Relatório de Pesquisa
Diretrizes de segurança para software na IoT

Alex Sander Alves dos Santos e Daniel Macêdo Batista

Dispositivos que não tinham conexão com quaisquer tipos de redes, passaram a fazer parte de comunicações por meio da internet. Com isso, um novo paradigma é originado, que é o da Internet das Coisas ou em inglês *Internet of Things* (IoT). Os dispositivos com baixo poder de processamento que tem interação entre si e cooperação com outros e tem um mesmo objetivo fazem parte desse novo paradigma. Diversos locais contam com a presença desses dispositivos por conta da diminuição dos custos e por conta da sua miniaturização.

Um modelo de arquitetura de hardware com específicos paradigmas de comunicação e conectividade foi construído como consequência da miniaturização dos dispositivos inseridos no paradigma da Internet das Coisas (IoT). As previsões para dispositivos IoT conectados à internet para o ano 2020 foram em torno de 20 bilhões. Por conta da menor capacidade de processamento presente nesses dispositivos e a falta de preocupação com a informação dos usuários finais que trafegam nesse meio, a questão sobre segurança e privacidade não ocorre no ritmo de evolução de desenvolvimento e aplicação na IoT .

Os dispositivos IoT, por conta da menor capacidade de processamento e bateria, contam com sistemas operacionais específicos como Contiki e Tiny OS. Os dispositivos que geralmente estão ao alcance do público, de modo que qualquer cidadão tenha acesso facilitado a esses sensores e que não possuem supervisão adequada, costumam apresentar vulnerabilidades a ataques. O acesso ao sinal desses dispositivos costuma ser livre pois a comunicação dos sensores em sua comunicação é por meio de redes sem fio. Por conta da miniaturização, os mecanismos de segurança aumentam o consumo de energia dos sensores causando efeitos indesejáveis e tornando a complexidade alta para tornar os dispositivos seguros.

Nos dispositivos de IoT, os protocolos são de baixa capacidade com diferença aos equipamentos utilizados em redes convencionais. Para especificar a camada física de acesso ao meio, nesses dispositivos, um padrão de comunicação (IEEE 802.15.4) foi criado.

A baixa taxa de transmissão dos dispositivos IoT é a principal característica desses dispositivos pela menor capacidade de bateria. Ter um gasto de energia razoável e maior facilidade de instalação é o objetivo do padrão de comunicação IEEE 802.15.4, fornecendo confiabilidade na transmissão dos dados.

As características apresentadas pelo IEEE 802.15.4 são:

· Baixo consumo de energia;
· Transmissão sem fio de até 250 Kbps;
· Endereços de 16 bits (curto) até 64 bits (estendido);
· 16 canais na banda de 2450 Mhz;
· Protocolo de reconhecimento para transferência confiável.

Para a utilização do IPv6 sobre o padrão IEEE 802.15.4 existe um protocolo denominado 6LoWPAN. Para agilizar o processo de transmissão, o 6LoWPAN comprime os cabeçalhos IPv6 fragmentando e desfragmentando os pacotes da camada física, conforme o *Maximum Transmission Unit* – Unidade máxima de transmissão da camada física.

A arquitetura básica de dispositivos IoT tem como modelo a composição de unidades processamento/memória, comunicação, fonte de energia e sensores/atuadores. Na unidade de processamento/memória, utiliza-se uma Unidade Central de Processamento – CPU (as mesmas utilizadas em sistemas embarcados), um microcontrolador e um conversor analógico digital. Tem como característica reduzir o espaço físico e o consumo de recursos. Para a unidade de comunicação é necessário no mínimo um canal, como exemplo aqueles definidos pelos padrões *Bluetooth Low Energy* (BLE), IEEE 802.15.4 e IEEE 802.11. Por se tratar de comunicação sem fio,

rádios de baixo custo e baixa potência são utilizados, o que resulta em baixo alcance e frequentemente perdas de pacotes. Com a função de prover energia aos componentes do dispositivo, há uma fonte energética que pode ser composta por uma bateria recarregável ou não. Por fim, o monitoramento do ambiente é realizado por sensores e atuadores no ambiente no qual o dispositivo está inserido. Dados como pressão, ambiente, acidez do solo, entre outros, são capturados por meio de alguma ação manual ou automaticamente enviados aos atuadores.

Rede autônoma, rede de objetos inteligentes limitada e IoT autêntica são modelos de dispositivos de conectividade IoT. Rede autônoma não possui acesso à internet, não podendo ser acessada por meio de ambiente externo. Como exemplo usinas nucleares, sistema de climatização predial. Modelos que adotam medidas de segurança e proteção como a internet estendida, tem seus dispositivos parcialmente ou totalmente conectados à rede externa. Nos modelos de cidades inteligentes é necessário o controle de acesso nos quais são utilizadas configurações de firewalls e proxies. Por ter conectividade direta à internet o modelo de IoT é autêntico e diferente do modelo de redes autônomas.

Alguns desafios com relação a segurança, como os metodológicos, são preocupações específicas para a evolução da IoT. É preciso resolver esses desafios. Processos de engenharia de segurança em sistemas embarcados, como os da IoT, se tornam formais e trabalhosos. Métodos Ágeis de Desenvolvimento de software, como modo de enfrentamento aos desafios de segurança dos sistemas de IoT devem ser adaptados pelas empresas de desenvolvimento.

O desenvolvimento de produtos de IoT não costuma priorizar a segurança com o grau de importância que o assunto necessita em sua prática. Em relação às pesquisas de requisitos funcionais e não funcionais é possível constatar que a preocupação com o desenvolvimento seguro na IoT é baixo. As preocupações quanto a segurança nas redes de sensores e/ou comunicação móvel com IoT se restringem as camadas de aplicação e nuvem.

Confidencialidade, integridade e disponibilidade são os principais requisitos a serem alcançados, como revelam os estudos da eficácia das metodologias. Diversas metodologias de desenvolvimento são abordadas por empresas como Siemens, Philips, Boch e startups por conta da não padronização dos dispositivos. Em termos de Desenvolvimento o foco é a captura do valor e agilidade do cliente.

O aumento da produção de software na IoT é facilitado por abordagens rápidas como Ágil, Devops e Lean Startup, que em seu processo de gerenciamento contam com feedbacks rápidos e iteratividade sem a abordagem em questões sobre segurança.

