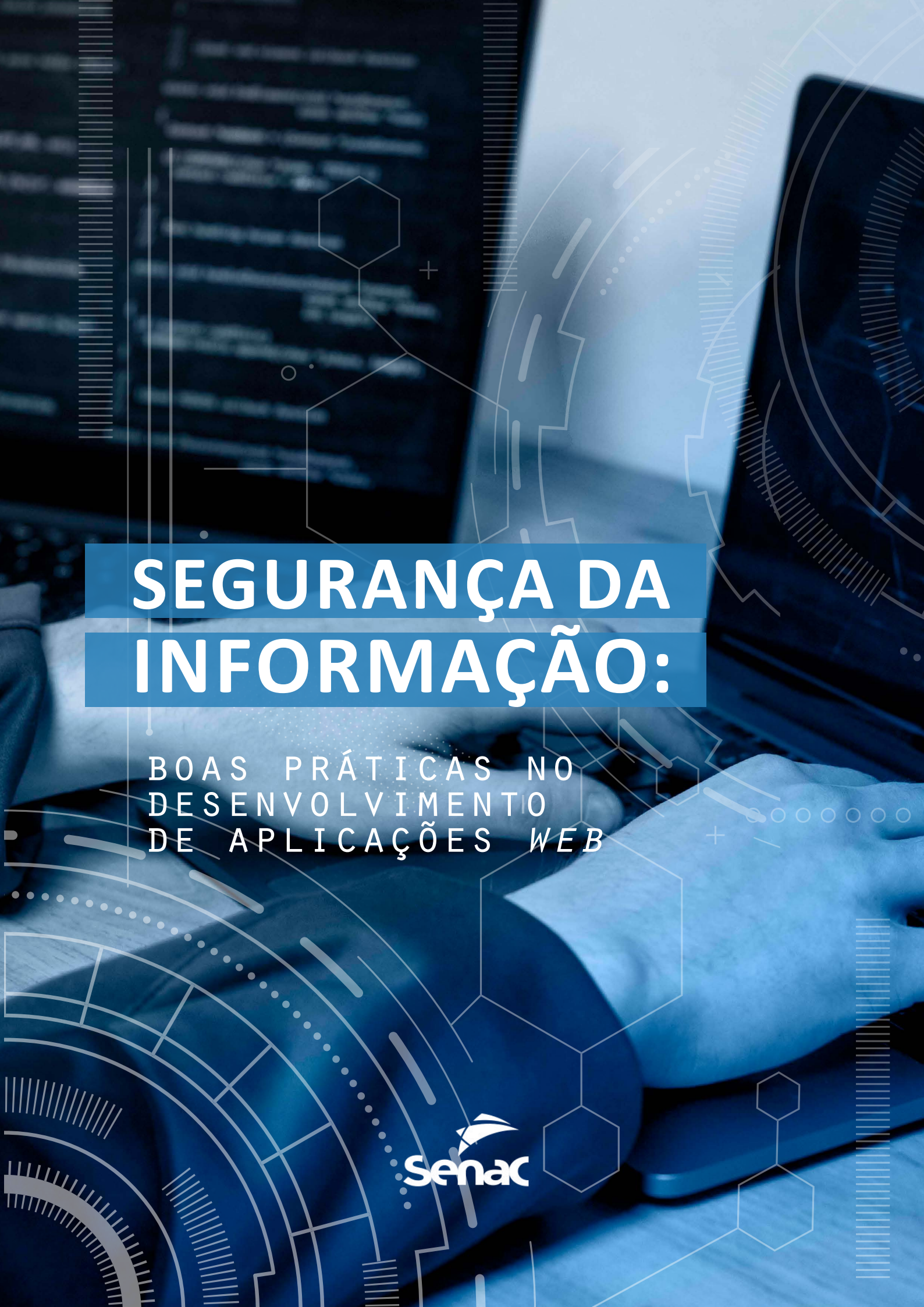
