Application Security Verification Standard 4.0.3

Final

Outubro de 2021

Table of Contents

Frontispício

Sobre o Padrão

O Application Security Verification Standard é uma lista de requisitos ou testes de segurança de aplicações que podem ser usados por arquitetos, desenvolvedores, testadores, profissionais de segurança, fornecedores de ferramentas e consumidores para definir, construir, testar e verificar aplicações seguros.

Direitos autorais e licença

Versão 4.0.3, outubro de 2021

Líderes de projeto

Andrew van der Stock	Daniel Cuthbert	Jim Manico
Josh C Grossman	Elar Lang	

Principais colaboradores

Abhay Bhargav	Benedikt Bauer	Osama Elnaggar
Ralph Andalis	Ron Perris	Sjoerd Langkemper
Tonimir Kisasondi		

Outros colaboradores e revisores

Aaron Guzman	Alina Vasiljeva	Andreas Kurtz	Anthony Weems	Barbara Schachner
Christian Heinrich	Christopher Loessl	Clément Notin	Dan Cornell	Daniël Geerts
David Clarke	David Johansson	David Quisenberry	Elie Saad	Erlend Oftedal
Fatih Ersinadim	Filip van Laenen	Geoff Baskwill	Glenn ten Cate	Grant Ongers
hello7s	Isaac Lewis	Jacob Salassi	James Sulinski	Jason Axley
Jason Morrow	Javier Dominguez	Jet Anderson	jeurgen	Jim Newman
Jonathan Schnittger	Joseph Kerby	Kelby Ludwig	Lars Haulin	Lewis Ardern
Liam Smit	lyz-code	Marc Aubry	Marco Schnüriger	Mark Burnett
Philippe De Ryck	Ravi Balla	Rick Mitchell	Riotaro Okada	Robin Wood
Rogan Dawes	Ryan Goltry	Sajjad Pourali	Serg Belkommen	Siim Puustusmaa
Ståle Pettersen	Stuart Gunter	Tal Argoni	Tim Hemel	Tomasz Wrobel
Vincent De Schutter	Mike Jang			

Se um crédito faltar na lista de créditos 4.0.3 acima, crie um ticket no GitHub para ser reconhecido em futuras atualizações.

O Application Security Verification Standard é construído com base no trabalho dos envolvidos desde o ASVS 1.0 em 2008 até o 3.0 em 2016. Grande parte da estrutura e itens de verificação que ainda estão no ASVS hoje foram originalmente escritos por Mike Boberski, Jeff Williams e Dave Wichers, mas há muitos mais contribuidores. Obrigado a todos os envolvidos. Para obter uma lista abrangente de todos aqueles que contribuíram para versões anteriores, consulte as versões anteriores.

Prefácio

Bem-vindo ao Application Security Verification Standard (ASVS) versão 4.0. O ASVS é um esforço orientado pela comunidade para estabelecer uma estrutura de requisitos e controles de segurança que se concentram na definição dos controles de segurança funcionais e não funcionais necessários ao projetar, desenvolver e testar aplicações e serviços da web modernos.

A versão 4.0.3 é o terceiro minor patch para a v4.0 destinado a corrigir erros ortográficos e tornar os requisitos mais claros sem fazer alterações significativas, como alterar substancialmente, fortalecer ou adicionar requisitos. No entanto, alguns destes podem ter sido ligeiramente enfraquecidos onde consideramos apropriado e alguns requisitos totalmente redundantes foram removidos (mas sem renumeração).

O ASVS v4.0 é o culminar do esforço da comunidade e do feedback da indústria na última década. Tentamos facilitar a adoção do ASVS para uma variedade de casos de uso diferentes em qualquer ciclo de vida de desenvolvimento de software seguro.

Esperamos que provavelmente nunca haverá 100% de concordância sobre o conteúdo de qualquer padrão de aplicação da web, incluindo o ASVS. A análise de risco é sempre subjetiva até certo ponto, criando um desafio ao tentar generalizar num padrão único. No entanto, esperamos que as atualizações mais recentes feitas nesta versão sejam um passo na direção certa e aprimorem os conceitos introduzidos neste padrão crítico do setor.

O que há de novo no 4.0

A mudança mais significativa nesta versão é a adoção da NIST 800-63-3 Digital Identity Guidelines, introduzindo controles de autenticação modernos e avançados, baseados em evidências. Embora esperemos algum retrocesso no alinhamento com um padrão de autenticação avançada, sentimos ser essencial que os padrões sejam alinhados, principalmente quando outro padrão de segurança de aplicações bem-conceituado é baseado em evidências.

Os padrões de segurança da informação devem tentar minimizar o número de requisitos exclusivos, para que as organizações em conformidade não tenham que decidir sobre controles concorrentes ou incompatíveis. O OWASP Top 10 2017 e agora o OWASP Application Security Verification Standard agora estão alinhados com o NIST 800-63 para autenticação e gestão de sessão. Incentivamos outros órgãos de definição de padrões a trabalhar conosco, NIST e outros para chegar a um conjunto geralmente aceito de controles de segurança de aplicações para maximizar a segurança e minimizar os custos de conformidade.

O ASVS 4.0 foi totalmente renumerado do início ao fim. O novo esquema de numeração permitiu-nos fechar lacunas de capítulos há muito desaparecidos e segmentar capítulos mais longos para minimizar o número de controles que um desenvolvedor ou equipe precisa cumprir. Por exemplo, se uma aplicação não usa JWT, toda a seção sobre JWT na gestão de sessão não é aplicável.

A novidade no 4.0 é um mapeamento abrangente para a Common Weakness Enumeration (CWE), uma das solicitações de recursos mais comumente desejadas que recebemos na última década. O mapeamento CWE permite que os fabricantes de ferramentas e aqueles que usam software de gestão de vulnerabilidade comparem os resultados de outras ferramentas e versões anteriores do ASVS para 4.0 e posteriores. Para abrir espaço para a input CWE, tivemos que retirar a coluna "Desde" que, como renumeramos completamente, faz menos sentido do que nas versões anteriores do ASVS. Nem todo item no ASVS tnum CWE associado e, como o CWE tem muita duplicação, tentamos usar o mais comumente usado em vez da correspondência mais próxima. Os controles de verificação nem sempre são mapeáveis para pontos fracos equivalentes.

Trabalhamos para atender e exceder de forma abrangente os requisitos para abordar o OWASP Top 10 2017 e o OWASP Proactive Controls 2018. Como o OWASP Top 10 2017 é o mínimo necessário para evitar negligência, fizemos deliberadamente todos, exceto os 10 principais requisitos de registro específicos. 1, facilitando para os adotantes do OWASP Top 10 avançarem para um padrão de segurança real.

Decidimos garantir que o ASVS 4.0 Nível 1 seja um superconjunto abrangente das Seções 6.5 do PCI DSS 3.2.1, para design de aplicações, codificação, teste, revisões de código seguro e testes de penetração. Isso exigia cobrir buffer overflow e operações de memória inseguras na V5, e sinalizadores de compilação relacionados à memória inseguras na V14, além dos requisitos existentes de verificação de serviços da Web e aplicações líderes do setor.

Concluímos a mudança do ASVS de controles monolíticos apenas do lado do servidor para fornecer controles de segurança para todas as aplicações e APIs modernos. Nos dias de programação funcional, API serverless, dispositivos móveis, nuvem, contêineres, CI/CD e DevSecOps, federação e muito mais, não podemos continuar ignorando a arquitetura moderna de aplicações. As aplicações modernas são projetados de maneira muito diferente daqueles criados quando o ASVS original foi lançado em 2009. O ASVS deve sempre olhar para o futuro para fornecer bons conselhos para nosso público principal - os desenvolvedores. Esclarecemos ou eliminamos qualquer requisito que suponha que as aplicações sejam executados em sistemas pertencentes a uma única organização.

Devido ao tamanho do ASVS 4.0, bem como ao nosso desejo de se torná-lo a linha de base para todos os outros esforços do mesmo, retiramos o capítulo sobre mobile, em favor do Mobile Application Security Verification Standard (MASVS). O apêndice Internet of Things aparecerá num futuro cuidado IoT ASVS do OWASP Internet of Things Project. Incluímos uma prévia do IoT ASVS no Apêndice C. Agradecemos tanto à equipe OWASP Mobile quanto à equipe do projeto OWASP IoT pelo suporte ao ASVS e esperamos trabalhar com eles no futuro para fornecer padrões complementares.

Por fim, desduplicamos e removemos os controles menos impactantes. Com o tempo, o ASVS passou a ser um conjunto abrangente de controles, mas nem todos os controles são iguais na produção de software seguro. Esse esforço para eliminar itens de baixo impacto poderia ir além. Em uma edição futura do ASVS, o Common Weakness Scoring System (CWSS) ajudará a priorizar ainda mais os controles que são realmente importantes e aqueles que devem ser retirados.

A partir da versão 4.0, o ASVS se concentrará exclusivamente em ser o padrão líder de aplicações e serviços da Web, abrangendo arquitetura de aplicações tradicionais e modernos, práticas de segurança ágeis e cultura DevSecOps.

Usando o ASVS

A ASVS tem dois objetivos principais:

- ajudar as organizações a desenvolver e manter aplicações seguros.
- permitir que fornecedores de serviços de segurança, fornecedores de ferramentas de segurança e consumidores alinhem seus requisitos e ofertas.

Níveis do Application Security Verification

O Application Security Verification Standard define três níveis de verificação de segurança, com cada nível aumentando em profundidade.

- O nível 1 do ASVS é para níveis de garantia baixos e é completamente testável quanto à penetração
- ASVS Nível 2 é para aplicações que contêm dados confidenciais, que requerem proteção e é o nível recomendado para a maioria das aplicações
- O nível 3 do ASVS é para as aplicações mais críticos - aplicações que executam transações de alto valor, contêm dados médicos confidenciais ou qualquer aplicação que exija o mais alto nível de confiança.

Cada nível ASVS contém uma lista de requisitos de segurança. Cada um desses requisitos também pode ser mapeado para recursos e capacidades específicos de segurança que devem ser incorporados ao software pelos desenvolvedores.