As garantias de segurança no desenvolvimento da IoT podem ser efetuadas por meio das práticas de engenharia de software, mesmo que a segurança não seja considerada como requisito funcional. Há de se considerar possíveis problemas mapeados, geralmente relacionados a outras questões, mas que apontam como erros de segurança. Um dos desafios de segurança no desenvolvimento da IoT é o inestimável valor que o hardware e o software desempenham juntos no ambiente de dispositivos conectados com importantes dados do cliente.

Em desenvolvimento de software, a abordagem da entrega contínua por meio da abordagem iterativa é uma proposta para as questões de segurança na IoT. Além da possibilidade de modelagem de possíveis ameaças, há o aumento da probabilidade de se desenvolver softwares seguros. Para os diferentes nichos do mercado em aplicações IoT é preciso, em cada etapa do processo iterativo, que sejam identificados controles adequados de segurança em conformidade com os regulamentos específicos.

Uma metodologia de desenvolvimento de IoT segura, apareceu como tema recente de pesquisa. Durante essa pesquisa, a literatura de engenharia de software e sistemas foi verificada e dois casos iniciais foram analisados e comparados com o intuito de extrair os desafios relacionados à segurança no processo de produção do produto com base em IoT. Foram identificados 17 desafios de segurança divididos e categorizados em técnica, organizacional e metodológica:

Categorias	Desafios	Descrição
Metodológicas	**Incorporar requisitos de segurança no processo de desenvolvimento ágil de software**	**Considerando a segurança como um item da lista de pendências no desenvolvimento iterativo**
Metodológicas	Abordagens de garantia de segurança	O teste de segurança é complexo para definir, planejar e executar, envolvendo especialistas em segurança e outras partes interessadas
Metodológicas	**Segurança em tempo de execução**	**Patches de segurança e atualizações precisam ser implementados e entregues oportunamente durante as operações do sistema**
Metodológicas	Segurança para código legado	Adicionando segurança a uma infraestrutura IoT existente, com impacto significativo
Metodológicas	Priorizando controles de segurança	Devido a um grande número de controles de segurança, o desafio é priorizar e liberar o planejamento sem comprometer a segurança para cada release
Metodológicas	Evolução dos requisitos de segurança	Objetivos, requisitos e controles de segurança podem mudar
Organizacional	Segurança para componentes de terceiros	Os desenvolvedores não têm ideia sobre a segurança dos componentes de terceiros
Organizacional	**Requisitos específicos regulatórios do mercado, padrões de segurança, políticas e objetivos**	**Normas de conformidade e segurança regulamentares específicas do mercado, políticas e objetivos precisam ser levados em consideração**
Organizacional	Variedade de requisitos de segurança sob medida	Os requisitos de segurança variam entre diferentes clientes, diferentes domínios de aplicativo
Organizacional	Fatores humanos na segurança operacional	Ameaças relacionadas à permissão, privilégios e controle de acesso durante operação de sistema
Técnica	Compreensão abrangente da arquitetura para segurança da IoT	Consideração arquitetural da segurança no sensor, rede, aplicação e nível de armazenamento de dados
Técnica	**Garantir a segurança dos dados**	**Identificação e implementação de controles de segurança apropriados para garantir a confidencialidade, integridade e disponibilidade dos dados**
Técnica	Portabilidade de rede	O tamanho e a estrutura da rede em um sistema IoT podem dinamicamente mudar
Técnica	**Inter-relacionamento com requisitos não funcionais**	**A segurança está intimamente relacionada à segurança e privacidade**
Técnica	**Restrições de recursos físicos**	**Mecanismos de segurança para nós da IoT, que são limitados por restrições de memória física, poder computacional e armazenamento de dados**
Técnica	Heterogeneidade e distribuição de sistemas	O aplicativo IoT se baseia fortemente na comunicação de máquina para máquina o que adiciona um nível de complexidade em termos de atenuadas ameaças relacionadas à segurança
Técnica	Fluxo de dados complexo	Entendendo os dados em torno de seu sistema e quais ativos precisam ser protegidos

Considerando a metodologia de *Security by Design*, existem requisitos fundamentais, sendo eles:

- Confidencialidade: vai desde a origem do dado, sua persistência e são definidos durante o ciclo de vida da informação.

- Integridade: garantem a proteção contra modificações sem autorização e buscam a garantia da confiabilidade.

- Disponibilidade: requisitos de manutenção e continuidade de negócios. Há nesse requisito os riscos de destruição dos ambientes de software e interrupção de serviços.

- Autenticação: reúnem requisitos que garantem a validação de identificação por entidades requisitantes com técnicas de autenticação multi-fator.

- Autorização: requisitos que fazem relação a uma entidade autenticada, aplicando princípios de privilégios.

- Responsabilidade: requisitos que por meio de auditoria constroem a história das ações de um usuário.

Os desafios de segurança na IoT são necessários por conta da diversidade de possibilidades de aplicações em dispositivos inteligentes utilizados na constituição de casas ou cidades, que fornecem meios de gerenciamento e controle automatizado de diferentes tipos de serviços. Dessa quantidade de possibilidades de aplicativos, surge também variedade de ataques com o objetivo de explorar as vulnerabilidades da infraestrutura da IoT.

Entre as possibilidades de ataque que podem prejudicar as aplicações de IoT, uma delas é o de Negação de Serviço Distribuído (DDoS do inglês *Distributed Denial of Service*). Para que haja um ataque DDoS é necessário que dispositivos maliciosos conectados à rede inundem as aplicações alvo forjando requisições falsas. Há uma sobrecarga na largura de banda da rede que acaba negando requisições legitimas, tornando o alvo inacessível. Em ambientes de IoT, os ataques de DDoS ocorrem da mesma maneira, o que difere é a quantidade diversa e heterogênea de dispositivos, o que aumenta as possibilidades de invasão. Esse tipo de ataque é o que tem causado o maior número de prejuízos em grande escala em projetos de IoT no qual uma rede opera em coordenação entre dispositivos sequestrados (conhecidos como bots ou zumbis) que enviam requisições em repetição ao alvo que esgotam os recursos da rede.

Ataques DDoS em ambientes IoT têm sido impulsionados por conta do maior número de dispositivos na rede. O impacto dessas ameaças é potencializado por conta das localizações geográficas dos mesmos e de suas limitações quanto a segurança. Na área de segurança de redes, o desenvolvimento de mecanismos de mitigação de ataques DDoS se torna um desafio. Um exemplo desses tipos de ataques, foi o Mirai Botnet, de 2016, um ataque DDoS massivo, ocorrido por meio de dispositivos IoT. Outro exemplo de ataque massivo foi o ocorrido no primeiro semestre de 2021 na plataforma Azure de um cliente Microsoft, composta por 70 mil bots vindos da Ásia e estados Unidos.