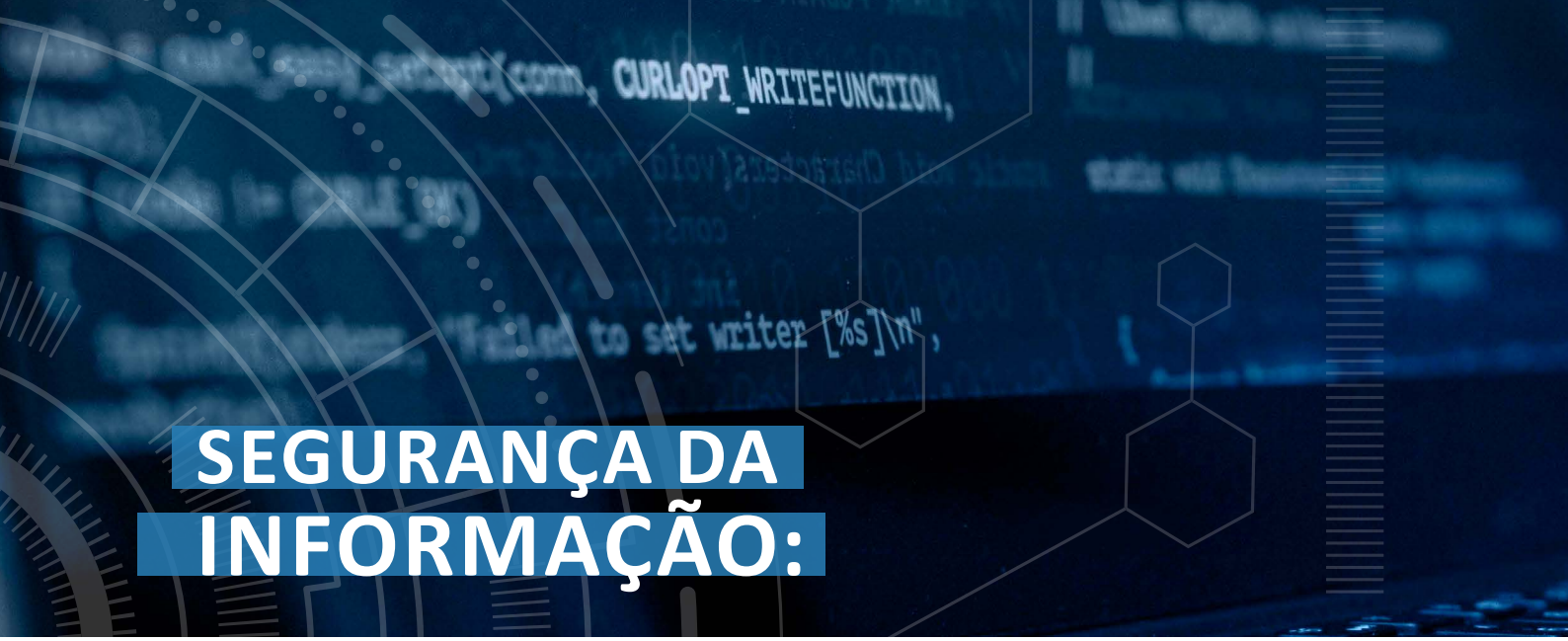


