id, e por útlimo as vul 1 e 2 do site 3, condizem a comments.aspx?id e login.aspx?tbUsername.

Ao analisar essa tabela, pode-se perceber facilmente que vários ataque foram bem sucedidos, mas por outro lado uma grande parte deles também não obtiveram êxito. Considerando que o scanner Acunetix está 100% correto em suas análises e que as vulnerabilidades escolhidas são realmente vulneráveis, por a empresa Acunetix ser a desenvolvedora dos sites utilizados, é possível afirmar que 50% dos ataques efetuados tiveram êxito. Destes 50% de invasões bem sucedidas 66.6% foram em sistemas que utilizavam os métodos GET e 33.3% em métodos POST, logo os ataques sem sucesso obtiveram uma porcentagem de 50% igualmente, dentre os quais 33.3% foram do método GET e 66.6% do método POST. Diante desses fatos, igualmente abordado na primeira parte deste trabalho por Kennedy (2002), os dois métodos possuem suas falhas, mas quando a segurança for visada, é aconselhável o método POST.

Pode-se concluir do mesmo modo que as ferramentas possuem algumas diferenças. Primeiramente é possível observar que a ferramenta Havij é significativamente mais rápida que a ferramenta SQLMap, devido ao Havij ter sido mais veloz em todos os ataques realizados. Em contrapartida, é possível observar algumas inconsistências desse software, de maneira que ao comparar as informações obtidas com a ferramenta SQLMap, é encontrado algumas divergências.

Nota-se que em grande parte dos ataques, as duas ferramentas obteram os mesmos resultados, somente em um caso foi diferente, no site 2, vulnerabilidade 2. Neste o SQLMap por sua vez encontrou mais tabelas, além de identificar o tipo de sistema de gerenciamento de banco de dados e encontrar dados nas colunas das tabelas, já com o recurso Havij, não foi possível encontrar nenhum dado contido nas colunas das tabelas e muito menos o SGBD.

De forma que todos os ataques realizados nos três sites foram efetuados de uma maneira simples, logo não estavam utilizando 100% da capacidade das ferramentas, mas visivelmente o SQLMap por diversos momentos no proporcionou opções de realizar buscas mais aprofundadas, além das quais ele pode realizar com um nível mais elevado de conhecimento em relação a ferramenta.

6 CONSIDERAÇÕES FINAIS

Com base no que foi apresentado, este trabalho manteve seu foco em informar e alertar, desenvolvedores e demais interessados, exemplificando meios de invasão com utilização da técnica SQL Injection. Diante disto, e a partir dos resultados obtidos, foi possível observar que existem diversos tipos de ferramentas que automatizam e auxiliam em um ataque, mesmo que possam ser incompletas, falhas, intuitivas ou não. De maneira que além desses recursos, ainda é possível, com o conhecimento de alto nível atacar sites de uma maneira muito provavelmente superior as apresentadas, mostrando a todos, que a técnica baseada no SQL Injection, que pode ser realizada de diversos modos, é apenas uma dentre tantas outras tipos técnicas de ataques que existem que na internet.

Como recomendação para trabalhos futuros, sugere-se um estudo das diversas técnicas baseadas no code injection, além de análises comparativas referente as outras ferramentas que existem para automatizar esse tipo de ataque.

REFÊRENCIAS

ACUNETIX, **Acunetix – Combating the Web Security Threat**, 2015. Disponível em: <https://www.acunetix.com/>. Acesso em: 25 de set. 2015.

ATHANASOPOULOS, Elias; PAPPAS, Vasilis; MARKATOS, Evangelos P. Code-Injection Attacks in Browsers Supporting Policies. **Computer Science**, 2009. Disponível em: <http://www.cs.columbia.edu/~vpappas/papers/code_injection.w2sp09.pdf>. Acesso em: 06 maio 2015.

BEZERRA, Luciano et al. Ataque a Computadores. **Instituto de Computação – UFF**, [2015?]. Disponível em: <http://www2.ic.uff.br/~otton/graduacao/informaticaI/Ataques.pdf> . Acesso em: 10 maio 2015.