Em cenário de redes da IoT, simulações são necessárias na busca de vulnerabilidades. Para as simulações no estudo corrente, fez-se a utilização do simulador Cooja. O simulador Cooja é executado por meio do sistema operacional Contiki OS de código aberto para ambientes de IoT. Permite a conexão de microcontroladores de baixo custo e baixo consumo de energia. O Contiki OS é instalado diretamente no hardware, e pode ser utilizado em maquinas virtuais.

Nos cenários de experimentação foram escolhidos os protocolos de roteamento IPv6 para redes de baixa potência e perdas o RPL. Uma classe de roteadores com restrita capacidade de processamento, memória e energia (LLN) fornece tráfego multiponto a ponto nos dispositivos com protocolo RPL em simulações no Cooja por meio do Contiki OS. Dois tipos de nós foram escolhidos, o nó RPL-collect sink e o nó RPL-collect sender. O alvo dos ataques é o nó sink (RPL-collect sink) e o nó sender (RPL-collect sender) serve tanto para alvo como nó com código malicioso.

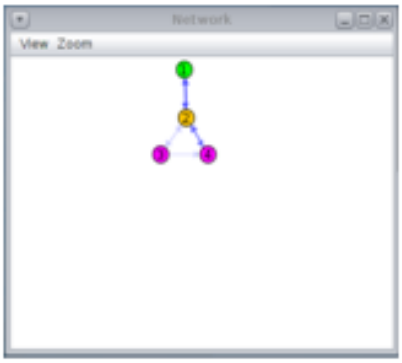
Os nós atacantes (sender) enviam uma quantidade alta de pacotes HELLO com o objetivo de provocar perda na coleta de dados pelo nó coletor (sink/sender). Para tornar efetiva a simulação de ataques DDoS em ambientes IoT, houve a necessidade da adaptação de um código malicioso

inserido aos nós atacantes no qual a taxa de transmissão foi alterada de um pacote por minuto para um pacote por segundo e os disparos são ativados por meio de uma ação de clique em algum nó (sender) disparado por um usuário. As simulações efetuaram testes do desempenho da rede antes e depois dos ataques para objetivo de comparação.

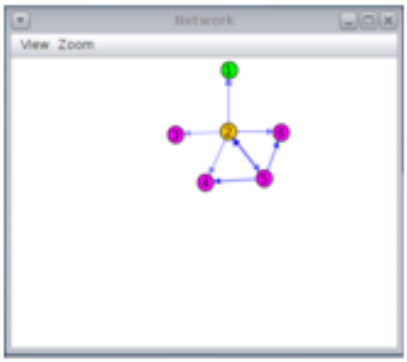
A simulação teve início com 1 dispositivo modificado com o código de ataque se comunicando com um dispositivo normal, contendo o código original do dispositivo. Para que um ataque fosse iniciado era necessária uma ação em um dispositivo com o código inserido. Essa ação foi programada com o clicar de um botão emulado nos dispositivos atacantes.

Na ferramenta Cooja*,* é possível replicar a quantidade de dispositivos permitindo diferentes simulações com diferentes quantidades e tipos de nós na rede. Nos experimentos realizados, os números de dispositivos foram aumentados de 1 atacante se comunicando com 1 alvo normal até 15 atacantes se comunicando com 1 alvo normal. A partir desse ponto os experimentos utilizaram 20, 30, 40, 50 e 60 atacantes se comunicando com 1 alvo normal. Tentativas de experimentos a partir de 75 nós causaram travamento da máquina utilizada, não sendo possível prosseguir com simulações maiores. Simulações também foram efetuadas com 15 atacantes se comunicando com 15 alvos normais e com 1 atacante se comunicando com 30 alvos normais.

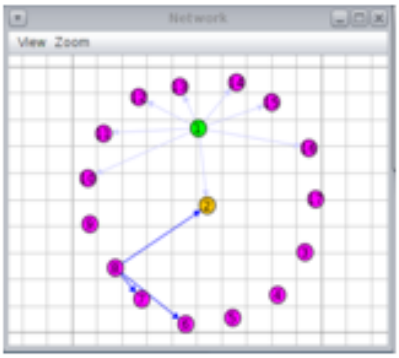
Para cada experimento, 2 simulações tiveram que ser efetuadas e catalogadas, a primeira simulação foi efetuada com os dispositivos em operação normal, sem que houvesse a ativação do ataque. O tempo nos dois experimentos em cada simulação foi igualada para comparação após a coleta de dados. A ferramenta de coleta de dados do Cooja não permite exportar os dados coletados em formatos editáveis, sendo que a cada experimento (2 para cada simulação), foi necessária a digitação e elaboração de gráficos em software de planilha eletrônica. Os dados analisados e coletados se referem a Perda de Dados, Transmissão de Pacotes e dados sobre energia dos dispositivos. Foram criadas 48 planilhas com 8 gráficos cada, totalizando 384 gráficos, sendo 192 em situação sem ataque (não codificado com ativação de ataque a rede) e os outros 192 com código alterado. Esses dados foram gerados para posterior comparação e comprovação de que a rede teve alteração. Os gráficos a seguir ilustram os experimentos que foram realizados.