SEGURANÇA DA INFORMAÇÃO:

BOAS PRÁTICAS NO DESENVOLVIMENTO DE APLICAÇÕES *WEB*

Senac



SEGURANÇA DA INFORMAÇÃO:

BOAS PRÁTICAS NO DESENVOLVIMENTO DE APLICAÇÕES *WEB*

Problemas de segurança em *softwares* podem afetar a vida das pessoas com relação a aspectos financeiros, de energia, de saúde etc. Está cada vez mais difícil garantir a segurança devido à alta demanda de *softwares* e ao aumento de complexidade e conectividade.

Um código perfeitamente funcional e confiável não é necessariamente seguro. A simples utilização de protocolos de páginas seguras (quando aparecem o cadeado fechado e o termo “https:” nos navegadores de Internet) e de funcionalidades de *log in* pelos desenvolvedores não garante segurança.

Existem várias entidades envolvidas em segurança de *software* espalhadas pelo mundo. Dentre elas, pode-se citar o OWASP (Open Web Application Security Project). Esse grupo de trabalho não tem fins lucrativos, não está associado a nenhuma empresa, trabalha com voluntários e disponibiliza gratuitamente ferramentas, vídeos e documentos atualizados relativos à segurança de *software*.

Vários produtos são desenvolvidos pelo OWASP. Contudo, o que tem mais destaque é o **Top 10.** O Top 10 é um trabalho incessante e de qualidade indiscutível que lista **os dez maiores problemas relativos a vulnerabilidades de segurança e as recomendações práticas para adotar em qualquer tipo de *software* e desenvolvimento de *sites*.**

Tal lista cobre de 80% a 90% das ameaças e dos ataques mais comuns, o que a torna um padrão para o desenvolvimento e a manutenção de *software* no mundo. Observe a seguir um resumo dessa lista de falhas e soluções:

1 Injeção de dados

A injeção de dados ocorre quando dados não confiáveis são enviados indevidamente a serviços de banco de dados (injeção de SQL – *structured query language*) e a servidores de autenticação. Esses dados "injetados" levam os servidores a responderem indevidamente, fornecendo informações não autorizadas. Como forma de prevenção, devem-se usar sistemas de validação de dados e tratamento de caracteres especiais na consulta ao serviço.

Em 2008, o sistema de pagamento *on-line* Heartland Payment Systems sofreu um ataque por injeção de SQL, expondo 134 milhões de cartões de crédito. O curioso é que a empresa demorou quase um ano para identificar o problema e acabou pagando cerca de 145 milhões de dólares em compensações. Os responsáveis pelo ataque foram condenados a 20 anos de prisão.

2 Quebra de autenticação

A quebra de autenticação acontece quando um invasor consegue acessar o sistema explorando falhas de implementação de senhas, chaves, *tokens* de sessão etc. Como forma de prevenção, devem-se implementar sistemas de autenticação múltipla, reCAPTCHA, verificação de senhas fracas, limites de tentativas de autenticação etc.

3 Exposição de dados sensíveis

Os problemas são as aplicações que não protegem os dados (pessoais, financeiros, entre outros) dos usuários. Tais dados podem ser usados por pessoas mal-intencionadas para clonar cartões de crédito etc. Como forma de prevenção, deve-se utilizar criptografia para o armazenamento e a transferência de dados sensíveis e guardar o mínimo necessário destes, descartando o que não será utilizado.

Um dos casos mais notórios de exposição de dados, que veio à tona em 2018, foi o escândalo da Cambridge Analytica, no qual a empresa coletou ilegalmente dados sensíveis de usuários do Facebook. A motivação era influenciar a campanha política presidencial dos Estados Unidos em 2016.

4 Entidades externas de XML

XML (*extensible markup language*) é uma linguagem de marcação usada para construir e descrever documentos. Se não for configurada corretamente, ela pode permitir o acesso a arquivos internos, portas de comunicação, execução de código remoto etc. Como forma de prevenção, não se deve utilizar o processamento externo de XML. Caso seja realmente necessário, devem-se atualizar as bibliotecas e implementar validações e filtragem de dados.

5 Quebra de controle de acessos

As verificações de restrição de acesso dos usuários nem sempre estão configuradas corretamente. Essas falhas permitem que um atacante acesse arquivos, modifique permissões e cause muitos estragos. Como forma de prevenção, devem-se utilizar códigos confiáveis para controlar os acessos e estabelecer modelos mais seguros desse controle.