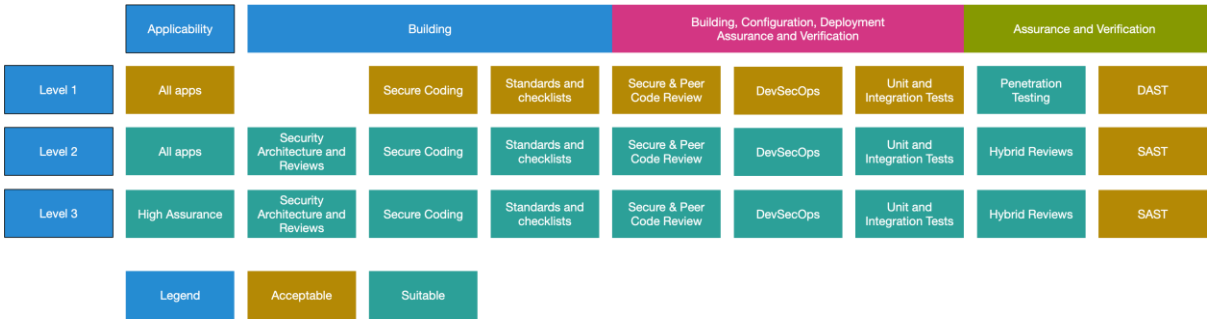

Figura 1 - Níveis 4.0 do Padrão de Application Security Verification OWASP

O nível 1 é o único nível que é completamente testável por penetração usando humanos. Todos os outros requerem acesso à documentação, código-fonte, configuração e pessoas envolvidas no processo de desenvolvimento. No entanto, mesmo que L1 permita a ocorrência de testes de "black box" (sem documentação e sem fonte), não é uma atividade de garantia eficaz e deve ser ativamente desencorajada. Os atacantes mal-intencionados têm muito tempo, a maioria dos testes de penetração termina em algumas semanas. Os defensores precisam incorporar controles de segurança, proteger, encontrar e resolver todos os pontos fracos, detectar e responder a agentes mal-intencionados num tempo razoável. Atores mal-intencionados têm tempo essencialmente infinito e requerem apenas uma única defesa porosa, uma única fraqueza ou detecção ausente para serem bem-sucedidos. O teste black box, muitas vezes realizado no final do desenvolvimento, rapidamente, ou não, são completamente incapazes de lidar com essa assimetria.

Ao longo dos últimos 30 anos, os testes black box provaram repetidas vezes perder problemas críticos de segurança que levaram diretamente a violações cada vez mais massivas. Incentivamos fortemente o uso de uma ampla gama de garantia e verificação de segurança, incluindo a substituição de testes de penetração por testes de penetração conduzidos por código-fonte (híbridos) no Nível 1, com acesso total aos desenvolvedores e à documentação durante todo o processo de desenvolvimento. Os reguladores financeiros não toleram auditorias financeiras externas sem acesso aos livros contábeis, amostras de transações ou pessoas que executam os controles. A indústria e os governos devem exigir o mesmo padrão de transparência no campo da engenharia de software.

Incentivamos fortemente o uso de ferramentas de segurança no próprio processo de desenvolvimento. As ferramentas DAST e SAST podem ser usadas continuamente pelo pipeline de construção para facilitar a localização de problemas de segurança que nunca deveriam estar presentes.

Ferramentas automatizadas e varreduras online não conseguem concluir mais da metade do ASVS sem assistência humana. Se for necessária uma automação de teste abrangente para cada compilação, será usada uma combinação de testes personalizados de unidade e integração, com varreduras online iniciadas por compilação. Falhas de lógica de negócios e testes de controle de acesso só são possíveis com assistência humana. Estes devem ser transformados em testes de unidade e integração.

Como usar este padrão

Uma das melhores maneiras de usar o Application Security Verification Standard é usá-lo como um modelo para criar uma Lista de Verificação de Codificação Segura específica para sua aplicação, plataforma ou organização. Adaptar o ASVS aos seus casos de uso aumentará o foco nos requisitos de segurança mais importantes para seus projetos e ambientes.

Nível 1 - Primeiras etapas, automatizada ou visão completa do portfólio

Uma aplicação atinge o nível 1 do ASVS se se defender adequadamente contra vulnerabilidades de segurança de aplicações que são fáceis de descobrir e incluídas no OWASP Top 10 e outras listas de verificação semelhantes.

O nível 1 é o mínimo que todas as aplicações devem buscar. Também é útil como uma primeira etapa num esforço multifásico ou quando as aplicações não armazenam ou manipulam dados confidenciais e, assim, não precisam de controles mais rigorosos de nível 2 ou 3. Os controles de nível 1 podem ser verificados automaticamente por ferramentas ou simplesmente manualmente, sem acesso ao código-fonte. Consideramos o Nível 1 o mínimo exigido para todas as aplicações.

Ameaças à aplicação provavelmente virão de invasores que usam técnicas simples e de baixo esforço para identificar vulnerabilidades fáceis de encontrar e explorar. Isso contrasta com um invasor determinado que gastará energia focada para atingir especificamente a aplicação. Se os dados processados pela sua aplicação tiverem alto valor, raramente irá parar numa revisão de nível 1.

Nível 2 - A maioria das aplicações

Uma aplicação atinge o nível 2 (ou padrão) do ASVS se se defender adequadamente contra a maioria dos riscos associados ao software atualmente.

O nível 2 garante que os controles de segurança estejam em vigor, sejam eficazes e sejam usados na aplicação. O nível 2 é normalmente apropriado para aplicações que lidam com transações business-to-business significativas, incluindo aquelas que processam informações de assistência médica, implementam funções críticas, ou confidenciais de negócios, ou processam outros ativos confidenciais, ou setores onde a integridade é uma faceta crítica para proteger os seus negócios, como a indústria de jogos para impedir trapaceiros e hacks de jogos.

Ameaças aas aplicações de nível 2 normalmente são invasores habilidosos e motivados com foco em alvos específicos usando ferramentas e técnicas que são altamente praticadas e eficazes na descoberta e exploração de pontos fracos nas aplicações.

Nível 3 - Alto valor, alta garantia ou alta segurança

ASVS Nível 3 é o mais alto nível de verificação no ASVS. Este nível é normalmente reservado para aplicações que requerem níveis significativos de verificação de segurança, como aqueles que podem ser encontrados em áreas militares, de saúde e segurança, infraestrutura crítica, etc.

As organizações podem exigir ASVS Nível 3 para aplicações que executam funções críticas, onde a falha pode afetar significativamente as operações da organização e até mesmo a sua capacidade de sobrevivência. Um exemplo de orientação sobre a aplicação do ASVS Nível 3 é fornecido abaixo. Uma aplicação atinge o Nível 3 do ASVS (ou Avançado) se defender adequadamente contra vulnerabilidades de segurança de aplicações avançados e também demonstrar princípios de um bom design de segurança.

Uma aplicação no nível 3 do ASVS requer uma análise mais aprofundada da arquitetura, codificação e teste do que todos os outros níveis. Uma aplicação seguro é modularizado de forma significativa (para facilitar a resiliência, escalabilidade e, primeiro, camadas de segurança), e cada módulo (separado por conexão de rede e/ou instância física) cuida das suas próprias responsabilidades de segurança (defesa em profundidade), que precisam ser devidamente documentados. As responsabilidades incluem controles para garantir a confidencialidade (por exemplo, criptografia), integridade (por exemplo, transações, validação de input), disponibilidade (por exemplo, manipulação de carga normalmente), autenticação (incluindo entre sistemas), autorização e auditoria (logging).

Aplicando ASVS na Prática

Ameaças diferentes têm motivações diferentes. Alguns setores têm informações exclusivas e ativos de tecnologia e requisitos de conformidade regulamentar específicos do domínio.

As organizações são fortemente encorajadas a analisar profundamente as suas características de risco exclusivas com base na natureza dos seus negócios e, com base nesse risco e nos requisitos de negócios, determinar o nível ASVS apropriado.

Como fazer referência aos requisitos ASVS

Cada requisito possui um identificador no formato <chapter>.<section>.<requirement> onde cada elemento é um número, por exemplo: 1.11.3.

- O valor <chapter> corresponde ao capítulo de onde vem o requisito, por exemplo: todos os requisitos 1.#.# são do capítulo Arquitetura.
- O valor <section> corresponde à seção dentro desse capítulo onde o requisito aparece, por exemplo: todos os requisitos 1.11.# estão na seção Arquitetura de lógica de negócios do capítulo Arquitetura.
- O valor <requirement> identifica o requisito específico no capítulo e seção, por exemplo: 1.11.3 que a partir da versão 4.0.3 desta norma é:

Verifique se todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gestão de sessão e controle de acesso, são thread-safe e resistentes a condições de corrida de tempo de verificação e tempo de uso.

Os identificadores podem mudar entre as versões do padrão, portanto, é preferível que outros documentos, relatórios ou ferramentas usem o formato: v<version>-<chapter>.<section>.<requirement>, onde: 'versão' é a tag de versão ASVS. Por exemplo: v4.0.3-1.11.3 seria entendido como significando especificamente o 3.º requisito na seção 'Arquitetura de lógica de negócios' do capítulo 'Arquitetura' da versão 4.0.3. (Isto pode ser resumido como v<version>-<requirement_identifier>.)

Nota: O v que precede a parte da versão deve ser minúsculo.

Se os identificadores forem usados sem incluir o elemento v<version>, eles devem ser assumidos para se referir ao conteúdo mais recente do Application Security Verification Standard. Obviamente, à medida que o padrão cresce e muda, isso se torna problemático, e é por isso que escritores ou desenvolvedores devem incluir o elemento de versão.

As listas de requisitos ASVS são disponibilizadas em CSV, JSON e outros formatos que podem ser úteis para referência ou uso programático.

Avaliação e Certificação

Postura da OWASP em certificações ASVS e marcas de confiança

A OWASP, como uma organização sem fins lucrativos neutra no âmbito de fornecedores, atualmente não certifica nenhum fornecedor, verificador ou software.

Todas essas afirmações de garantia, marcas de confiança ou certificações não são oficialmente examinadas, registradas ou certificadas pela OWASP, portanto, uma organização que confia em tal visão precisa ser cautelosa com a confiança depositada em qualquer terceiro ou marca de confiança que reivindique a certificação ASVS.

Isso não deve inibir as organizações de oferecer tais serviços de garantia, desde que não reivindiquem a certificação OWASP oficial.

Orientação para organizações certificadoras

O Application Security Verification Standard pode ser usado como uma verificação de livro aberto da aplicação, incluindo acesso aberto e irrestrito a recursos-chave, como arquitetos e desenvolvedores, documentação do projeto, código-fonte, acesso autenticado a sistemas de teste (incluindo acesso a uma ou mais contas em cada função), especialmente para verificações L2 e L3.

Historicamente, os testes de penetração e as revisões de código seguro incluíram problemas “por exceção”, ou seja, apenas os testes com falha aparecem no relatório final. Uma organização certificadora deve incluir em qualquer relatório o escopo da verificação (especialmente se um componente-chave estiver fora do escopo, como a autenticação SSO), um resumo das descobertas da verificação, incluindo testes aprovados e reprovados, com indicações claras de como resolver o problema testes falhados.

Certos requisitos de verificação podem não ser aplicáveis à aplicação em teste. Por exemplo, se você fornecer uma API de camada de serviço sem estado snuma implementação de cliente para seus clientes, muitos dos requisitos do V3 Session Management não serão diretamente aplicáveis. Nesses casos, uma organização certificadora ainda pode reivindicar conformidade total com a ASVS, mas deve indicar em qualquer relatório um motivo para a não aplicabilidade de tais requisitos de verificação excluídos.

Manter papéis de trabalho detalhados, capturas de tela ou filmes, scripts para explorar um problema de maneira confiável e repetida e registros eletrônicos de teste, como interceptar logs de proxy e notas associadas, como uma lista de limpeza, é considerado uma prática padrão da indústria e pode ser realmente útil como prova das descobertas para os desenvolvedores mais duvidosos. Não basta simplesmente rodar uma ferramenta e reportar as falhas; isso não fornece (de forma alguma) evidência suficiente de que todos os problemas num nível de certificação foram testados e testados exaustivamente. Em caso de disputa, deve haver evidência de garantia suficiente para demonstrar que cada requisito verificado foi de fato testado.