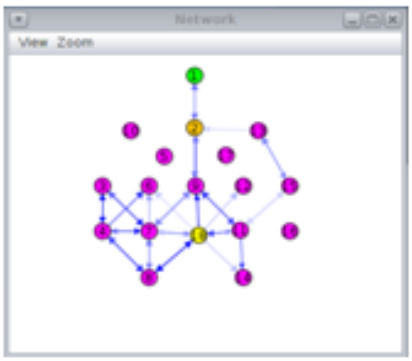
2 ataca 1



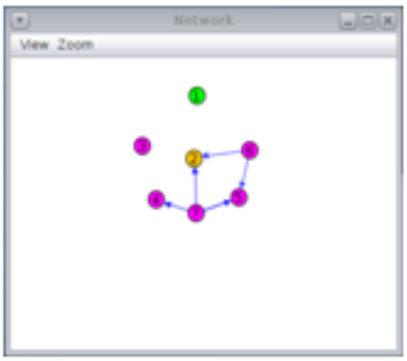
4 ataca 1



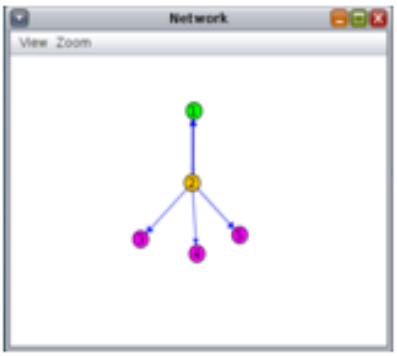
15 ataca 1



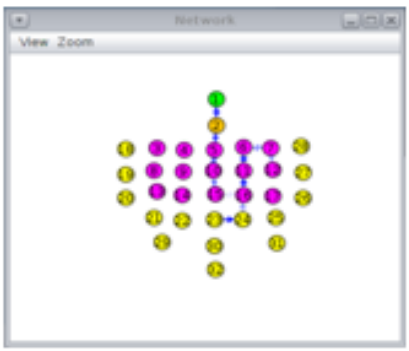
15 ataca 1



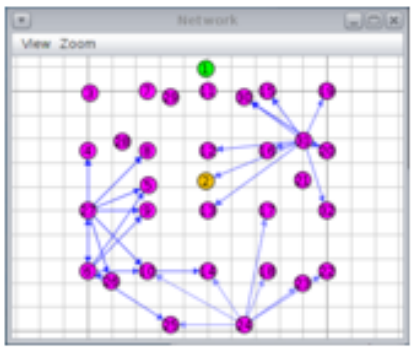
5 ataca 1



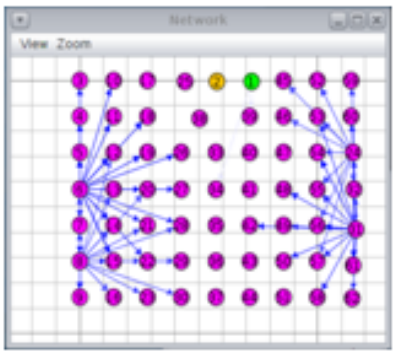
3 ataca 1



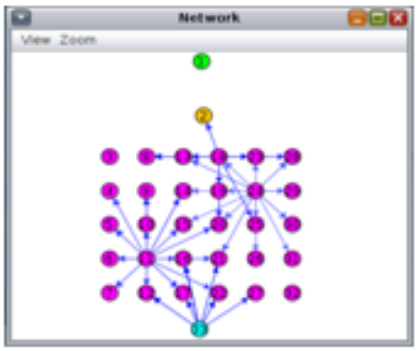
15 ataca 15



30 ataca 1



60 ataca 1



1 ataca 30

Há ainda o desenvolvimento de outros malwares em IoT, que utilizam ataques DDoS, trazendo para a indústria e a academia a necessidade de pesquisas e estudos dessas ameaças. Pesquisadores no assunto, documentam e atualizam com periodicidade, modelos, modos de operacionalização e protocolos que esses ataques utilizam.

A segurança é uma questão que deve ser levada em consideração em todo o processo de desenvolvimento de software. A capacidade de recuperação, ou seja, a resiliência, é uma característica de segurança de um software. Problemas de segurança ocorrem à medida que as empresas evoluem digitalmente, de modo que aumenta a demanda por ferramentas, metodologias e tecnologias de desenvolvimento de software. As empresas costumam tomar medidas proativas de segurança no momento posterior a ocorrência de ataques em seus sistemas.

Implementar práticas de segurança durante todo o processo de desenvolvimento de software é o que se conhece por abordagem *Security by Design*. É necessário que haja a interação das equipes de desenvolvimento e segurança definindo claramente os requisitos e testagem de código que seja adequado ao projeto do sistema. Conforme cresce o número de vulnerabilidades, surge a necessidade de inserir as questões de segurança desde o início do desenvolvimento de software. O principal empecilho da utilização do Security by Design é a falta de conhecimento técnico da equipe de desenvolvimento em aspectos relacionados à segurança da informação.

Para a aplicação do Security by Design é necessário que os requisitos de segurança da informação sejam aplicados cotidianamente ao ciclo de trabalho das equipes de desenvolvimento de modo que tal abordagem seja um padrão. Ao utilizar a abordagem Security by Design em uma arquitetura segura de software, as melhores práticas e padrões de segurança são consideradas, garantindo assim segurança e privacidade. Os principais benefícios do Security by Design em relação a uma abordagem tradicional são:

- Economia: aumento da eficiência e economia desde o início do projeto ao se considerar os problemas de segurança da informação.

- Resiliência: O sistema se torna resiliente em termos de ataques quando as práticas de Security by Design são incorporadas se tornando um padrão no processo de desenvolvimento de software e não apenas de modo corretivo.

- Correção de vulnerabilidades: em tempo de desenvolvimento, as vulnerabilidades de software são sanadas por meio das práticas de conscientização, conhecimento e verificações em tempo de desenvolvimento.

- Adaptação: no processo de desenvolvimento, um meio de evitar novos erros é determinar quais os erros já foram cometidos e adaptá-los.

Segundo a *Open Web Application Security Project* (OWASP), uma arquitetura de software segura baseada em Internet das Coisas (IoT) para cidades inteligentes deve ter os seguintes 10 princípios de relevância e importância:

1 – Superfície de ataque minimizada

Restringir as funções de usuário de modo a minimizar a área superficial de ataque é um meio de reduzir as vulnerabilidades do sistema; Aplicar monitoramento integrado a ferramentas protetivas em tempo real com processo corretivo; Técnicas de desenvolvimento seguro aplicadas em servidores (entrada e saída monitorada; processos de backup e recuperação), servem como medidas de redução de superfície.

2 – Definição de Padrões

Questões relacionadas à segurança durante o desenvolvimento de software devem ser incorporadas como padrão de projeto de modo que auxilie as equipes. As equipes precisam ter orientação para o desenvolvimento de software seguro que englobem:

- Filtros nos campos de entrada
- Implementação no cliente sem validação
- Codificação das respostas
- Formulários sem memorização de informações

Informações com sigilo devem ter:

- Tratamento de erros
- Definição de senha com padrões de segurança fortificados
- Autenticação em Web Services

3 – Principio do menor privilégio

Uma estratégia de oferecer aos usuários permissões mínimas no qual tempos e direitos são determinados para a realização de determinadas tarefas com o objetivo de garantir privacidade e a segurança das informações. Determinados perfis de usuários devem ter permissões que impeçam o acesso, edição ou extração de informações.