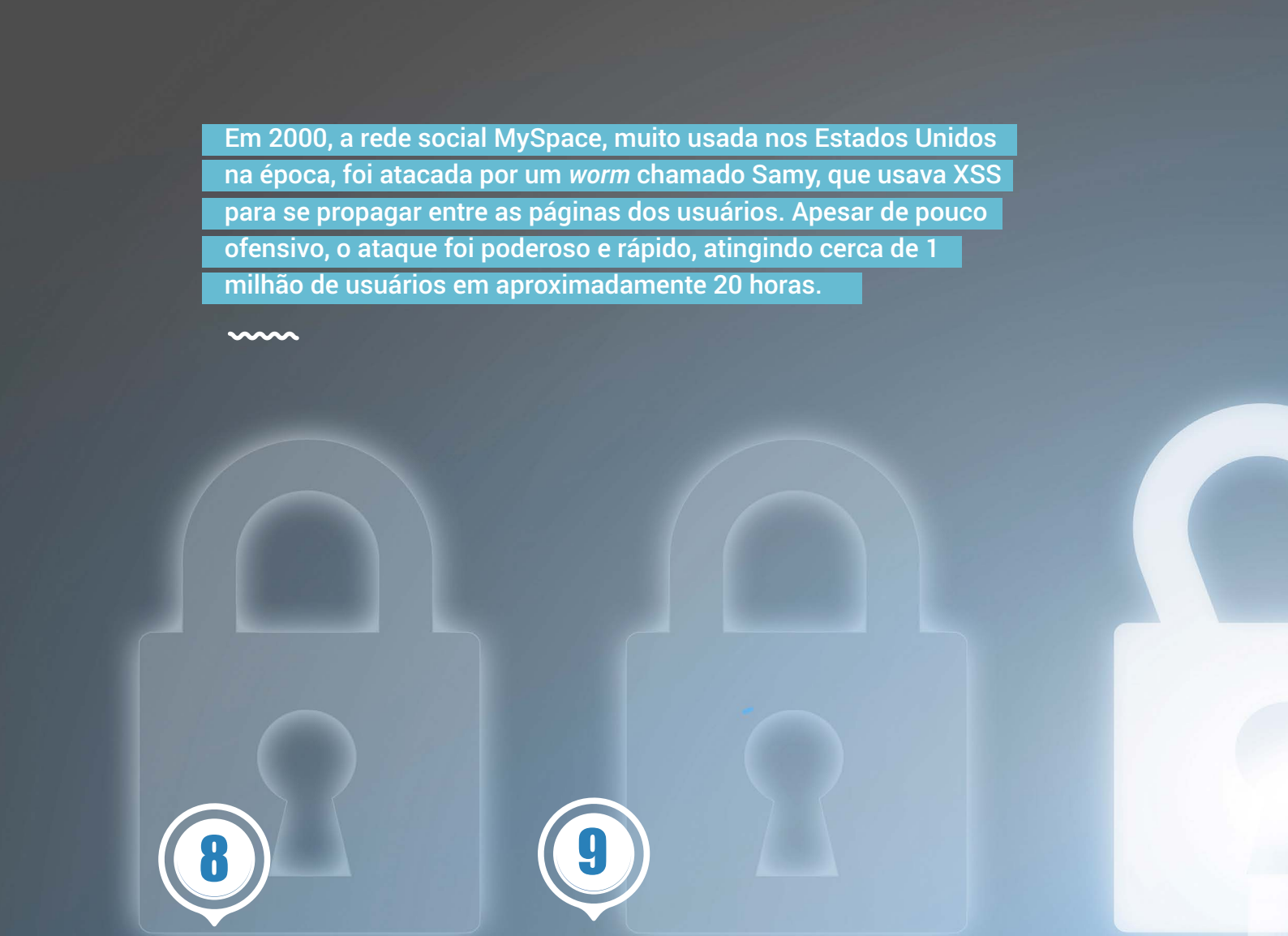
6 Configurações de segurança incorretas

As configurações de segurança incorretas são os erros mais comuns (falta de segurança no acesso à nuvem, cabeçalhos http, sistemas operacionais, *frameworks* mal configurados). Como forma de prevenção, deve-se utilizar uma plataforma mínima e sempre atualizada para obter os resultados desejados.