Método de teste

As organizações certificadoras são livres para escolher o(s) método(s) de ensaio apropriado(s), mas devem indicá-los num relatório.

Dependendo da aplicação em teste e do requisito de verificação, diferentes métodos de teste podem ser usados para obter a mesma confiança nos resultados. Por exemplo, validar a eficácia dos mecanismos de verificação de input de uma aplicação pode ser analisado com um teste de penetração manual ou por análises de código-fonte.

O papel das ferramentas automatizadas de teste de segurança

O uso de ferramentas automatizadas de teste de penetração é incentivado para fornecer o máximo de cobertura possível.

Não é possível concluir totalmente a verificação ASVS usando apenas ferramentas automatizadas de teste de penetração. Embora a grande maioria dos requisitos em L1 possa ser realizada usando testes automatizados, a maioria geral dos requisitos não é passível de testes de penetração automatizados.

Observe que as linhas entre testes automatizados e manuais tornaram-se indistintas à medida que o setor de segurança de aplicações amadureceu. As ferramentas automatizadas são geralmente ajustadas manualmente

por especialistas e os testadores manuais geralmente utilizam uma ampla variedade de ferramentas automatizadas.

O papel do teste de penetração

Na versão 4.0, decidimos tornar o L1 completamente testável sem acesso ao código-fonte, documentação ou desenvolvedores. Dois itens de registro, necessários para cumprir o OWASP Top 10 2017 A10, exigirão entrevistas, capturas de tela ou outra coleta de evidências, assim como no OWASP Top 10 2017. No entanto, testar sem acesso às informações necessárias não é um método ideal de verificação de segurança, pois perde a possibilidade de revisar a fonte, identificar ameaças e controles ausentes e realizar um teste muito mais completo num período menor.

Sempre que possível, é necessário acesso a desenvolvedores, documentação, código e acesso a uma aplicação de teste com dados que não sejam de produção ao realizar uma avaliação L2 ou L3. Os testes de penetração feitos nesses níveis requerem esse nível de acesso, que chamamos "revisões híbridas" ou "testes de penetração híbridos".

Outros usos para o ASVS

Além de ser usado para avaliar a segurança de uma aplicação, identificamos vários outros usos potenciais para o ASVS.

Conforme orientação detalhada da arquitetura de segurança

Um dos usos mais comuns do Application Security Verification Standard é como um recurso para arquitetos de segurança. A Sherwood Applied Business Security Architecture (SABSA) carece de uma grande quantidade de informações necessárias para concluir uma revisão completa da arquitetura de segurança da aplicação. O ASVS pode ser usado para preencher essas lacunas, permitindo que os arquitetos de segurança escolham melhores controles para problemas comuns, como padrões de proteção de dados e estratégias de validação de input.

Como um substituto para listas de verificação de codificação segura prontas para uso

Muitas organizações podem se beneficiar da adoção do ASVS, escolhendo um dos três níveis ou bifurcando o ASVS e alterando o que é necessário para cada nível de risco de aplicação de uma maneira específica de domínio. Incentivamos esse tipo de bifurcação, desde que a rastreabilidade seja mantida, de modo que, se uma aplicação passou no requisito 4.1, isso significa o mesmo para cópias bifurcadas e para o padrão à medida que ele evolui.

Como um guia para testes automatizados de unidade e integração

O ASVS foi projetado para ser altamente testável, com a única exceção dos requisitos de arquitetura e código malicioso. Ao criar testes de unidade e integração que testam casos de fuzz e abuso específicos e relevantes, a aplicação torna-se quase autoverificável a cada compilação. Por exemplo, testes adicionais podem ser criados para o conjunto de testes para um controlador de login, testando o parâmetro username para nomes de usuários padrões comuns, enumeração de contas, força bruta, injeção de LDAP e SQL e XSS. Da mesma forma, um teste no parâmetro de senha deve incluir senhas comuns, tamanho da senha, injeção de byte nulo, remoção do parâmetro, XSS e muito mais.

Para treinamento de desenvolvimento seguro

ASVS também pode ser usado para definir características de software seguro. Muitos cursos de “codificação segura” são simplesmente cursos de hacking ético com uma leve mancha de dicas de codificação. Isso pode não necessariamente ajudar os desenvolvedores a escrever um código mais seguro. Em vez disso, os cursos de desenvolvimento seguro podem usar o ASVS com um forte foco nos controles proativos encontrados no ASVS, em vez das 10 principais coisas negativas a não fazer.

Como um driver para segurança de aplicações ágeis

O ASVS pode ser usado num processo de desenvolvimento ágil como um framework para definir tarefas específicas que precisam ser implementadas pela equipe para ter um produto seguro. Uma abordagem pode ser: começando com o Nível 1, verifique a aplicação ou sistema específico conforme os requisitos ASVS para o nível especificado, encontre quais controles estão faltando e crie tickets/tarefas específicos no backlog. Isso ajuda na priorização de tarefas específicas (ou preparação) e torna a segurança visível no processo ágil. Isso

também pode ser usado para priorizar tarefas de auditoria e revisão na organização, onde um requisito específico do ASVS pode ser um driver para revisão, refatoração ou auditoria para um membro específico da equipe e visível como "dívida" no backlog que precisa ser feito eventualmente.

Como uma estrutura para orientar a aquisição de software seguro

O ASVS é uma ótima estrutura para ajudar na aquisição segura de software ou na aquisição de serviços de desenvolvimento personalizados. O comprador pode simplesmente definir um requisito de que o software que deseja adquirir deve ser desenvolvido no nível X da ASVS e solicitar que o vendedor comprove que o software atende ao nível X da ASVS. Isso funciona bem quando combinado com o Anexo do Contrato de Software Seguro OWASP.

V1 Arquitetura, Design e Modelagem de Ameaças

Objetivo de controle

A arquitetura de segurança quase se tornou uma arte perdida em muitas organizações. Os dias do arquiteto corporativo já passaram na era do DevSecOps. O campo de segurança de aplicações deve acompanhar e adotar princípios de segurança ágeis enquanto reintroduz os principais princípios de arquitetura de segurança para profissionais de software. A arquitetura não é uma implementação, mas uma maneira de pensar sobre um problema que tem potencialmente muitas respostas diferentes e nenhuma resposta "correta". Com muita frequência, a segurança é vista como inflexível e exigindo que os desenvolvedores corrijam o código de uma maneira específica, quando os desenvolvedores podem conhecer uma maneira muito melhor de resolver o problema. Não existe uma solução única e simples para a arquitetura, e fingir o contrário é um desserviço ao campo da engenharia de software.

É provável que uma implementação específica de uma aplicação da Web seja revisada continuamente ao longo da sua vida útil. A arquitetura geral raramente mudará, mas evoluirá lentamente. A arquitetura de segurança é idêntica - precisamos de autenticação hoje, exigiremos autenticação amanhã e precisaremos dela daqui a cinco anos. Se tomarmos decisões sensatas hoje, podemos economizar muito esforço, tempo e dinheiro se selecionarmos e reutilizarmos soluções compatíveis com a arquitetura. Por exemplo, uma década atrás, a autenticação multifator era raramente implementada.

Se os desenvolvedores tivessem investido num modelo único e seguro de provedor de identidade, como a identidade federada SAML, o provedor de identidade poderia ser atualizado para incorporar novos requisitos, como conformidade com NIST 800-63, sem alterar as interfaces da aplicação original. Se muitas aplicações compartilharem a mesma arquitetura de segurança e, assim, o mesmo componente, todos se beneficiarão dessa atualização de uma só vez. No entanto, o SAML nem sempre permanecerá como a melhor ou mais adequada solução de autenticação - pode ser necessário trocá-lo por outras soluções à medida que os requisitos mudam. Mudanças como essa são complicadas, tão caras que exigem uma reescrita completa ou totalmente impossíveis sem arquitetura de segurança.

Neste capítulo, o ASVS cobre os aspectos primários de qualquer arquitetura de segurança sólida: disponibilidade, confidencialidade, integridade de processamento, não repúdio e privacidade. Cada um desses princípios de segurança deve ser integrado e inato a todos os aplicativos. É fundamental "mudar para a esquerda", começando com a capacitação do desenvolvedor com listas de verificação de codificação segura, orientação e treinamento, codificação e teste, construção, implantação, configuração e operações e terminando com testes independentes de acompanhamento para garantir que todos os controles de segurança estão presentes e funcionais. A última etapa costumava ser tudo o que fazíamos como indústria, mas isso não é mais suficiente quando os desenvolvedores colocam o código em produção dezenas ou centenas de vezes por dia. Os profissionais de segurança de aplicativos devem acompanhar as técnicas ágeis, o que significa adotar ferramentas de desenvolvedor, aprender a codificar e trabalhar com desenvolvedores, em vez de criticar o projeto meses depois que todos os outros já partiram.

V1.1 Ciclo de vida de desenvolvimento de software seguro

#	Descrição	L1	L2	L3	CWE
1.1.1	Verifique o uso de um ciclo de vida de desenvolvimento de software seguro que aborde a segurança em todos os estágios de desenvolvimento. (C1)		✓	✓	
1.1.2	Verifique o uso da modelagem de ameaças para cada alteração de design ou planejamento de sprint para identificar ameaças, planejar contramedidas, facilitar respostas apropriadas a riscos e orientar testes de segurança.		✓	✓	1053
1.1.3	Verifique se todas as histórias e recursos do usuário contêm restrições de segurança funcionais, como "Como usuário, devo poder visualizar e editar meu perfil. Não devo visualizar ou editar o perfil de outra pessoa"		✓	✓	1110

#	Descrição	L1	L2	L3	CWE
1.1.4	Verifique a documentação e justificativa de todos os limites de confiança, componentes e fluxos de dados significativos da aplicação.		✓	✓	1059
1.1.5	Verifique a definição e a análise de segurança da arquitetura de alto nível da aplicação e de todos os serviços remotos conectados. (C1)		✓	✓	1059
1.1.6	Verifique a implementação de controles de segurança centralizados, simples (economia de design), verificados, seguros e reutilizáveis para evitar controles duplicados, ausentes, ineficazes ou inseguros. (C10)		✓	✓	637
1.1.7	Verifique a disponibilidade de uma lista de verificação de codificação segura, requisitos de segurança, diretriz ou política para todos os desenvolvedores e testadores.		✓	✓	637

V1.2 Arquitetura de autenticação

Ao projetar a autenticação, não importa se você tem autenticação multifator habilitada para hardware forte se um invasor puder redefinir uma conta ligando para um call center e respondendo a perguntas comumente conhecidas. Ao provar a identidade, todos os caminhos de autenticação devem ter a mesma força.