Para a garantia de menor privilégio em servidores e sistemas as medidas a serem tomadas são:

- Usuários convidados com perfil determinado
- Especificar o que é proibido
- Tarefas básicas especificadas nas permissões de usuários
- Definição de pontos com determinadas permissões para perfis específicos de usuários

4 – Princípio da defesa em profundidade

Um conjunto de ações práticas que tem como prioridade detectar e reagir em casos de invasão. Devem ser utilizadas ferramentas de contra-ataques como firewall, antivírus de conteúdo, criptografia e controle de acesso.

5 – Falhar com segurança

A manipulação de erros tem relevância importante para o desenvolvimento de software seguro em uma abordagem de *Security by design*. Dois tipos de erros são destacados:

- Controle de segurança processado deve ser a primeira exceção
- Exceção relevante é aquela em que um código não pertence a um controle de segurança

As exceções devem reconhecer e tratar comportamentos anormais no sistema. Mecanismos de exceção, proibição e/ou permissão não autorizada devem fazer parte de todo sistema desenvolvido de forma segura.

6 – Confiança zero em serviços

O modelo de confiança zero recomenda que as empresas não confiem em dispositivos, sistemas ou pessoas antes que haja alguma verificação padronizada nas conexões. Esse modelo foi criado por conta da falsa crença que as ameaças de segurança da informação ocorrem por meio do ambiente externo e o ambiente interno está imune a ataques. As ameaças internas ocorrem por meio de qualquer pessoa da equipe que tenha acesso privilegiado, ou seja, pode vir de colaboradores, ex-colaboradores, parceiro de negócio e/ou prestador de serviço. Empresas

podem levar anos para que descubram esse tipo de ameaça. Um modelo de confiança zero deve considerar os seguintes tópicos:

- Conhecimento da estrutura e arquitetura da empresa
- Usuários com identificação única e fortificada
- Desenvolvimento de autenticidade por meio de processo
- Monitoramento de serviços e dispositivos
- Definição de políticas em conformidade com dados e serviços
- Controle no acesso de dados e serviços
- Não confiar nem na rede local

7 – Funções devem ser segregadas

A divisão de funções de usuários por perfil, atividade ou função em um sistema deve ser priorizada. Os modelos de acesso privilegiado em sistemas e infraestrutura de empresas fornecem modelos administrativos de perfis por função (*Role Based Access Control*). Agrupar funções permite a visualização dos privilégios dos usuários possibilitando controle de acesso com segurança.

Para a implementação da segregação de funções é preciso:

- Aplicação de ambiente com regras de segregação bem definidas
- Matrizes de risco
- As análises de riscos devem ser utilizadas para a identificação das violações da segregação de funções
- Conflitos envolvendo usuários alternativos devem ser analisados.
- Resolução dos conflitos de alto risco

8 – Segurança por obscuridade evitada

Há por parte de desenvolvedores a crença de que resultados que resultem em vulnerabilidades em seus sistemas, podem não ser encontrados, caso a codificação do software seja secreta. O problema é depender do sigilo do código implementado. Alguns desenvolvedores acreditam que não obter resultados em vulnerabilidades de um sistema, constitui segurança. Os modos adotados com o intuito de prevenir a segurança por obscuridade são:

- Palavras fortes em senhas
- Treinamento e conscientização dos desenvolvedores
- Mínimo privilégio possível
- Proteção de software e sistemas de recuperação como backup

9 – Simplicidade na segurança

Na metodologia *Security by design*, ao buscar obter um sistema seguro é normal utilizar ferramentas, controles e processos. Ao iniciar um projeto deve-se pensar e planejar a utilização de ferramentas com maior segurança e menor complexidade aos controles do sistema. Procedimentos, processos e ferramentas que falham em sua documentação aumentam as vulnerabilidades no software projetado.

Para que haja eficácia na busca da simplificação de um sistema seguro é preciso considerar que:

- Um sistema seguro vai além da tecnologia. Processos, políticas e pessoas devem ser levados em consideração. O que garante um software seguro, são as medidas proativas e não somente a execução de checklists de segurança.
- As boas práticas de segurança em tempo de desenvolvimento devem estar interiorizadas pelas equipes por meio de treinamento e todos os interessados devem estar cientes de possíveis alterações necessárias no sistema.

- Atender os princípios básicos de segurança e utilizar ferramentas adequadas a arquitetura do sistema.

10 – Segurança no processo de manutenção do software

São necessários estudos, por parte das equipes de desenvolvedores, que possibilitem corrigir eficientemente as vulnerabilidades dos sistemas desenvolvidos. Precisam entender como as vulnerabilidades deformam as arquiteturas dos sistemas. A falta de processo de controle das correções ou que identifique e trate as vulnerabilidades do sistema pode fazer que novos problemas de segurança surjam com o tempo. A *Security by Design* colabora estrategicamente na análise dos processos e riscos que os sistemas são envolvidos.

Em processos que envolvam gestão eficiente de vulnerabilidades devem ser encontrados:

- Mapeamento de riscos
- Riscos analisados e prioridade
- Identidade e responsabilização
- Vulnerabilidades tratadas
- Relatórios de acompanhamento

Há significativa dependência de software e sistemas pelas organizações modernas. As proteções contra falhas de segurança são garantidas por meio de codificação padronizada. Lacunas que possam comprometer a segurança de um sistema são mitigadas por meio desses padrões de codificação. Para o desenvolvimento moderno de sistemas, sejam eles, mobile, desktop ou WEB é crítica a prática de codificação. Conforme o *Software Engineering Institute*, 90% das falhas de segurança são encontradas em códigos ou sistemas mal arquitetados. Práticas seguras de desenvolvimento são essenciais. Mesmo existindo diversas abordagens de desenvolvimento de sistemas com segurança a OWASP oferece padrões de codificação segura bem criteriosos. A Lista de Verificação OWASP relaciona as 14 medidas de tecnologia a serem aplicadas para minimizar as ocorrências de problemas de segurança no desenvolvimento de software.

1 - Validação de Entrada

A formatação correta de entrada de dados em formulários permite que dados errados sejam armazenados permitindo a falha de elementos sequenciais. Assim que o usuário insere algum dado no formulário a validação deve ser inserida no fluxo de dados. Variáveis que permitem ao sistema ações não padronizadas causando ameaças como injeção e scripts devem ter critérios rigorosos de entrada.