BOSCARIOLI, Clodis. Uma reflexão sobre banco de dados orientados a objetos. In: CONGRESSO DE TECNOLOGIA PARA A GESTÃO DE DADOS E METADADOS DO CONE SUL, 4., 2006, Ponta Grossa. **Anais...** Ponta Grossa: UEPG, 2006. p. [1-12]. Disponível em: <http://conged.deinfo.uepg.br/artigo4.pdf>. Acesso em: 30 mar. 2015.

CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL (CERT). Estatísticas dos Incidentes Reportados ao CERT.br. **CERT**, 2014. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 06 maio 2015.

CLARKE, Justin. **SQL injection attacks and defense**. Burlington: Syngress Publishing, 2009. Diponível em: <http://adrem.ua.ac.be/sites/adrem.ua.ac.be/files/sqlinjbook.pdf>. Acesso em: 13 abr. 2015.

DAMAS, Luís. **SQL Structured Query Language**. 6. ed. atual. rev. Rio de Janeiro: Livros Técnicos e Científicos, 2007.

FILHO, Antônio Mendes da Silva. Segurança da Informação: Sobre a Necessidade de Proteção de Sistemas de Informações**. Revista Espaço Acadêmico**, Maringá, n. 42, nov. 2004. Disponível em: <http://www.espacoacademico.com.br/042/42amsf.htm>. Acesso em: 12 maio 2015.

GATTO, Raquel F.; MOREIRAS Antonio M.; GETSCHKO, Demi. Governança da Internet: conceitos, evolução e abrangência. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 27. 2009, UFMG. **Anais...** Belo Horizonte: UFMG, 2009. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbrc/2009/081.pdf>. Acesso em: 26 abr. 2015.

GIL, Antônio Carlos. **Como Elaborar Projetos de Pesquisa**, Altas, 3 edição, 1991. Disponível em: < https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=21&cad=rj

a&uact=8&ved=0CBwQFjAAOBQ&url=https%3A%2F%2Fxa.yimg.com%2Fkq%2Fgroups%2F19704728%2F225686590%2Fname%2F13259802-Como-Elaborar-Projetos-de-Pesquisa-ANTONIO-CARLOS-GIL-Editora-Atlas.pdf&ei=nh1WVeSRAojIsATP44CADg&usg=AFQjCNEt1rG-igGUJq2WtsukajtsGJGp5g&sig2=5illtEcDIcIBE0FQFcKYsw >. Acesso em : 15/05/2015.

GILLENSON, Mark L. et al. **Introdução à gerência de banco de dados**. Rio de Janeiro: Livros Técnicos e Científicos, 2009.

GOURLEY, David et al. **HTTP**: the definitive guide. Sebastopol: O'Reilly, 2002. Disponível em: <https://books.google.com.br/books?id=qEoOl9bcV_cC&pg=PT657&lpg=PT657&dq=HTTP:+The+Definitive+Guide&source=bl&ots=zxoRTuOB6i&sig=5720cZ3hYozo4GjFH-CbEdFBLwQ&hl=pt-BR&sa=X&ei=L3I-VZ69O8a2yAS51IDIAg&ved=0CEkQ6AEwBQ#v=onepage&q=HTTP%3A%20The%20Definitive%20Guide&f=false> . Acesso em: 27 abr. 2015.

HALFOND, William G. J., VIEGAS, Jeremy, ORSO, Alessandro. **A Classification of SQL Injection Attacks and Countermeasures**. College of Computing Georgia Institute of Technology, 2006. Diponível em: <http://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf>. Acesso em: 09 de maio 2015.

IEPSEN, Edécio Fernando. Segurança de Sistemas, **FACULDADE DE TECNOLOGIA SENAC PELOTAS**, Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, [2015?]. Disponível em: <http://187.7.106.14/edecio/seg/tudo_textos.pdf>. Acesso em: 06/05/2015.

INTERNET users. **Internet Live Stats**, 2015. Disponível em : <http://www.internetlivestats.com/internet-users/> Acesso em: 04 de maio 2015.

ISO/IEC 9075-14:2006. Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML), International Organization for Standardization, 2006. Disponível em: <**http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=38647**>. Acesso em: 4 abril 2015.

ISO/IEC 9075-3:2008. Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI), International Organization for Standardization, 2008. Disponível em: <**http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38641**>. Acesso em: 4 abril 2015.