7 XSS

XSS (*cross-site scripting*) é um tipo de ataque de injeção de código malicioso em uma aplicação *web* que atinge o navegador do usuário, podendo acessar dados deste ou redirecioná-lo para *sites* perigosos sem que ele perceba. Como forma de prevenção, devem-se utilizar *frameworks* mais seguros, além de tratar adequadamente as informações não confiáveis direto no código http.

Em 2000, a rede social MySpace, muito usada nos Estados Unidos na época, foi atacada por um *worm* chamado Samy, que usava XSS para se propagar entre as páginas dos usuários. Apesar de pouco ofensivo, o ataque foi poderoso e rápido, atingindo cerca de 1 milhão de usuários em aproximadamente 20 horas.



8

Desserialização insegura

A serialização é utilizada para que haja comunicação entre os processos dos aplicativos e entre clientes e servidores. Se esse serviço não for seguro, há a possibilidade de execução de código remoto e acessos indevidos por meio de ataques de vários tipos. Como forma de prevenção, deve-se certificar que as fontes de dados são confiáveis. Para tanto, podem-se utilizar verificações de integridade usando assinaturas digitais e políticas que definem com rigor os tipos de dados que trafegam pelo sistema.

9

Utilização de componentes vulneráveis

Qualquer módulo de *software*, *framework* ou biblioteca que é executado tem os mesmos privilégios da aplicação principal. Assim, qualquer vulnerabilidade em um desses objetos pode afetar diretamente todo o sistema. Isso pode ocasionar desde a perda de dados até o controle completo do servidor. Como forma de prevenção, deve-se eliminar tudo que não seja absolutamente necessário para a aplicação. Para tanto, podem-se utilizar somente componentes confiáveis e implementar algum tipo de monitoramento quanto às atualizações de segurança.

10 Registro e monitoramento insuficientes

O monitoramento e o registro de incidente, quando feitos insuficientemente ou, pior, quando inexistentes, podem acarretar ataques persistentes e massivos. Assim, é apenas uma questão de tempo para que as invasões se tornem efetivas e o atacante tome conta do sistema, podendo extrair, apagar ou modificar dados. Como forma de prevenção, deve-se implementar o monitoramento com alertas de todas as autenticações, falhas no controle de acessos, integridade e validação de dados.

Em 2014, o *site* de compra e venda eBay sofreu um ataque que expôs dados de 145 milhões de usuários. Segundo a empresa, o ataque partiu de *hackers* que conseguiram acesso por meio de credenciais de funcionários da eBay. A ação durou 229 dias, tempo suficiente para alcançar os bancos de dados de usuários.

O OWASP disponibiliza uma **tabela de classificação** de riscos para que o desenvolvedor defina as probabilidades de ocorrência e os impactos sobre a sua empresa. Cada empresa é única, e os impactos podem ser mínimos ou até mesmo destruir totalmente o empreendimento.

A comunidade também oferece documentos completos descrevendo os próximos passos para os programadores a fim de produzir uma aplicação web segura. Entre os documentos, estão os **requisitos de segurança da aplicação; arquitetura de segurança das aplicações; controles de padrão da segurança; ciclo de vida de desenvolvimento seguro etc**.

Existe um consenso entre os programadores, no qual o **desenvolvimento seguro de aplicações *web*** pode ser resumido em três grandes pilares:

1.Gerenciamento e controle de acessos

Gerenciar e controlar acessos são técnicas que impedem que alguém consiga acessar indevidamente o sistema. Por meio do uso correto de autenticação e autorização, consegue-se garantir o acesso devido. Entende-se por autenticação a garantia de que o usuário é realmente quem diz ser. Para tal, implementa-se o uso da dupla usuário e senha, preferencialmente em conjunto com outro mecanismo auxiliar, como certificados digitais, reCAPTCHA, *smartcards* e até sensores biométricos. Já a autorização refere-se às permissões de acesso e às restrições para determinados usuários.