Algumas técnicas de segurança em validações de entrada conforme a OWASP:

- Todos os dados devem ser validados;
- Padrão centrado na aplicação na rotina de validação de dados;
- Verificação de caracteres ASCII nos valores de cabeçalho em solicitações e respostas;
- Criar lista “branca” de caracteres permitidos em entradas de dados para validação;
- Caracteres especiais permitidos devem ter implementação de controles específicos com codificação de saída.

Exemplo:

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>

<h2>JavaScript Validation</h2>

<form name="myForm" action="/action_page.php" onsubmit="return
validateForm()" method="post">
  Name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

Figura 1 Fonte: w3schools. Snippet de código de validação de JavaScript

No caso da Figura 1, o campo `Name` é obrigatório e exibe a mensagem de alerta "*Name must be filled out*" se o usuário ignorar o campo.

2 - *Output Encoding*

Os sistemas devem, ao receber dados, codificá-los anteriormente à saída. Um formato distinto de codificação na saída converte os caracteres especiais em formatos que não apresentam perigo para o mecanismo de tradução de destino.

Algumas técnicas de codificação de saída conforme a OWASP são:

- Testagem padronizada para as codificações de saída;
- As saídas de dados de consultas SQL, XML e LDAP devem ser limpas em contexto.

3 - Autenticação e senhas

Um dos métodos menos seguros de autenticação são as senhas amplamente utilizadas na proteção de dados. É um processo que confirma que uma pessoa é quem diz ser. Os indivíduos devem seguir certas diretrizes e procedimentos quanto às suas senhas de modo a manter a confidencialidade e disponibilidade dos serviços.

Os procedimentos recomendados pela OWASP são:

- Identificação monitorada de ataques em diversas contas de usuário com a mesma senha;
- Sistemas confiáveis por meio de implementação hash em senhas;
- Mensagens que contenham falhas de autenticação não devem informar quais dados estão com entrada inválida. As mensagens de falha devem ser as mesmas na visualização e na codificação;
- Políticas ou regulamentos que definam comprimento como requisito de senhas;
- Páginas e recursos que não são de acesso público devem exigir autenticação;
- A segurança das funções administrativas e de gerenciamento devem seguir os padrões do mecanismo principal de autenticação;
- Contas confidenciais e de alto valor devem ter mecanismos de autenticação multi-fator.

4 - Gerenciamento de Sessão

Em uma aplicação web com muitas solicitações, gerenciar as sessões se torna um dos requisitos de segurança a serem efetivados. O gerenciamento de sessão é utilizado pelo protocolo HTTP na interação entre páginas web e nos navegadores e as sessões armazenam solicitações pessoais.

Um exemplo de código PHP para criar uma nova sessão.

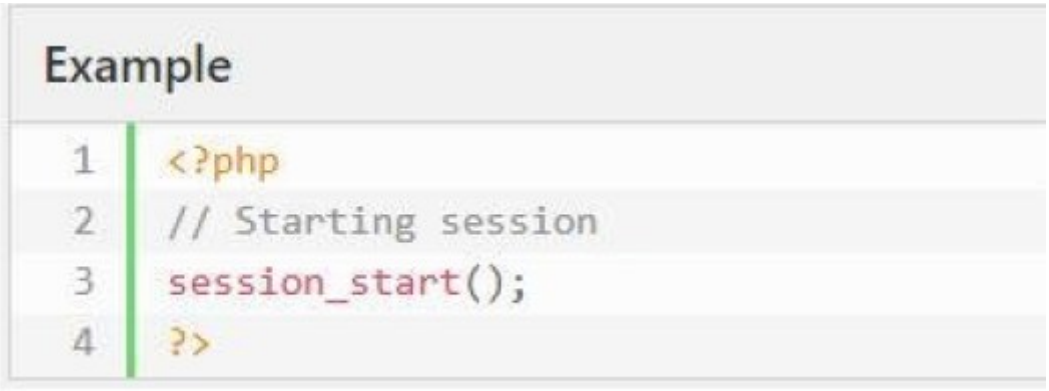


Figura 2 Fonte: tutorialrepublic.com

As melhores práticas, segundo a OWASP são:

- Restringir valor apropriado em site com sessão autenticada definindo domínio e caminho em cookies com identificadores;
- Exibir em páginas com proteção de autorização a funcionalidade de logout;
- Criação de novo identificador em cada nova autenticação;
- Cookies enviados via conexão TLS devem ter atributo definido como “seguro”;
- Os algoritmos devem garantir identificadores de sessão aleatórios e serem bem avaliados;
- Sessões inativas devem contar com tempo limite curto equilibrando requisitos funcionais do negócio e riscos.

5 - Controle de Acesso

Essencial em qualquer tipo de serviço e suportado pela maioria de serviços de segurança. Tem a função de autorizar ou não requisições de aplicativos ou usuário. Concede ou não credenciais de acesso.

Entre as abordagens de controle de acesso, algumas são destacadas pela OWASP:

- Funções protegidas apenas devem ter acesso restrito a usuários com autorização;
- Caso as informações de segurança não possam ser acessadas pelo aplicativo, todo o acesso deve ser negado;
- Utilizar objetos confiáveis ao sistema;
- As regras de negócio dos aplicativos devem ser documentadas e a criação de uma Política de Controle de Acesso com regras e critérios bem definidos é efetiva no provisionamento e controle adequado;
- Acesso a dados somente por usuários com autorização.

6 - Autorização e identificação para gerenciamento de acesso (por exemplo, logar no sistema da empresa e obter documentos).

- Identificação utilizando credenciais de login no formulário de entrada;
- Inserção de senha correspondente obtendo acesso à página conforme autenticação;
- Direitos de acesso atribuídos a conta são autorizações. Ao verificar identidade o controle de acesso define o que o usuário pode ou não fazer com os documentos.

7 - Práticas de Criptografia

A confidencialidade e integridade dos dados são mantidas por meio das práticas criptográficas. Utilização de chave simétrica ou criptografia de chave pública em transmissão de dados ou armazenamento, que dependem das demandas e riscos de segurança podem ser implementadas.

Boas práticas da utilização de criptografia:

- As falhas em módulos de criptografia, se ocorrerem, devem ser com segurança;
- As chaves de criptografia devem obedecer a processos e políticas estabelecidas;
- Segredos principais devem ser protegidos contra acesso não autorizado;
- Funções de criptografia de proteção de segredos devem estar implementados em sistemas confiáveis.