KENNEDY, Bill, MUSCIANO, Chuck. **HTML & XHTML: The Definitive Guide**, 5th Edition. Sebastopol: O'Reilly, 2002. . Disponível em : https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CDAQFjAC&url=http%3A%2F%2Fwiki.ouitech.fr%2Flib%2Fexe%2Ffetch.php%2Febooks%2Fo_reilly-html_xhtml_the_definitive_guide_2002_.pdf%3Fid%3Debooks%3Aebooks&ei=g54_V

bnNFsrBggTTs4DoBw&usg=AFQjCNFsrWGPz9im5q3uS6xO3MO4y2wBMw&sig2=2jqpBftXsKg2e_RW5jiJuw&bvm=bv.91665533,d.eXY > . Acesso em : 27 abr. 2015.

KOST, Stephen. **An Introduction to SQL Injection Attacks for Oracle Developers,** Integrigy, Illinois, 2004. Disponível em: <http://www.net-security.org/dl/articles/IntegrigyIntrotoSQLInjectionAttacks.pdf >. Acesso em : 10 maio 2015.

LIMA, Allyn Grey de Almeida. Padrão SQL e sua Evolução, **Instituto de Computação - Unicamp**, [2015?]. Disponível em: <http://www.ic.unicamp.br/~geovane/mo410-091/Ch05-PadraoSQL-art.pdf >. Acesso em : 4 abril 2015.

LITCHFIELD, David. et al. **The Database Hacker's Handbook:** Defending Database Servers. Indianapolis: Wiley Publishing, 2005. Disponível em : <https://leaksource.files.wordpress.com/2014/08/the-database-hackers-handbook.pdf>. Acesso em: 10 maio 2015.

Magalhães, Dimmy K. S.; SOUSA, Luiz Cláudio D. De Mata. JSafeFilter: Filtro de Requisições HTTP para anular Ataques de SQL Injection, **Universidade Federal do Piauí, Departamento de Informática e Estatística**, Bacharelado em Ciência da Computação, [2010?]. Disponível em: <http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/artigos/graduacao/g4.pdf>. Acesso em: 1 jun 2015.

MICROSOFT . **Access SQL:** conceitos básicos, vocabulário e sintaxe, 2007. Disponível em: https://support.office.com/pt-br/article/Access-SQL-conceitos-b%C3%A1sicos-vocabul%C3%A1rio-e-sintaxe-444d0303-cde1-424e-9a74-e8dc3e460671?ui=pt-BR&rs=pt-BR&ad=BR>. Acesso em: 3 abril 2015.

NAKAMURA, Emílio Tissato, Paulo Licio de Geus, **Segurança de Redes em Ambientes Corporativos**. Novatec, 2007. Disponível em: <https://www.academia.edu/9475240/Seguran%C3%A7a_de_Redes_em_Ambientes_Cooperativos>. Acesso em: 04 abr. 2015.

NEVES, Pedro M. C. **O Guia Prático da HTML** , Lisboa: Centro Atlântico, 2004. Disponível em: <http://www.centroatl.pt/titulos/tecnologias/imagens/excerto-e-book-ca-oguiapraticodahtml.pdf >. Acesso em: 30 abr. 2015.

NUNES, Sérgio Sobral. Comunicações Digitais e Internet**, Universidade do Porto - Faculdade de Engenharia**, [2015?]. Disponível em: <http://paginas.fe.up.pt/~ssn/disciplinas/cdi/www/5.html> . Acesso em: 26 abr. 2015.

NUNAN, Eduardo.et al. Classificação Automática de Cross-Site Scripting em Páginas Web, **Instituto de Computação – Universidade Federal do Amazonas,** Manaus, [2010?]. Disponível em: <http://ce-resd.facom.ufms.br/sbrc/2012/ST10_1.pdf>. Acesso em: 01 jun. 2015.

OLIVEIRA, Celso Henrique Poderoso. **SQL**:Curso Prático, São Paulo: Novatec, 2002. Disponível em:

<http://minhateca.com.br/gahdumont/Cursos/Cursos/Programa*c3*a7*c3*a3o/Livro+-+SQL+Curso+Pr*c3*a1tico+-+Celso+Oliveira,75190141.pdf >. Acesso em: 31 março 2015.