#	Descrição	L1	L2	L3	CWE
1.2.1	Verifique o uso de contas exclusivas ou especiais de sistema operacional de baixo privilégio para todos os componentes de aplicações, serviços e servidores. (C3)		✓	✓	250
1.2.2	Verifique se as comunicações entre os componentes da aplicação, incluindo APIs, middleware e camadas de dados, são autenticadas. Os componentes devem ter os privilégios mínimos necessários. (C3)		✓	✓	306
1.2.3	Verifique se a aplicação usa um único mecanismo de autenticação verificado que é conhecido por ser seguro, pode ser estendido para incluir autenticação forte e tem registro e monitoramento suficientes para detectar abuso ou violações de conta.		✓	✓	306
1.2.4	Verifique se todos os caminhos de autenticação e APIs de gestão de identidade implementam força de controle de segurança de autenticação consistente, de modo que não haja alternativas mais fracas pelo risco da aplicação.		✓	✓	306

V1.3 Arquitetura de gestão de sessão

Este é um espaço reservado para futuros requisitos de arquitetura.

V1.4 Arquitetura de controle de acesso

#	Descrição	L1	L2	L3	CWE
1.4.1	Verifique se os pontos de imposição confiáveis, como gateways de controle de acesso, servidores e funções serverless, reforçam os controles de acesso. Nunca imponha controles de acesso no cliente.		✓	✓	602
1.4.2	[EXCLUÍDO, NÃO ACIONÁVEL]				
1.4.3	[EXCLUÍDO, DUPLICADO DE 4.1.3]				

#	Descrição	L1	L2	L3	CWE
1.4.4	Verifique se a aplicação usa um mecanismo de controle de acesso único e bem testado para acessar dados e recursos protegidos. Todas as solicitações devem passar por esse mecanismo único para evitar copiar e colar ou caminhos alternativos inseguros. (C7)		✓	✓	284
1.4.5	Verifique se o controle de acesso baseado em atributo ou recurso é usado pelo qual o código verifica a autorização do usuário para um item de recurso/dado em vez de apenas sua função. As permissões ainda devem ser alocadas usando funções. (C7)		✓	✓	275

V1.5 Arquitetura de input e output

Na versão 4.0, deixamos de usar o termo "lado do servidor" como um termo de limite de confiança carregado. O limite de confiança ainda é preocupante - é possível ignorar a tomada de decisões em navegadores ou dispositivos clientes não confiáveis. No entanto, nas implantações arquitetônicas convencionais de hoje, o ponto de imposição de confiança mudou drasticamente. Portanto, onde o termo "camada de serviço confiável" é usado no ASVS, queremos dizer qualquer ponto de aplicação confiável, independentemente da localização, como um microsserviço, API serverless, lado do servidor, uma API confiável num dispositivo cliente com inicialização segura, parceiro ou APIs externas e assim por diante.

O termo "cliente não confiável" aqui refere-se a tecnologias do lado do cliente que renderizam a camada de apresentação, geralmente chamadas tecnologias de 'front-end'. O termo "serialização" aqui não se refere apenas ao envio de dados pela rede como uma matriz de valores ou à obtenção e leitura de uma estrutura JSON, mas também à passagem de objetos complexos que podem conter lógica.

#	Descrição	L1	L2	L3	CWE
1.5.1	Verifique se os requisitos de input e output definem claramente como lidar e processar dados com base no tipo, conteúdo e leis aplicáveis, regulamentos e outras conformidades com políticas.		✓	✓	1029
1.5.2	Verifique se a serialização não é usada ao se comunicar com clientes não confiáveis. Se isso não for possível, certifique-se de que os controles de integridade adequados (e possivelmente a criptografia se dados confidenciais forem enviados) sejam aplicados para evitar ataques de desserialização, incluindo injeção de objeto.		✓	✓	502
1.5.3	Verifique se a validação de input é aplicada numa camada de serviço confiável. (C5)		✓	✓	602
1.5.4	Verifique se a codificação de output ocorre perto ou pelo interpretador para o qual se destina. (C4)		✓	✓	116

V1.6 Arquitetura Criptográfica

Aplicações precisam ser projetados com arquitetura criptográfica forte para proteger os ativos de dados conforme a sua classificação. Criptografar tudo é um desperdício, não criptografar nada é legalmente negligente. Um equilíbrio deve ser alcançado, geralmente durante o projeto arquitetônico ou de alto nível, sprints de design ou picos arquitetônicos. Projetar a criptografia à medida que avança ou adaptá-la inevitavelmente custará muito mais para implementar com segurança do que simplesmente incorporá-la desde o início.

Os requisitos de arquitetura são intrínsecos a toda a base de código e, assim, difíceis de unidade ou teste integrado. Os requisitos de arquitetura exigem consideração nos padrões de codificação, durante a fase de codificação, e devem ser revisados durante a arquitetura de segurança, revisões de código ou pares, ou retrospectivas.

#	Descrição	L1	L2	L3	CWE
1.6.1	Verifique se há uma política explícita para gestão de chaves criptográficas e se o ciclo de vida de uma chave criptográfica segue um padrão de gestão de chaves, como NIST SP 800-57.		✓	✓	320
1.6.2	Verifique se os consumidores de serviços criptográficos protegem o material de chaves e outros segredos usando cofres de chaves ou alternativas baseadas em API.		✓	✓	320
1.6.3	Verifique se todas as chaves e senhas são substituíveis e fazem parte de um processo bem definido para criptografar novamente dados confidenciais.		✓	✓	320
1.6.4	Verifique se a arquitetura trata os segredos do lado do cliente, como chaves simétricas, senhas ou tokens de API, como inseguros e nunca os usa para proteger ou acessar dados confidenciais.		✓	✓	320

V1.7 Erros, registro e arquitetura de auditoria

#	Descrição	L1	L2	L3	CWE
1.7.1	Verifique se um formato e uma abordagem de log comuns são usados em todo o sistema. (C9)		✓	✓	1009
1.7.2	Verifique se os logs são transmitidos com segurança para um sistema preferencialmente remoto para análise, detecção, alerta e escalonamento. (C9)		✓	✓	

V1.8 Proteção de Dados e Arquitetura de Privacidade

#	Descrição	L1	L2	L3	CWE
1.8.1	Verifique se todos os dados confidenciais são identificados e classificados em níveis de proteção.		✓	✓	
1.8.2	Verifique se todos os níveis de proteção têm um conjunto associado de requisitos de proteção, como requisitos de criptografia, requisitos de integridade, retenção, privacidade e outros requisitos de confidencialidade, e se eles são aplicados na arquitetura.		✓	✓	

V1.9 Arquitetura de Comunicações

#	Descrição	L1	L2	L3	CWE
1.9.1	Verifique se a aplicação criptografa as comunicações entre os componentes, especialmente quando esses componentes estão em contêineres, sistemas, sites ou provedores de nuvem diferentes. (C3)		✓	✓	319
1.9.2	Verifique se os componentes da aplicação verificam a autenticidade de cada lado num link de comunicação para evitar ataques de pessoa no meio. Por exemplo, os componentes da aplicação devem validar cadeias e certificados TLS.		✓	✓	295

V1.10 Arquitetura de software malicioso

#	Descrição	L1	L2	L3	CWE
1.10.1	Verifique se um sistema de controle de código-fonte está em uso, com procedimentos para garantir que os check-ins sejam acompanhados de problemas ou tickets de alteração. O sistema de controle de código-fonte deve ter controle de acesso e usuários identificáveis para permitir a rastreabilidade de quaisquer alterações.		✓	✓	284

V1.11 Arquitetura de lógica de negócios

#	Descrição	L1	L2	L3	CWE
1.11.1	Verifique a definição e a documentação de todos os componentes da aplicação em termos de negócios ou funções de segurança que eles fornecem.		✓	✓	1059
1.11.2	Verifique se todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gestão de sessão e controle de acesso, não compartilham estado não sincronizado.		✓	✓	362
1.11.3	Verifique se todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gestão de sessão e controle de acesso, são thread-safe e resistentes a condições de corrida de tempo de verificação e tempo de uso.			✓	367

V1.12 Arquitetura segura de upload de arquivos

#	Descrição	L1	L2	L3	CWE
1.12.1	[EXCLUÍDO, DUPLICADO DE 12.4.1]				
1.12.2	Verifique se os arquivos carregados pelo usuário - se necessário para serem exibidos ou baixados da aplicação - são servidos por downloads de fluxo de octetos ou de um domínio não relacionado, como um depósito de armazenamento de arquivos em nuvem. Implemente uma Política de Segurança de Conteúdo (CSP) adequada para reduzir o risco de vetores XSS ou outros ataques do arquivo carregado.		✓	✓	646

V1.13 Arquitetura da API

Este é um espaço reservado para futuros requisitos de arquitetura.

V1.14 Arquitetura de configuração

#	Descrição	L1	L2	L3	CWE
1.14.1	Verifique a segregação de componentes de diferentes níveis de confiança por meio de controles de segurança bem definidos, regras de firewall, gateways de API, proxies reversos, grupos de segurança baseados em nuvem ou mecanismos semelhantes.		✓	✓	923
1.14.2	Verifique se as assinaturas binárias, conexões confiáveis e endpoints verificados são usados para implantar binários em dispositivos remotos.		✓	✓	494
1.14.3	Verifique se o pipeline de construção avisa sobre componentes desatualizados ou inseguros e toma as ações apropriadas.		✓	✓	1104

#	Descrição	L1	L2	L3	CWE
1.14.4	Verifique se o pipeline de construção contém uma etapa de construção para construir e verificar automaticamente a implantação segura da aplicação, especialmente se a infraestrutura da aplicação for definida por software, como scripts de construção do ambiente de nuvem.		✓	✓	
1.14.5	Verifique se as implantações de aplicações são adequadamente protegidas, conteinerizadas e/ou isoladas no nível da rede para atrasar e impedir que invasores ataquem outras aplicações, especialmente quando estiverem executando ações confidenciais ou perigosas, como desserialização. (C5)		✓	✓	265
1.14.6	Verifique se a aplicação não usa tecnologias não suportadas, inseguras ou obsoletas do lado do cliente, como plug-ins NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL ou miniaplicações Java do lado do cliente.		✓	✓	477

Referências

Para mais informações, consulte também:

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL
- NIST SP 800-57

V2 Autenticação

Objetivo de controle

Autenticação é o ato de estabelecer ou confirmar que alguém (ou algo) é autêntico e que as afirmações feitas por uma pessoa ou sobre um dispositivo são corretas, resistentes à representação e impedem a recuperação ou interceptação de senhas.

Quando o ASVS foi lançado pela primeira vez, nome de usuário + senha era a forma mais comum de autenticação fora dos sistemas de alta segurança. A autenticação multifator (MFA) era comumente aceita nos círculos de segurança, mas raramente exigida em outros lugares. À medida que o número de violações de senha aumentava, a ideia de que os nomes de usuário são de alguma forma confidenciais e as senhas desconhecidas tornou muitos controles de segurança insustentáveis. Por exemplo, o NIST 800-63 considera nomes de usuário e autenticação baseada em conhecimento (KBA) como informações públicas, SMS e notificações por e-mail como tipos de autenticadores "restritos" e senhas como pré-violadas. Essa realidade torna os autenticadores baseados em conhecimento, a recuperação de SMS e e-mail, o histórico de senhas, a complexidade e os controles de rotação inúteis. Esses controles sempre foram menos úteis,

De todos os capítulos do ASVS, os capítulos de autenticação e gestão de sessão foram os que mais mudaram. A adoção de práticas de liderança eficazes e baseadas em evidências será um desafio para muitos, e isso é perfeitamente aceitável. Temos que começar a transição para um futuro pós-senha agora.