8 - Tratamento de Registro de Erros

Resultados inesperados em sistemas quando recebem alguma entrada incomum devem ter tratamentos de erros. Falhas em sistemas distribuídos causadas por inconsistências no tratamento de erros podem ocasionar vulnerabilidades.

Um erro em uma consulta SQL é mostrado abaixo, no qual o local de instalação do site é mostrado.

```
Warning: odbc_fetch_array() expects parameter /1 to be resource, boolean given
in D:\app\index_new.php on line 188
```

As modificações no sistema devem ser registradas e rastreadas. Invasores podem utilizar como ataque, perfis que utilizam as mesmas credenciais e com isso obter acesso total às contas. Isso pode ser mitigado, caso as modificações no sistema sejam rastreadas.

A OWASP lista algumas medidas a serem tomadas quanto ao manuseio de tratamento de erros nos sistemas de informação.

- Manipuladores de erros não devem conter informações de depuração ou rastro de pilha;
- Memória alocada em condição de erro devem ser liberadas;
- Falhas de segurança específicas devem ser suportadas pelos controles de registros;
- Validações de entrada com falhas precisam ser registradas;
- Todas as falhas e tentativas de autenticação devem ser registradas;
- As integridades de entrada de log devem ser validadas com função hash;
- Utilizar personalização em páginas de erro e implementar as mensagens de erro genéricas.

9 - Proteção de dados

Invasores utilizam diversas técnicas e métodos como o objetivo de obter informações de sistemas. Informações protegidas e/ou críticas podem ficar expostas ou serem furtadas e ainda adulteradas, por isso a proteção de dados é importante.

As melhores práticas de Proteção de Dados, conforme a OWASP são:

- Remoção de documentação não necessária em aplicativos que podem auxiliar atacantes;
- Formulários com informações importantes devem ter preenchimento automático removidos;
- Usuários devem ter privilégios mínimos em tarefas relacionadas a dados e informações do sistema;
- Dados armazenados em servidores devem ter controles implementados com controles de acesso apropriados.

10 - Segurança da comunicação

Em telecomunicações, informações podem ser interceptadas e levadas a agentes ilegais. Métodos de criptografia protegem os dados confidenciais de um sistema. A utilização de protocolos de segurança como o *Transport Layer Security* (TLS) combinado com técnicas criptográficas é útil na proteção de sistemas de comunicação. Como exemplo de falha de segurança de comunicação temos a invasão de email. O Outlook, por exemplo, em 2019 teve 773 milhões de invasões.

A lista de verificação, segundo a OWASP, traz algumas técnicas de segurança a serem observadas nos sistemas de comunicação:

- Padronizar de modo apropriado a utilização de TLS de modo único;
- Criptografar as transmissões de todas as informações confidenciais;
- Codificação de caracteres devem ser especificadas; e
- Informações confidencias com referência ao HTTP em vínculo com sites externos e parâmetros de filtros devem conter criptografia.

11 - Configurações do sistema

A composição do hardware do computador e toda a sua estrutura fazem parte da configuração do sistema. Pontos fracos são explorados de modo a serem encontradas falhas de segurança que permitam que atacantes obtenham direitos administrativos no sistema.

Os critérios, segundo a OWASP, para as configurações de sistemas são:

- Mínimo de privilegio possível para contas e processos em servidores web;
- Os métodos HTTP, Get ou Post, devem ter regras de suporte a aplicativos bem definidas e tratamento diferenciado em páginas do aplicativo;
- Implementação de gerenciamento de ativos do sistema com registro dos componentes de software;
- Código de teste ou funcionalidade fora do ambiente de produção devem ser descartados;

12 - Banco de Dados Seguro

Confidencialidade, Integridade e Disponibilidade são as garantias que o conjunto de tecnologias, procedimentos e políticas se relacionam a segurança em sistemas de banco de dados. Como exemplo, a Microsoft relata o caso ocorrido em 2020 no qual 250 milhões de dados de clientes da empresa foram publicados sem segurança de senha. Um ataque comum em banco de dados que explora a segurança é a injeção SQL. O exemplo abaixo é uma instrução OR que sendo TRUE, recupera todos os dados de uma determinada tabela:

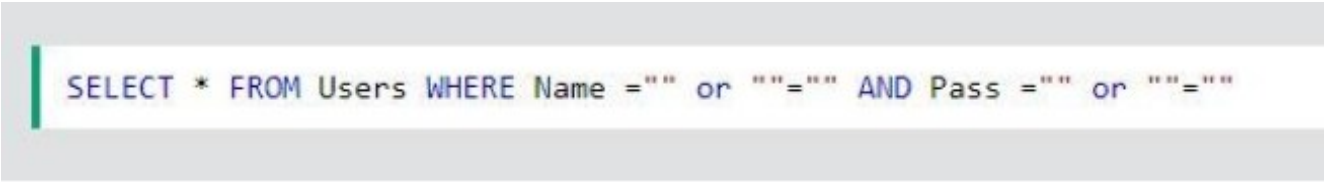


Figura 3 fonte: w3schools

As seguintes medidas de segurança de dados são recomendadas:

- Utilização de credenciais com segurança ao acesso ao banco de dados;
- O acesso aos dados deve ser por meio de procedimentos armazenados e que possam remover permissões nas tabelas base do banco de dados;
- Funcionalidades desnecessárias do banco de dados devem ser desativadas;
- Desativar contas padronizadas sem suporte às regras de negócio;
- Metacaracteres devem ser endereçados e as entradas e códigos de saída devem ser validados.

13 - Gerenciamento de Arquivos

A proteção de dados sigilosos contra invasores é efetuada por meio de políticas rígidas de gerenciamento dos direitos e autorizações. O gerenciamento de arquivos permite o estabelecimento de limites, responsabilidades e privilégios aos usuários do sistema. Conforme o Departamento de Saúde dos Estados Unidos, no primeiro semestre de 2021, 98,2 milhões de pessoas foram afetadas por roubo de dados.

A OWASP recomenda algumas estratégias:

- Tipos de arquivos devem ser limitados quanto ao seu fim e perfil de acesso;
- Arquivos que possam ter interpretação em servidores web devem ter upload restringindo ou impedido;
- Caminhos absolutos de arquivos não devem ser divulgados; e
- Arquivos carregados por usuários devem passar por análises virais.