OSWAP. Code Injection, **Open Web Application Security Project**, 2013a. Disponível em: <https://www.owasp.org/index.php/Code_Injection>. Acesso em: 06 maio 2015.

OSWAP. Types of Cross-Site Scripting, **Open Web Application Security Project**, 2013b. Disponível em : <https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting>. Acesso em: 06 maio 2015.

OSWAP. Blind SQL Injection, **Open Web Application Security Project**, 2013. Disponível em: <https://www.owasp.org/index.php/Blind_SQL_Injection>. Acesso em: 06 maio 2015.

OSWAP. LDAP injection, **Open Web Application Security Project**, 2015. Disponível em: <https://www.owasp.org/index.php/LDAP_injection>. Acesso em: 06 maio 2015.

OSWAP. As 10 vulnerabilidades de segurança mais críticas em aplicações WEB, **Open Web Application Security Project**, 2007. Disponível em: <https://www.owasp.org/images/4/42/OWASP_TOP_10_2007_PT-BR.pdf>. Acesso em: 10 maio 2015.

PYTHON, **The Python Wiki**, 2015. Disponível em: <https://wiki.python.org/moin/>. Acesso em: 1 de out. 2015.

QIAN, Kai et al. **Desenvolvimento Web Java**, Rio de Janeiro: Livros Técnicos e Científicos, 2010.

SANTO, Adrielle Fernanda Silva do Espírito. Segurança da Informação, **Departamento de Ciência da Computação - Instituto Cuiabano de Educação (ICE)**, Cuiabá, 2010. Disponível em: <http://www.ice.edu.br/TNX/encontrocomputacao/artigos-internos/aluno_adrielle_fernanda_seguranca_da_informacao.pdf>. Acesso em: 12 maio 2015.

SILVA, Maurício Samy. **HTML 5** : A linguagem de marcação que revolucionou a web, São Paulo: Novatec, 2011. Disponível em: <https://www.novatec.com.br/livros/html5/capitulo9788575222614.pdf>. Acesso em: 29 abr. 2015.

SILVA, Suellen de Castro Gomes da. UMA FERRAMENTA PARA A EXECUÇÃO DE ATAQUES SQL INJECTION**, FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"**, BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO, CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA, 2012. Disponível em: <http://aberto.univem.edu.br/bitstream/handle/11077/874/UMA%20FERRAMENTA%20PARA%20A%20EXECU%C3%87%C3%83O%20DE%20ATAQUES%20SQL%20INJECTION.pdf?sequence=2>. Acesso em: 12 maio 2015.

SQLMap, **SQLMap**, 2015. Disponível em: <http://sqlmap.org/>. Acesso em : 10 de out. 2015.

TAYLOR, Allen G.. **SQL for Dummies**, Indianapolis: Wiley Publishing, 5th Edition, 2003. Disponível em: <http://edc.tversu.ru/elib/inf/0142.pdf >. Acesso em: 4 abr. 2015.

USP, Formulários, **Instituto de Matemática e Estatística da Universidade de São Paulo, Universidade de São Paulo**, [2015?]. Disponível em: <https://www.ime.usp.br/~glauber/html/forms.htm>. Acesso em: 29 de abr. 2015.

WIKIPÉDIA, **Havij – Advanced SQL Injection**, 2015. Disponível em: <https://pt.wikipedia.org/wiki/Havij_-_Advanced_SQL_Injection> . Acesso em: 29 de out. 2015.

W3C. Tim Berners-Lee, **The World Wide Web Consortium**, [2014?]. Disponível em: <http://www.w3.org/People/Berners-Lee/> . Acesso em: 4 abr. 2015.

W3C. The WorldWideWeb browser, **The World Wide Web Consortium**, [2015?]. Disponível em: <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html />. Acesso em: 27 abr. 2015.

ZEMKE, Fred. What`s new in SQL: 2011, **Special Interest Group on Management of Data**, 2012 .Diponível em: <http://www.sigmod.org/publications/sigmod-record/1203/pdfs/10.industry.zemke.pdf>. Acesso em : 4 abr. 2015.