2. Validação das informações recebidas

Validar as informações recebidas consiste em validar e tratar o fluxo de informações entre o usuário e a aplicação. Deve-se sempre ter em mente que toda informação pode ser insegura ou maliciosa. As técnicas mais usadas são a validação semântica e a sanitização. A validação semântica valida as informações do ponto de vista lógico, ou seja, verifica se números ou letras recebidos são realmente números ou letras, evitando erros de digitação. A sanitização "limpa" as informações recebidas, evitando possíveis trechos de código que poderiam ser maliciosos. Essa limpeza pode ocorrer nos dois sentidos, protegendo tanto o usuário usando o seu navegador quanto a aplicação no lado do servidor.

3. Gerenciamento dos ataques sofridos

O gerenciamento dos ataques sofridos engloba todo o monitoramento de eventos, o tratamento de erros, a geração de relatórios e os alertas de segurança. Um desenvolvedor consciente deve sempre ter em mente que qualquer aplicativo pode ser alvo das mais variadas técnicas de invasão. Portanto, é muito importante implementar técnicas para lidar com esses ataques. Qualquer aplicação pode ter erros, e estes devem ser tratados adequadamente. Os relatórios de erro devem ser detalhados, para que seja possível identificar rapidamente a causa e a posterior solução.

Além disso, é fundamental que o relatório mostre erros de autenticação, transações financeiras, tentativas de acesso bloqueadas e detecção de dados maliciosos. Por fim, os mecanismos de alerta em tempo real devem ser disparados sempre que houver algum uso de recursos anormal dos servidores, regras de negócios sendo utilizadas fora do normal e requisições contendo dados maliciosos ou informações sendo manipuladas indevidamente.

Além dessas proteções, o uso de um WAF (*web application firewall*) pode incrementar significativamente a segurança do servidor de aplicação. WAFs, tais como o ModSecurity, ajudam a deter as tentativas de exploração de vulnerabilidades mais conhecidas e são reconhecidos por manter total segurança seguindo as regras da comunidade OWASP, conforme descrito anteriormente.

A aplicação de boas práticas para o desenvolvimento *web* garante alta segurança, mas a manutenção é essencial.

CRIPTOGRAFIA

A palavra “criptografia” origina-se do grego e significa “escrita secreta”. Para os profissionais da área da informática, significa a **técnica de transformar mensagens para torná-las seguras e imunes a ataques**, ou seja, tais mensagens não podem ser modificadas sem deixar rastros. Para iniciar, veja a seguir alguns conceitos:

- **TEXTO CLARO E TEXTO CIFRADO:** o texto claro é a mensagem original, legível, que é transformada por meio de um algoritmo de criptografia no texto cifrado. Para que a mensagem volte ao normal (ou texto claro), usa-se um algoritmo de descriptografia aplicado no texto cifrado. O algoritmo falado anteriormente, também chamado de cifra, nada mais é do que um programa de computador que faz as funções criptográficas.
- **CHAVE CRIPTOGRÁFICA:** a chave criptográfica é um número ou um conjunto de números e letras que, juntamente com a cifra, criptografa e descriptografa as mensagens. A chave tem que ser usada tanto no emissor da mensagem (para criptografar) quanto no receptor (para descriptografar).

Existem duas grandes categorias de algoritmos de criptografias. Veja-as a seguir.

Criptografia de chave simétrica

Além de ser a forma mais usada, na criptografia de chave simétrica utiliza-se a mesma chave para criptografar e descriptografar as mensagens. Nesse caso, a chave deve ser compartilhada, e os dois algoritmos devem usar a mesma chave.

Fazendo uma analogia, seria o mesmo que enviar uma carta dentro de uma caixa fechada com um cadeado. Somente quem tiver a chave pode ter acesso à carta.

Observe a seguir a figura 1.