NIST 800-63 - Padrão de autenticação moderno e baseado em evidências

NIST 800-63b é um padrão moderno baseado em evidências e representa o melhor conselho disponível, independentemente da aplicabilidade. O padrão é útil para todas as organizações em todo o mundo, mas é particularmente relevante para as agências dos EUA e aquelas que lidam com agências dos EUA.

A terminologia NIST 800-63 pode ser um pouco confusa no início, especialmente se estiver acostumado apenas com autenticação de nome de usuário + senha. Avanços na autenticação moderna são necessários, por isso temos que introduzir uma terminologia que se tornará comum no futuro, mas entendemos a dificuldade de compreensão até que o setor se estabeleça nesses novos termos. Fornecemos um glossário no final deste capítulo para ajudar. Reformulamos muitos requisitos para satisfazer a intenção do requisito, em vez da letra do requisito. Por exemplo, o ASVS usa o termo "senha" quando o NIST usa "segredo memorizado" neste padrão.

Autenticação ASVS V2, gestão de sessão V3 e, em menor grau, controles de acesso V4 foram adaptados para ser um subconjunto compatível de controles NIST 800-63b selecionados, focados em ameaças comuns e vulnerabilidades de autenticação comumente exploradas. Quando a conformidade total com o NIST 800-63 for necessária, consulte o NIST 800-63.

Selecionando um nível NIST AAL apropriado

O Application Security Verification Standard tentou mapear os requisitos ASVS L1 para NIST AAL1, L2 para AAL2 e L3 para AAL3. No entanto, a abordagem do ASVS Nível 1 como controles "essenciais" pode não ser necessariamente o nível AAL correto para verificar uma aplicação ou API. Por exemplo, se a aplicação for uma aplicação de Nível 3 ou tiver requisitos regulatórios para ser AAL3, o Nível 3 deve ser escolhido nos capítulos V2 e V3 Gestão de Sessões. A escolha do Nível de Asserção de Autenticação (AAL) compatível com NIST deve ser realizada conforme as diretrizes NIST 800-63b, conforme estabelecido em *Selecionar AAL* na Seção 6.2 do NIST 800-63b.

Legenda

As aplicações sempre podem exceder os requisitos do nível atual, especialmente se a autenticação moderna estiver no roteiro de uma aplicação. Anteriormente, o ASVS exigia MFA obrigatório. O NIST não requer MFA obrigatório. Portanto, usamos uma designação opcional neste capítulo para indicar onde o ASVS incentiva, mas não requer um controle. As seguintes chaves são usadas ao longo deste padrão:

Marcar	Descrição
	Não é necessário
o	Recomendado, mas não obrigatório
✓	Obrigatório

V2.1 Segurança de senha

As senhas, chamadas "Memorized Secrets" pelo NIST 800-63, incluem senhas, PINs, padrões de desbloqueio, escolha o gatinho correto ou outro elemento de imagem e frases secretas. Eles são geralmente considerados "algo que você conhece" e frequentemente usados como autenticadores de fator único. Existem desafios significativos para o uso contínuo da autenticação de fator único, incluindo bilhões de nomes de usuários e senhas válidos divulgados na Internet, senhas padrão ou fracas, tabelas arco-íris e dicionários ordenados das senhas mais comuns.

As aplicações devem incentivar fortemente os usuários a se inscreverem na autenticação multifator e devem permitir que os usuários reutilizem tokens que já possuem, como tokens FIDO ou U2F, ou se conectem a um provedor de serviços de credencial que forneça autenticação multifator.

Provedores de serviços de credenciais (CSPs) fornecem identidade federada para usuários. Os usuários geralmente têm mais de uma identidade com vários CSPs, como uma identidade corporativa usando Azure AD, Okta, Ping Identity ou Google, ou identidade de consumidor usando Facebook, Twitter, Google ou WeChat, para citar apenas algumas alternativas comuns. Esta lista não é um endosso dessas empresas ou serviços, mas simplesmente um incentivo para que os desenvolvedores considerem a realidade de que muitos usuários têm muitas identidades estabelecidas. As organizações devem considerar a integração com identidades de usuário existentes, conforme o perfil de risco da força de comprovação de identidade do CSP. Por exemplo, é improvável que uma organização governamental aceite uma identidade de mídia social como um login para sistemas confidenciais, pois é fácil criar identidades falsas ou jogar fora,

#	Descrição	L1	L2	L3	CWE	NIST §
2.1.1	Verifique se as senhas definidas pelo usuário têm pelo menos 12 caracteres (após a combinação de vários espaços). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verifique se senhas com pelo menos 64 caracteres são permitidas e se senhas com mais de 128 caracteres são negadas. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verifique se o truncamento de senha não é executado. No entanto, vários espaços consecutivos podem ser substituídos por um único espaço. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verifique se qualquer caractere Unicode imprimível, incluindo caracteres neutros de idioma, como espaços e Emojis, são permitidos em senhas.	✓	✓	✓	521	5.1.1.2
2.1.5	Verifique se os usuários podem alterar suas senhas.	✓	✓	✓	620	5.1.1.2
2.1.6	Verifique se a funcionalidade de alteração de senha requer a senha atual e nova do usuário.	✓	✓	✓	620	5.1.1.2

#	Descrição	L1	L2	L3	CWE	NIST §
2.1.7	Verifique se as senhas enviadas durante o registro da conta, login e alteração de senha são verificadas em relação a um conjunto de senhas violadas localmente (como as 1.000 ou 10.000 senhas mais comuns que correspondem à política de senha do sistema) ou usando uma API externa. Se estiver usando uma API, uma prova de conhecimento zero ou outro mecanismo deve ser usado para garantir que a senha de texto simples não seja enviada ou usada na verificação do status de violação da senha. Se a senha for violada, a aplicação deve exigir que o usuário defina uma nova senha não violada. (C6)	✓	✓	✓	521	5.1.1.2
2.1.8	Verifique se um medidor de força da senha é fornecido para ajudar os usuários a definir uma senha mais forte.	✓	✓	✓	521	5.1.1.2
2.1.9	Verifique se não há regras de composição de senha limitando o tipo de caracteres permitidos. Não deve haver nenhum requisito para letras maiúsculas ou minúsculas, números ou caracteres especiais. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Verifique se não há rotação periódica de credenciais ou requisitos de histórico de senha.	✓	✓	✓	263	5.1.1.2
2.1.11	Verifique se a funcionalidade "colar", auxiliares de senha do navegador e gerenciadores de senha externos são permitidos.	✓	✓	✓	521	5.1.1.2
2.1.12	Verifique se o usuário pode optar por visualizar temporariamente toda a senha mascarada ou visualizar temporariamente o último caractere digitado da senha em plataformas que não possuem essa funcionalidade integrada.	✓	✓	✓	521	5.1.1.2

Nota: O objetivo de permitir que o usuário visualize a sua senha ou veja o último caractere temporariamente é melhorar a usabilidade do input de credenciais, particularmente em relação ao uso de senhas mais longas, frases secretas e gerenciadores de senhas. Outro motivo para incluir o requisito é impedir ou evitar relatórios de teste que exigem desnecessariamente que as organizações substituam o comportamento do campo de senha da plataforma integrada para remover essa experiência de segurança moderna e amigável.

V2.2 Segurança geral do autenticador

A agilidade do autenticador é essencial para aplicações preparados para o futuro. Refatore os verificadores de aplicações para permitir autenticadores adicionais conforme as preferências do usuário, bem como permitir a retirada de autenticadores obsoletos ou inseguros de maneira ordenada.

O NIST considera e-mail e SMS como tipos de autenticador "restritos" e provavelmente serão removidos do NIST 800-63 e, assim, o ASVS em algum momento no futuro. As candidaturas devem planear um roteiro que não implique a utilização de email ou SMS.

#	Descrição	L1	L2	L3	CWE	NIST §
2.2.1	Verifique se os controles antiautomação são eficazes na mitigação de testes de credenciais violadas, força bruta e ataques de bloqueio de conta. Esses controles incluem o bloqueio das senhas violadas mais comuns, bloqueios suaves, limitação de taxa, CAPTCHA, atrasos cada vez maiores entre tentativas, restrições de endereço IP ou restrições baseadas em risco, como localização, primeiro login num dispositivo, tentativas recentes de desbloquear a conta, ou similar. Verifique se não é possível mais de 100 tentativas com falha por hora numa única conta.	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2

#	Descrição	L1	L2	L3	CWE	NIST §
2.2.2	Verifique se o uso de autenticadores fracos (como SMS e e-mail) é limitado a verificação secundária e aprovação de transações e não como substituto para métodos de autenticação mais seguros. Verifique se métodos mais fortes são oferecidos antes de métodos fracos, se os usuários estão cientes dos riscos ou se as medidas adequadas estão em vigor para limitar os riscos de comprometimento da conta.	✓	✓	✓	304	5.2.10
2.2.3	Verifique se as notificações seguras são enviadas aos usuários após atualizações nos detalhes de autenticação, como redefinições de credenciais, alterações de e-mail ou endereço, login de locais desconhecidos ou arriscados. O uso de notificações push - em vez de SMS ou e-mail - é preferível, mas na ausência de notificações push, SMS ou e-mail é aceitável, desde que nenhuma informação confidencial seja divulgada na notificação.	✓	✓	✓	620	
2.2.4	Verifique a resistência à representação contra phishing, como o uso de autenticação multifator, dispositivos criptográficos com intenção (como chaves conectadas com um push para autenticar) ou em níveis AAL mais altos, certificados do lado do cliente.			✓	308	5.2.5
2.2.5	Verifique se onde um Provedor de Serviços de Credenciais (CSP) e a aplicação que verifica a autenticação estão separados, o TLS mutuamente autenticado está em vigor entre os dois pontos de extremidade.			✓	319	5.2.6
2.2.6	Verifique a resistência à reprodução por meio do uso obrigatório de dispositivos de senhas descartáveis (OTP), autenticadores criptográficos ou códigos de pesquisa.			✓	308	5.2.8
2.2.7	Verifique a intenção de autenticação exigindo a input de um token OTP ou ação iniciada pelo usuário, como pressionar um botão numa chave de hardware FIDO.			✓	308	5.2.9

V2.3 Ciclo de vida do autenticador

Autenticadores são senhas, soft tokens, tokens de hardware e dispositivos biométricos. O ciclo de vida dos autenticadores é crítico para a segurança de uma aplicação - se qualquer pessoa pode registrar uma conta sem evidência de identidade, pode haver pouca confiança na declaração de identidade. Para sites de mídia social como o Reddit, tudo bem. Para sistemas bancários, um maior foco no registro e emissão de credenciais e dispositivos é fundamental para a segurança da aplicação.