14 - Gerenciamento de Memória

Para o Gerenciamento de Memória eficiente a OWASP lista como medidas:

- O tamanho do buffer deve ser verificado quanto a sua especificação;
- Memória alocada deve ser liberada apropriadamente após a finalização dos pontos de saída das funções;
- Não confiar em coleta de lixo em recursos próximos como objetos de conexão, identificadores de arquivos, etc; e
- Em chamadas de loop os limites de buffer devem ser verificados para que não seja permitida a gravação no espaço alocado;

A integração das abordagens acima mencionadas em aplicativos e sistemas online e IoT quando implementadas na produção e adotadas como padrão de desenvolvimento ajuda na proteção quanto à tríade confidencialidade, integridade e disponibilidade.

As vulnerabilidades de código possibilitam que ataques cibernéticos ocorram nos sistemas de informação. Erros de programação e bugs ocorrem no processo de desenvolvimento de software e são tratados nas fases de testes. Porém, testes que envolvam cenários relacionados a segurança nos sistemas desenvolvidos, em uma abordagem tradicional, geralmente não são tratados em processos de desenvolvimento.

No caso de testes em dispositivos de IoT, se faz necessário uma gama de ferramentas que possam disponibilizar dados de vulnerabilidades nesses equipamentos sem análise prévia dos mesmos, quando aplicados numa abordagem *Security by Design*. Não há ferramenta especifica (desenvolvida para o negócio analisado) que possibilite a realização de testes de forma precisa e em processo de desenvolvimento sem que se recorra a métodos externos de análise.

Uma organização que se dedica em analisar e disseminar temas, documentos e ferramentas com relação à segurança é a fundação OWASP. A organização publica as principais vulnerabilidades em seu projeto com foco em IoT. Para o ano de 2018 a OWASP indica os 10 principais problemas associados a segurança na IoT que as empresas devem levar em consideração:

01 – Senhas Fracas ou previsíveis na codificação: Senhas que não permitem modificações e tem acesso público facilitado por meio de técnicas de força bruta e/ou backdoors em firmware ou em software cliente com permissão não autorizada em sistemas.

02 – Redes com serviços inseguros: Dispositivos expostos na internet com serviços de rede configurados sem necessidade e de modo inseguro com permissão de controle não autorizado e que comprometem a confidencialidade, autenticidade e disponibilidade.

03 – Ecossistemas com interfaces sem segurança: Interfaces Web, Móveis, em Nuvem ou APIs para backends nos ecossistemas ao redor dos dispositivos e/ou fora deles que apresentam problemas de segurança devem ser corrigidos.

04 – Mecanismos sem atualização de modo seguro: Atualizações complexas nos dispositivos. Problemas nas validações de firmware dos dispositivos, tráfego sem criptografia no envio (falha de segurança). Sistemas sem procedimento de backup ou notificação de modificações de segurança devido às atualizações.

05 – Utilização de componentes desatualizados e sem segurança: Bibliotecas ou componentes de softwares inseguros podem comprometer o dispositivo. Utilização de software de terceiros ou hardware comprometido.

06 – Privacidade com proteção insuficiente: Usuários com dados gravados direto nos dispositivos, sem permissão, sem segurança e de modo inadequado.

07 – Armazenamento de Dados e Transferência insegura: Dados de confiança dentro do ecossistema armazenados em processamento ou em trânsito sem criptografia ou controle de acesso adequados.

08 – Falta de controles de gerenciamento: Dispositivos em produção sem suporte para segurança como gerenciamento dos ativos, sem gerenciamento das atualizações, desarme com segurança e monitoramento dos sistemas e respostas sem recursos.

09 – Configuração insegura por padrão: Configuração padronizada dos dispositivos e/ou sistemas com pouca segurança, sem aplicação de restrições com base em alteração de configuração.

10 – Falta de *Hardening:* Os dispositivos não contam com proteção com relação a sua estrutura física, permitindo aos cibercriminosos acesso a dados confidenciais que possam possibilitar o controle local dos equipamentos.

Para o ano de 2021 a OWASP listou as 10 principais vulnerabilidades por ordem de relevância e importância:

01 - Quebra de Controle de Acesso: houveram comprometimento em 94% dos sistemas nos testes realizados pela OWASP, de alguma maneira, nos controles de acesso dos sistemas. As quebras de Controle de Acesso ocorreram em maior parte em aplicativos.

02 - Falhas de Criptografia: falhas relacionadas a criptografia levam a exposição de dados confidenciais e comprometem todo o sistema.

03 – Injeção: foram encontradas algumas formas de injeção em 94% dos testes efetuados.

04 – Design inseguro: riscos com relação a falhas de design, encontrados em 2021. Informa que as falhas ocorrem em padrões e princípios de designer inseguro incluindo arquiteturas de referência.

05 – Configuração de segurança incorreta: 90% dos aplicativos testados, segundo a OWASP, tinham alguma configuração incorreta. Isso é devido às alterações e inovações em software que cada vez mais são configuráveis.

06 – Componentes desatualizados: um problema já reportado pela OWASP, pois é algo que apresenta dificuldades em testes.

07 – Quebra de autenticação: pelo grande número de padronizações ainda ocorrem falhas com relação a identificação.

08 – Falhas de software e integridade de dados: ocorrência de vulnerabilidades decorrentes de suposições feitas em relação a atualizações de software sem que haja verificação de integridade.

09 – Falhas de registro e monitoramento de segurança: apontam erros na visibilidade, monitoramento, e alertas de incidentes.

10 – *Server-Side Request Forgery*: Taxas baixas de incidência de dados e com cobertura de testes acima da média que geram classificações de riscos acima da média. Profissionais da indústria apontam riscos de segurança, mas não há ilustração nos dados até o momento conforme a OWASP.

No site da OWASP (https://owasp.org/) são encontradas as métricas e padrões e como os riscos de segurança foram classificados. Mais de 500 mil aplicações, de diversas organizações, algumas anônimas, forneceram dados para a geração das vulnerabilidades apontadas. As listas apontadas pela OWASP compõem grandes categorias de riscos que englobam diversas vulnerabilidades. Essa lista tem como base dados reais que os profissionais devem seguir. Fornecem acompanhamento com base cientifica no qual são encontrados os principais problemas de segurança web e que valem ser estudadas. Visam a orientação que inclui parâmetros mínimos de segurança no processo de desenvolvimento de software. A listagem da OWASP auxilia os desenvolvedores a terem um norteamento quanto ao desenvolvimento de sistemas com segurança.