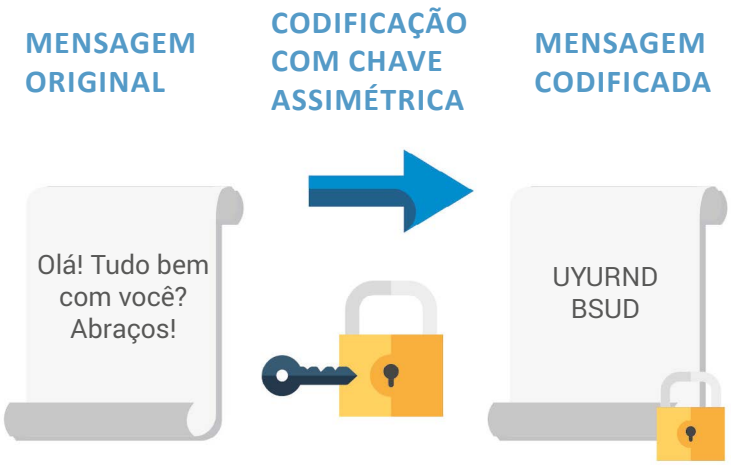


Figura 1 – Criptografia simétrica

Fonte: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/hugo/Criptografia.html>. Acesso em: 19 ago. 2019.

Tal método apresenta uma vantagem: o baixo uso de recursos para o processo que envolve a codificação e a decodificação. A grande desvantagem é a segurança necessária para enviar a chave para o receptor, pois qualquer um que tiver a chave pode abrir a mensagem. Alguns exemplos de algoritmos são: DES (*data encryption standard*), AES (*advanced encryption standard*), IDEA (*international data encryption algorithm*), os quais **são muito** utilizados em redes Wi-Fi.

Criptografia de chave assimétrica

O problema do compartilhamento de chave é contornado por uma técnica chamada **chave pública**. Nela, cada usuário utiliza duas chaves: a pública e a privada. A chave privada é guardada somente pelo seu dono. Já a chave pública fica acessível para qualquer um. Para enviar uma mensagem, o transmissor faz a codificação usando a chave pública do receptor.

Observe a figura 2. A chave azul é a chave pública do receptor que o transmissor usa para codificar a mensagem.

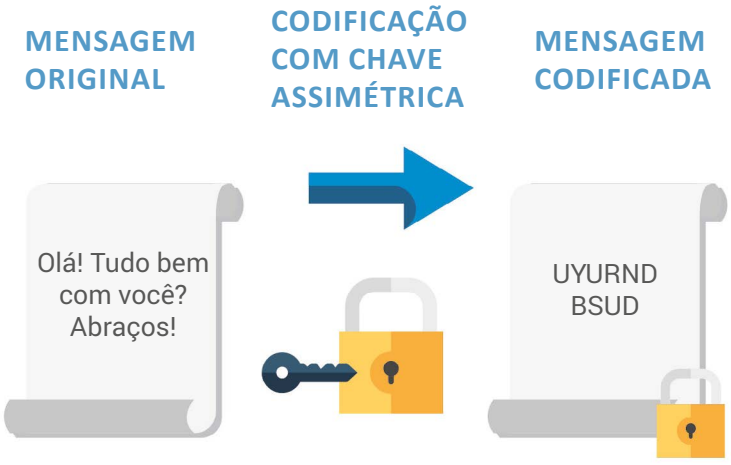


Figura 2 – Criptografia assimétrica (processo de codificação pelo transmissor)
Fonte: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/hugo/Criptografia.html>. Acesso em: 19 ago. 2019.

Para decodificar a mensagem depois de recebida, o receptor usa a sua chave privada, que tem correspondência com a sua chave pública utilizada para codificar pelo transmissor. Na figura 3, a chave privada está representada pela cor amarela. Assim, o receptor tem acesso à mensagem original com toda segurança, pois se alguém tentar decodificar a mensagem sem a chave privada não obterá êxito.