Observação: as senhas não devem ter um tempo de vida máximo ou estar sujeitas à rotação de senha. As senhas devem ser verificadas quanto a violação, não substituídas regularmente.

#	Descrição	L1	L2	L3	CWE	NIST §
2.3.1	Verifique se as senhas iniciais ou os códigos de ativação gerados pelo sistema DEVEM ser gerados aleatoriamente de forma segura, DEVEM ter pelo menos 6 caracteres e PODEM conter letras e números e expiram após um curto período de tempo. Esses segredos iniciais não devem ser permitidos para se tornar a senha de longo prazo.	✓	✓	✓	330	5.1.1.2 / A.3

#	Descrição	L1	L2	L3	CWE	NIST §
2.3.2	Verifique se há suporte para inscrição e uso de dispositivos de autenticação fornecidos pelo usuário, como tokens U2F ou FIDO.		✓	✓	308	6.1.3
2.3.3	Verifique se as instruções de renovação são enviadas com tempo suficiente para renovar os autenticadores com limite de tempo.		✓	✓	287	6.1.4

V2.4 Armazenamento de credenciais

Arquitetos e desenvolvedores devem aderir a esta seção ao criar ou refatorar o código. Esta seção só pode ser totalmente verificada usando a revisão do código-fonte ou por meio de testes seguros de unidade, ou integração. O teste de penetração não consegue identificar nenhum desses problemas.

A lista de funções de derivação de chave unidirecional aprovadas é detalhada na seção 5.1.1.2 do NIST 800-63 B e em BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018). O algoritmo nacional ou regional mais recente e os padrões de comprimento de chave podem ser escolhidos no lugar dessas opções.

Esta seção não pode ser testada quanto à penetração, então os controles não são marcados como L1. No entanto, esta seção é de vital importância para a segurança das credenciais se forem roubadas, portanto, se for bifurcar o ASVS para uma arquitetura ou diretriz de codificação, ou lista de verificação de revisão de código-fonte, coloque esses controles de volta em L1 na sua versão privada.

#	Descrição	L1	L2	L3	CWE	NIST §
2.4.1	Verifique se as senhas são armazenadas de forma resistente a ataques off-line. As senhas DEVEM ser salted e hash usando uma derivação de chave unidirecional aprovada ou função de hashing de senha. As funções de derivação de chave e hash de senha usam uma senha, um sal e um fator de custo como inputs ao gerar um hash de senha. (C6)		✓	✓	916	5.1.1.2
2.4.2	Verifique se o sal tem pelo menos 32 bits de comprimento e é escolhido arbitrariamente para minimizar as colisões de valor de sal entre os hashes armazenados. Para cada credencial, um valor salt único e o hash resultante DEVEM ser armazenados. (C6)		✓	✓	916	5.1.1.2
2.4.3	Verifique se PBKDF2 é usado, a contagem de iteração DEVE ser tão grande quanto o desempenho do servidor de verificação permitir, normalmente pelo menos 100.000 iterações. (C6)		✓	✓	916	5.1.1.2
2.4.4	Verifique se o bcrypt é usado, o fator de trabalho DEVE ser tão grande quanto o desempenho do servidor de verificação permitir, com um mínimo de 10. (C6)		✓	✓	916	5.1.1.2
2.4.5	Verifique se uma iteração adicional de uma função de derivação de chave é executada, usando um valor salt que é secreto e conhecido apenas pelo verificador. Gere o valor salt usando um gerador de bit aleatório aprovado [SP 800-90Ar1] e forneça pelo menos a segurança mínima especificada na última revisão do SP 800-131A. O valor salt secreto DEVERÁ ser armazenado separadamente das senhas com hash (por exemplo, num dispositivo especializado como um módulo de segurança de hardware).		✓	✓	916	5.1.1.2

Quando os padrões dos EUA são mencionados, um padrão regional ou local pode ser usado no lugar, ou em adição ao padrão dos EUA, conforme necessário.

V2.5 Recuperação de credenciais

#	Descrição	L1	L2	L3	CWE	NIST §
2.5.1	Verifique se um segredo inicial de ativação ou recuperação gerado pelo sistema não é enviado em texto não criptografado ao usuário. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Verifique se dicas de senha ou autenticação baseada em conhecimento (as chamadas "perguntas secretas") não estão presentes.	✓	✓	✓	640	5.1.1.2
2.5.3	Verifique se a recuperação da credencial de senha não revela a senha atual de forma alguma. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Verifique se contas compartilhadas ou padrão não estão presentes (por exemplo, "root", "admin" ou "sa").	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Verifique se um fator de autenticação é alterado ou substituído, o usuário é notificado sobre esse evento.	✓	✓	✓	304	6.1.2.3
2.5.6	Verifique a senha esquecida e outros caminhos de recuperação usam um mecanismo de recuperação seguro, como OTP baseado em tempo (TOTP) ou outro token flexível, push móvel ou outro mecanismo de recuperação offline. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Verifique se os fatores de autenticação OTP ou multifator são perdidos, se a prova de identidade é realizada no mesmo nível que durante o registro.		✓	✓	308	6.1.2.3

V2.6 Verificador secreto de pesquisa

Os segredos de pesquisa são listas pré-geradas de códigos secretos, como números de autorização de transação (TAN), códigos de recuperação de mídia social ou uma grade contendo um conjunto de valores aleatórios. Estes são distribuídos de forma segura para os usuários. Esses códigos de pesquisa são usados uma vez e, uma vez usados, a lista secreta de pesquisa é descartada. Esse tipo de autenticador é considerado "algo que você tem".

#	Descrição	L1	L2	L3	CWE	NIST §
2.6.1	Verifique se os segredos de pesquisa podem ser usados apenas uma vez.		✓	✓	308	5.1.2.2
2.6.2	Verifique se os segredos de pesquisa têm aleatoriedade suficiente (112 bits de entropia) ou, se tiverem menos de 112 bits de entropia, salteados com um sal exclusivo e aleatório de 32 bits e hash com um hash unidirecional aprovado.		✓	✓	330	5.1.2.2
2.6.3	Verifique se os segredos de pesquisa são resistentes a ataques offline, como valores previsíveis.		✓	✓	310	5.1.2.2

V2.7 Verificador fora de banda

No passado, um verificador comum fora de banda seria um e-mail ou SMS contendo um link de redefinição de senha. Os invasores usam esse mecanismo fraco para redefinir contas que ainda não controlam, como assumir o controle da conta de e-mail de uma pessoa e reutilizar qualquer link de redefinição descoberto. Existem maneiras melhores de lidar com a verificação fora da banda.

Autenticadores seguros fora da banda são dispositivos físicos que podem se comunicar com o verificador por um canal secundário seguro. Os exemplos incluem notificações push para dispositivos móveis. Esse tipo de

autenticador é considerado "algo que você tem". Quando um usuário deseja autenticar, a aplicação de verificação envia uma mensagem para o autenticador fora de banda por meio de uma conexão com o autenticador direta ou indiretamente por um serviço terceirizado. A mensagem contém um código de autenticação (normalmente um número aleatório de seis dígitos ou uma caixa de diálogo de aprovação modal). A aplicação verificador espera receber o código de autenticação por meio do canal primário e compara o hash do valor recebido com o hash do código de autenticação original. Se forem iguais, o verificador fora da banda pode presumir que o usuário foi autenticado.

O ASVS assume que apenas alguns desenvolvedores desenvolverão novos autenticadores fora de banda, como notificações push e, assim, os seguintes controles ASVS se aplicam a verificadores, como API de autenticação, aplicações e implementações de logon único. Se estiver desenvolvendo um novo autenticador fora de banda, consulte NIST 800-63B § 5.1.3.1.

Autenticadores inseguros fora de banda, como e-mail e VOIP, não são permitidos. A autenticação PSTN e SMS são atualmente "restritas" pelo NIST e devem ser substituídas em favor de notificações por push ou similares. Se você precisar usar autenticação fora de banda por telefone ou SMS, consulte o § 5.1.3.3.

#	Descrição	L1	L2	L3	CWE	NIST §
2.7.1	Verifique se os autenticadores de texto não criptografado fora da banda (NIST "restrito"), como SMS ou PSTN, não são oferecidos por padrão e alternativas mais fortes, como notificações por push, são oferecidas primeiro.	✓	✓	✓	287	5.1.3.2
2.7.2	Verifique se o verificador fora de banda expira solicitações, códigos ou tokens de autenticação fora de banda após 10 minutos.	✓	✓	✓	287	5.1.3.2
2.7.3	Verifique se as solicitações, códigos ou tokens de autenticação do verificador fora de banda podem ser usados apenas uma vez e apenas para a solicitação de autenticação original.	✓	✓	✓	287	5.1.3.2
2.7.4	Verifique se o autenticador e o verificador fora da banda se comunicam por um canal independente seguro.	✓	✓	✓	523	5.1.3.2
2.7.5	Verifique se o verificador fora da banda retém apenas uma versão com hash do código de autenticação.		✓	✓	256	5.1.3.2
2.7.6	Verifique se o código de autenticação inicial é gerado por um gerador de número aleatório seguro, contendo pelo menos 20 bits de entropia (normalmente, um número aleatório de seis dígitos é suficiente).		✓	✓	310	5.1.3.2

V2.8 Verificador Único

Senhas únicas de fator único (OTPs) são tokens físicos ou flexíveis que exibem um desafio único pseudoaleatório em constante mudança. Esses dispositivos tornam o phishing (personificação) difícil, mas não impossível. Esse tipo de autenticador é considerado "algo que você tem". Os tokens multifatores são semelhantes aos OTPs de fator único, mas exigem a inserção de um código PIN válido, desbloqueio biométrico, inserção USB ou emparelhamento NFC, ou algum valor adicional (como calculadoras de assinatura de transação) para criar o OTP final.

#	Descrição	L1	L2	L3	CWE	NIST §
2.8.1	Verifique se os OTPs baseados em tempo têm um tempo de vida definido antes de expirar.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2

#	Descrição	L1	L2	L3	CWE	NIST §
2.8.2	Verifique se as chaves simétricas usadas para verificar os OTPs enviados são altamente protegidas, por exemplo, usando um módulo de segurança de hardware ou armazenamento de chave baseado em sistema operacional seguro.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Verifique se algoritmos criptográficos aprovados são usados na geração, propagação e verificação de OTPs.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Verifique se o OTP baseado em tempo pode ser usado apenas uma vez no período de validade.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Verifique se um token OTP multifator baseado em tempo é reutilizado durante o período de validade, ele é registrado e rejeitado com notificações seguras sendo enviadas ao proprietário do dispositivo.		✓	✓	287	5.1.5.2
2.8.6	Verifique se o gerador OTP físico de fator único pode ser revogado em caso de roubo ou outra perda. Assegure-se de que a revogação entre em vigor imediatamente nas sessões de login, independentemente do local.		✓	✓	613	5.2.1
2.8.7	Verifique se os autenticadores biométricos estão limitados ao uso apenas como fatores secundários em conjunto com algo que você possui e algo que você conhece.		o	✓	308	5.2.3

V2.9 Verificador Criptográfico

As chaves de segurança criptográficas são cartões inteligentes ou chaves FIDO, nas quais o usuário deve conectar ou emparelhar o dispositivo criptográfico ao computador para concluir a autenticação. Os verificadores enviam um nonce de desafio para os dispositivos criptográficos ou software, e o dispositivo ou software calcula uma resposta com base numa chave criptográfica armazenada com segurança.