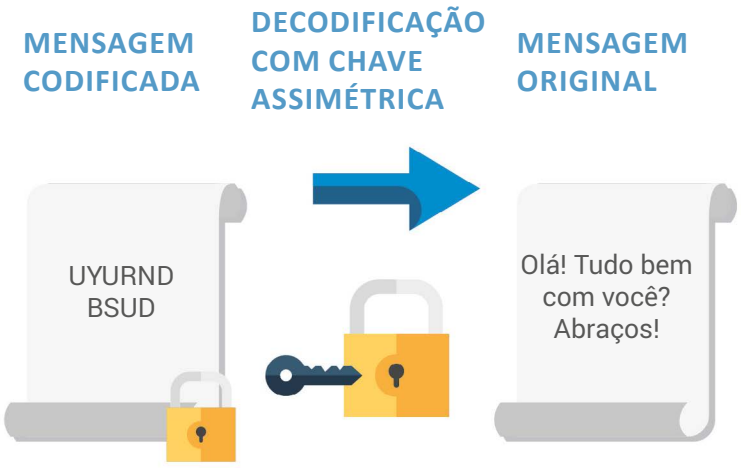


Figura 3 – Criptografia assimétrica (processo de decodificação pelo receptor)
Fonte: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/hugo/Criptografia.html>. Acesso em: 19 ago. 2019.

Na criptografia assimétrica, devido à complexidade das operações matemáticas envolvidas e à utilização de números primos muito grandes, o uso de recursos computacionais é elevado. Em contraponto, a segurança é muito maior, pois resolve o problema do envio seguro de chaves, como no caso das chaves simétricas.

Alguns exemplos são os algoritmos RSA (Rivest-Shamir-Adleman) e Diffie-Hellman, amplamente utilizados em inúmeros aplicativos. A correta combinação dessas duas técnicas e de suas variações garante confiabilidade nas comunicações via Internet e compreende os quatro princípios fundamentais de segurança. São eles:

- **CONFIDENCIALIDADE:** a informação apenas é legível para o transmissor e o receptor. Para os demais, é "lixo".
- **INTEGRIDADE:** a informação deve chegar totalmente intacta ao receptor. Qualquer modificação na informação original é detectada.
- **AUTENTICAÇÃO:** garante-se que o receptor reconheça o transmissor, não dando oportunidade para a ação de impostores.
- **NÃO REPÚDIO:** o transmissor não pode negar que emitiu uma mensagem. O receptor é quem tem que provar que o transmissor enviou a mensagem.

A CRIPTOGRAFIA RSA APOIA-SE ESSENCIALMENTE NA COMPLEXIDADE DO PROCESSAMENTO DE SUA DECODIFICAÇÃO (O QUE, DEPENDENDO DO TAMANHO DA CHAVE, LEVA DEZENAS OU MILHARES DE ANOS PARA SER DECODIFICADO). ESSE PANORAMA VEM MUDANDO COM A EVOLUÇÃO DA TECNOLOGIA E O APRIMORAMENTO DA CAPACIDADE DE PROCESSAMENTO DOS NOVOS EQUIPAMENTOS. UMA TECNOLOGIA EM ESPECIAL QUE PODE AMEAÇAR DEFINITIVAMENTE A SEGURANÇA DESSA CRIPTOGRAFIA É A COMPUTAÇÃO QUÂNTICA, A QUAL É CAPAZ, TEORICAMENTE, DE PROCESSAR MUITO MAIS OPERAÇÕES POR SEGUNDO DO QUE A TECNOLOGIA BINÁRIA UTILIZADA HOJE.

Atualmente, os *sites* seguros utilizam os protocolos SSL (*secure sockets layer*) e/ou TLS (*transport layer security*) para proporcionar uma camada muito segura e trafegar as informações entre os servidores e os navegadores. Para demonstrar para o usuário a correta aplicação desses protocolos, o famoso cadeado fechado deve aparecer na barra de endereços à esquerda da URL (*uniform resource locator*).