Os requisitos para dispositivos e software criptográficos de fator único e dispositivos e software criptográficos multifatores são os mesmos, pois a verificação do autenticador criptográfico prova a posse do fator de autenticação.

#	Descrição	L1	L2	L3	CWE	NIST §
2.9.1	Verifique se as chaves criptográficas usadas na verificação são armazenadas com segurança e protegidas contra divulgação, como o uso de um Módulo de plataforma confiável (TPM) ou Módulo de segurança de hardware (HSM) ou um serviço de sistema operacional que pode usar esse armazenamento seguro.		✓	✓	320	5.1.7.2
2.9.2	Verifique se o nonce de desafio tem pelo menos 64 bits de comprimento e é estatisticamente exclusivo ou exclusivo durante a vida útil do dispositivo criptográfico.		✓	✓	330	5.1.7.2
2.9.3	Verifique se os algoritmos criptográficos aprovados são usados na geração, propagação e verificação.		✓	✓	327	5.1.7.2

V2.10 Autenticação de Serviço

Esta seção não é passível de teste de penetração, portanto, não possui nenhum requisito L1. No entanto, se usado numa arquitetura, codificação ou revisão de código seguro, assuma que o software (assim como Java Key Store) é o requisito mínimo em L1. O armazenamento de segredos em texto não criptografado não é aceitável em nenhuma circunstância.

#	Descrição	L1	L2	L3	CWE	NIST §
2.10.1	Verifique se os segredos intra-serviço não dependem de credenciais imutáveis, como senhas, chaves de API ou contas compartilhadas com acesso privilegiado.		SO assistido	HSM	287	5.1.1.1
2.10.2	Verifique se as senhas são necessárias para autenticação de serviço, a conta de serviço usada não é uma credencial padrão. (por exemplo, root/root ou admin/admin são padrão em alguns serviços durante a instalação).		SO assistido	HSM	255	5.1.1.1
2.10.3	Verifique se as senhas são armazenadas com proteção suficiente para evitar ataques de recuperação offline, incluindo acesso ao sistema local.		SO assistido	HSM	522	5.1.1.1
2.10.4	Verifique se senhas, integrações com bancos de dados e sistemas de terceiros, sementes e segredos internos e chaves de API são gerenciados com segurança e não incluídos no código-fonte ou armazenados em repositórios de código-fonte. Esse armazenamento DEVE resistir a ataques offline. O uso de um armazenamento de chave de software seguro (L1), TPM de hardware ou um HSM (L3) é recomendado para armazenamento de senha.		SO assistido	HSM	798	

Requisitos adicionais da agência dos EUA

As agências dos EUA têm requisitos obrigatórios relativos ao NIST 800-63. O Application Security Verification Standard sempre foi sobre os 80% dos controles que se aplicam a quase 100% das aplicações, e não os últimos 20% dos controles avançados ou aqueles que têm aplicabilidade limitada. Como tal, o ASVS é um subconjunto estrito do NIST 800-63, especialmente para as classificações IAL1/2 e AAL1/2, mas não é suficientemente abrangente, particularmente no que diz respeito às classificações IAL3/AAL3.

Nós instamos fortemente as agências governamentais dos EUA a revisar e implementar o NIST 800-63 na sua totalidade.

Glossário de termos

Termo	Significado
PSC	Provedor de Serviços de Credencial também chamado Provedor de Identidade
Autenticador	Código que autentica uma senha, token, MFA, declaração federada e assim por diante.
Verificador	"Uma entidade que verifica a identidade do reclamante verificando a posse e o controle do reclamante de um ou dois autenticadores usando um protocolo de autenticação. Para fazer isso, o verificador também pode precisar validar as credenciais que vinculam o(s) autenticador(es) ao identificador do assinante e verificar o seu estado"
OTP	Senha de uso único
SFA	Autenticadores de fator único, como algo que conheça (segredos memorizados, senhas, senhas, PINs), algo que você é (biometria, impressão digital, digitalizações faciais) ou algo que possui (tokens OTP, um dispositivo criptográfico como um cartão inteligente) ,
AMF	Autenticação multifator, que inclui dois ou mais fatores únicos

Referências

Para mais informações, consulte também:

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management
- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3 Gestão de sessão

Objetivo de controle

Um dos principais componentes de qualquer aplicação baseada na Web ou API com estado é o mecanismo pelo qual ele controla e mantém o estado de um usuário ou dispositivo que interage com ele. A gestão de sessão altera um protocolo sem estado para com estado, o que é crítico para diferenciar diferentes usuários ou dispositivos.

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de gestão de sessão de alto nível:

- As sessões são únicas para cada indivíduo e não podem ser adivinhadas ou compartilhadas.
- As sessões são invalidadas quando não são mais necessárias e expiram durante períodos de inatividade.

Conforme observado anteriormente, esses requisitos foram adaptados para serem um subconjunto compatível de controles NIST 800-63b selecionados, focados em ameaças comuns e vulnerabilidades de autenticação comumente exploradas. Os requisitos de verificação anteriores foram retirados, desduplicados ou, geralmente, adaptados para serem fortemente alinhados com a intenção dos NIST 800-63b requisitos.

Requisitos de verificação de segurança

V3.1 Fundamentos de Segurança de Gestão de Sessão

#	Descrição	L1	L2	L3	CWE	NIST §
3.1.1	Verifique se a aplicação nunca revela tokens de sessão em parâmetros de URL.	✓	✓	✓	598	

V3.2 Ligação de Sessão

#	Descrição	L1	L2	L3	CWE	NIST §
3.2.1	Verifique se a aplicação gera um novo token de sessão na autenticação do usuário. (C6)	✓	✓	✓	384	7.1
3.2.2	Verifique se os tokens de sessão possuem pelo menos 64 bits de entropia. (C6)	✓	✓	✓	331	7.1
3.2.3	Verifique se a aplicação armazena apenas tokens de sessão no navegador usando métodos seguros, como cookies devidamente protegidos (consulte a seção 3.4) ou armazenamento de sessão HTML 5.	✓	✓	✓	539	7.1
3.2.4	Verifique se os tokens de sessão são gerados usando algoritmos criptográficos aprovados. (C6)		✓	✓	331	7.1

O TLS ou outro canal de transporte seguro é obrigatório para a gestão de sessão. Isso é abordado no capítulo Segurança das Comunicações.

V3.3 Encerramento da Sessão

Os session timeouts foram alinhados com o NIST 800-63, que permite session timeouts muito mais longos do que os tradicionalmente permitidos pelos padrões de segurança. As organizações devem revisar a tabela abaixo e, se um tempo limite mais longo for desejável com base no risco da aplicação, o valor NIST deve ser o limite superior dos tempos limite de inatividade da sessão.

L1 neste contexto é IAL1/AAL1, L2 é IAL2/AAL3, L3 é IAL3/AAL3. Para IAL2/AAL2 e IAL3/AAL3, o tempo limite de inatividade mais curto é o limite inferior de tempos de inatividade para ser desconectado ou autenticado novamente para retomar a sessão.

#	Descrição	L1	L2	L3	CWE	NIST §
3.3.1	Verifique se o logoff e a expiração invalidam o token de sessão, de modo que o botão Voltar ou uma parte confiável downstream não retome uma sessão autenticada, inclusive entre partes confiáveis. (C6)	✓	✓	✓	613	7.1
3.3.2	Se os autenticadores permitirem que os usuários permaneçam conectados, verifique se a reautenticação ocorre periodicamente quando usados ativamente ou após um período ocioso. (C6)	30 dias	12 horas ou 30 minutos de inatividade, 2FA opcional	12 horas ou 15 minutos de inatividade, com 2FA	613	7.2
3.3.3	Verifique se a aplicação oferece a opção de encerrar todas as outras sessões ativas após uma alteração de senha bem-sucedida (incluindo alteração por redefinição/recuperação de senha) e se isso é eficaz na aplicação, no login federado (se houver) e em quaisquer partes confiáveis.		✓	✓	613	
3.3.4	Verifique se os usuários podem visualizar e (tendo inserido novamente as credenciais de login) fazer logoff de qualquer ou de todas as sessões e dispositivos ativos no momento.		✓	✓	613	7.1

V3.4 Gestão de sessão baseado em cookies

#	Descrição	L1	L2	L3	CWE	NIST §
3.4.1	Verifique se os tokens de sessão baseados em cookie têm o atributo 'Seguro' definido. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Verifique se os tokens de sessão baseados em cookie têm o atributo 'HttpOnly' definido. (C6)	✓	✓	✓	1004	7.1.1
3.4.3	Verifique se os tokens de sessão baseados em cookies utilizam o atributo 'SameSite' para limitar a exposição a ataques de falsificação de solicitação entre sites. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Verifique se os tokens de sessão baseados em cookie usam o prefixo "__Host-" para que os cookies sejam enviados apenas para o host que inicialmente definiu o cookie.	✓	✓	✓	16	7.1.1
3.4.5	Verifique se a aplicação é publicado sob um nome de domínio com outras aplicações que definem ou usam cookies de sessão que podem divulgar os cookies de sessão, defina o atributo path em tokens de sessão baseados em cookie usando o caminho mais preciso possível. (C6)	✓	✓	✓	16	7.1.1

V3.5 Gestão de sessão baseado em token

A gestão de sessão baseado em token inclui chaves JWT, OAuth, SAML e API. Destas, as chaves de API são conhecidas por serem fracas e não devem ser usadas em novos códigos.

#	Descrição	L1	L2	L3	CWE	NIST §
3.5.1	Verifique se a aplicação permite que os usuários revoguem tokens OAuth que formam relacionamentos de confiança coma aplicaçãos vinculados.		✓	✓	290	7.1.2
3.5.2	Verifique se a aplicação usa tokens de sessão em vez de chaves e segredos de API estáticos, exceto com implementações herdadas.		✓	✓	798	
3.5.3	Verifique se os tokens de sessão sem estado usam assinaturas digitais, criptografia e outras contramedidas para proteger contra adulteração, envelopamento, repetição, cifra nula e ataques de substituição de chave.		✓	✓	345	

V3.6 Reautenticação federada

Esta seção se refere àqueles que escrevem códigos de Parte Confiável (RP) ou Provedor de Serviços de Credenciais (CSP). Se depender do código que implementa esses recursos, certifique-se de que esses problemas sejam tratados corretamente.

#	Descrição	L1	L2	L3	CWE	NIST §
3.6.1	Verifique se as Partes Confiáveis (RPs) especificam o tempo máximo de autenticação para Provedores de Serviços de Credenciais (CSPs) e se os CSPs autenticam novamente o usuário se eles não tiverem usado uma sessão dentro desse período.			✓	613	7.2.1
3.6.2	Verifique se os provedores de serviços de credenciais (CSPs) informam as partes confiáveis (RPs) sobre o último evento de autenticação, para permitir que os RPs determinem se precisam autenticar novamente o usuário.			✓	613	7.2.1

V3.7 Defesas contra exploits de gestão de sessão

Há um pequeno número de ataques de gestão de sessão, alguns relacionados à experiência do usuário (UX) das sessões. Anteriormente, com base nos requisitos da ISO 27002, o ASVS exigia o bloqueio de várias sessões simultâneas. Bloquear sessões simultâneas não é mais apropriado, não apenas porque os usuários modernos têm muitos dispositivos ou a aplicação é uma API sem uma sessão do navegador, mas na maioria dessas implementações, o último autenticador vence, que geralmente é o invasor. Esta seção fornece orientações importantes sobre dissuasão, atraso e detecção de ataques de gestão de sessão usando código.

Descrição do ataque semi-aberto

No início de 2018, várias instituições financeiras foram comprometidas usando o que os invasores chamaram "ataques semiabertos". Este termo ficou preso na indústria. Os invasores atingiram várias instituições com diferentes bases de código proprietárias e, de fato, parecem diferentes bases de código dentro das mesmas instituições. O ataque semiaberto está explorando uma falha de padrão de design comumente encontrada em muitos sistemas existentes de autenticação, gestão de sessão e controle de acesso.

Os invasores iniciam um ataque semiaberto tentando bloquear, redefinir ou recuperar uma credencial. Um padrão de design de gestão de sessão popular reutiliza objetos/modelos de sessão de perfil de usuário entre código não autenticado, parcialmente autenticado (redefinições de senha, nome de usuário esquecido) e totalmente autenticado. Esse padrão de design preenche um objeto ou token de sessão válido contendo o

perfil da vítima, incluindo hashes de senha e funções. Se as verificações de controle de acesso em controladores ou roteadores não verificarem corretamente se o usuário está totalmente conectado, o invasor poderá agir como o usuário. Os ataques podem incluir alterar a senha do usuário para um valor conhecido, atualizar o endereço de e-mail para executar uma redefinição de senha válida, desabilitar a autenticação multifator ou registrar um novo dispositivo MFA, revelar ou alterar chaves de API e assim por diante.

#	Descrição	L1	L2	L3	CWE	NIST §
3.7.1	Verifique se a aplicação garante uma sessão de login completa e válida ou requer reautenticação ou verificação secundária antes de permitir transações confidenciais ou modificações de conta.	✓	✓	✓	306	

Referências

Para mais informações, consulte também:

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie Host- prefix details

V4 Controle de Acesso

Objetivo de controle

Autorização é o conceito de permitir acesso a recursos apenas para aqueles autorizados a usá-los. Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- As pessoas que acessam recursos possuem credenciais válidas para fazê-lo.
- Os usuários estão associados a um conjunto bem definido de funções e privilégios.
- Os metadados de função e permissão são protegidos contra repetição ou adulteração.

Requisitos de verificação de segurança

V4.1 Projeto de controle de acesso geral

#	Descrição	L1	L2	L3	CWE
4.1.1	Verifique se a aplicação impõe regras de controle de acesso numa camada de serviço confiável, especialmente se o controle de acesso do lado do cliente estiver presente e puder ser ignorado.	✓	✓	✓	602
4.1.2	Verifique se todos os atributos de usuário e dados e as informações de política usadas pelos controles de acesso não podem ser manipulados pelos usuários finais, a menos que especificamente autorizados.	✓	✓	✓	639
4.1.3	Verifique se existe o princípio do menor privilégio - os usuários só devem poder acessar funções, arquivos de dados, URLs, controladores, serviços e outros recursos, para os quais possuam autorização específica. Isso implica proteção contra falsificação e elevação de privilégio. (C7)	✓	✓	✓	285
4.1.4	[EXCLUÍDO, DUPLICADO DE 4.1.3]				
4.1.5	Verifique se os controles de acesso falham com segurança, inclusive quando ocorre uma exceção. (C10)	✓	✓	✓	285

V4.2 Controle de acesso de nível de operação

#	Descrição	L1	L2	L3	CWE
4.2.1	Verifique se os dados confidenciais e as APIs estão protegidos contra ataques Insecure Direct Object Reference (IDOR) direcionados à criação, leitura, atualização e exclusão de registros, como criar ou atualizar o registro de outra pessoa, visualizar os registros de todos ou excluir todos os registros.	✓	✓	✓	639
4.2.2	Verifique se a aplicação ou estrutura impõe um forte mecanismo anti-CSRF para proteger a funcionalidade autenticada e se a antiautomação ou anti-CSRF eficaz protege a funcionalidade não autenticada.	✓	✓	✓	352

V4.3 Outras considerações de controle de acesso

#	Descrição	L1	L2	L3	CWE
4.3.1	Verifique se as interfaces administrativas usam autenticação multifator apropriada para impedir o uso não autorizado.	✓	✓	✓	419

#	Descrição	L1	L2	L3	CWE
4.3.2	Verifique se a navegação no diretório está desativada, a menos que seja deliberadamente desejada. Além disso, as aplicações não devem permitir a descoberta ou divulgação de metadados de arquivo ou diretório, como pastas Thumbs.db, .DS_Store, .git ou .svn.	✓	✓	✓	548
4.3.3	Verifique se a aplicação possui autorização adicional (como intensificação ou autenticação adaptativa) para sistemas de baixo valor e/ou segregação de funções para aplicações de alto valor para impor controles antifraude de acordo com o risco da aplicação e fraudes anteriores.		✓	✓	732

Referências

Para mais informações, consulte também:

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet

V5 Validação, Sanitização e Codificação

Objetivo de controle

A falha de segurança da aplicação da Web mais comum é a falha em validar adequadamente a input proveniente do cliente ou do ambiente antes de usá-la diretamente sem qualquer codificação de output. Essa fraqueza leva a quase todas as vulnerabilidades significativas em aplicações da Web, como Cross-Site Scripting (XSS), injeção de SQL, interpreter injection, ataques de localidade/Unicode, ataques de sistema de arquivos e buffer overflows.

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- Validação de input e arquitetura de codificação de output têm um pipeline acordado para evitar ataques de injeção.
- Os dados de input são fortemente digitados, validados, com intervalo ou comprimento verificados, ou, na pior das hipóteses, higienizados ou filtrados.
- Os dados de output são codificados ou escapados conforme o contexto dos dados o mais próximo possível do interpretador.

Com a arquitetura moderna de aplicações da Web, a codificação de output é mais importante do que nunca. É difícil fornecer validação de input robusta em determinados cenários, portanto, o uso de API mais segura, como consultas parametrizadas, estruturas de modelo de escape automático ou codificação de output cuidadosamente escolhida, é fundamental para a segurança da aplicação.

V5.1 Validação de input

Controles de validação de input implementados adequadamente, usando listas de permissões positivas e digitação forte de dados, podem eliminar mais de 90% de todos os ataques de injeção. As verificações de comprimento e alcance podem reduzir isso ainda mais. Construir a validação de input segura é necessário durante a arquitetura da aplicação, sprints de design, codificação e testes de unidade e integração. Embora muitos desses itens não possam ser encontrados em testes de penetração, os resultados de não implementá-los são geralmente encontrados em V5.3 - Codificação de output e requisitos de prevenção de injeção. Desenvolvedores e revisores de código seguro são recomendados para tratar esta seção como se L1 fosse necessário para todos os itens para evitar injeções.

#	Descrição	L1	L2	L3	CWE
5.1.1	Verifique se a aplicação possui defesas contra ataques de poluição de parâmetro HTTP, principalmente se a estrutura da aplicação não fizer distinção sobre a origem dos parâmetros de solicitação (GET, POST, cookies, cabeçalhos ou variáveis de ambiente).	✓	✓	✓	235
5.1.2	Verifique se as estruturas protegem contra ataques de atribuição de parâmetros em massa ou se a aplicação possui contramedidas para proteger contra atribuição de parâmetros não segura, como marcar campos como privados ou similares. (C5)	✓	✓	✓	915
5.1.3	Verifique se todas as inputs (campos de formulário HTML, solicitações REST, parâmetros de URL, cabeçalhos HTTP, cookies, arquivos em lote, feeds RSS, etc.) são validadas usando validação positiva (listas de permissão). (C5)	✓	✓	✓	20
5.1.4	Verifique se os dados estruturados são fortemente digitados e validados em relação a um esquema definido, incluindo caracteres permitidos, comprimento e padrão (por exemplo, números de cartão de crédito, endereços de e-mail, números de telefone ou validação de que dois campos relacionados são razoáveis, como verificar o subúrbio e o CEP /correspondência de código postal). (C5)	✓	✓	✓	20

#	Descrição	L1	L2	L3	CWE
5.1.5	Verifique se os redirecionamentos e encaminhamentos de URL permitem apenas destinos que aparecem numa lista de permissões ou exibem um aviso ao redirecionar para conteúdo potencialmente não confiável.	✓	✓	✓	601

V5.2 Sanitização e Sandbox

#	Descrição	L1	L2	L3	CWE
5.2.1	Verifique se todos os inputs de HTML não confiáveis de editores WYSIWYG ou similares foram devidamente sanitizadas com uma biblioteca, ou recurso de estrutura do higienizador de HTML. (C5)	✓	✓	✓	116
5.2.2	Verifique se os dados não estruturados são limpos para impor medidas de segurança, como caracteres e comprimento permitidos.	✓	✓	✓	138
5.2.3	Verifique se a aplicação limpa a input do usuário antes de passar para os sistemas de e-mail para proteger contra injeção de SMTP ou IMAP.	✓	✓	✓	147
5.2.4	Verifique se a aplicação evita o uso de eval() ou outros recursos dinâmicos de execução de código. Onde não houver alternativa, qualquer input do usuário incluída deve ser limpa ou protegida antes de ser executada.	✓	✓	✓	95
5.2.5	Verifique se a aplicação protege contra-ataques de injeção de modelo, garantindo que qualquer input do usuário incluída seja limpa ou protegida.	✓	✓	✓	94
5.2.6	Verifique se a aplicação protege contra-ataques SSRF, validando ou limpando dados não confiáveis, ou metadados de arquivos HTTP, como nomes de arquivos e campos de input de URL, e usa listas de permissão de protocolos, domínios, caminhos e portas.	✓	✓	✓	918
5.2.7	Verifique se a aplicação sanitiza, desativa ou coloca em área restrita o conteúdo de scripts Scalable Vector Graphics (SVG) fornecido pelo usuário, especialmente no que se refere a XSS resultante de scripts embutidos e ao ForeignObject.	✓	✓	✓	159
5.2.8	Verifique se a aplicação sanitiza, desativa ou coloca em sandbox o conteúdo de linguagem de modelo de expressão ou script fornecido pelo usuário, como folhas de estilo Markdown, CSS ou XSL, BBCode ou similares.	✓	✓	✓	94

V5.3 Codificação de Output e Prevenção de Injeção

A codificação de output próxima ou adjacente ao interpretador em uso é crítica para a segurança de qualquer aplicação. Normalmente, a codificação de output não é mantida, mas usada para tornar a output segura no contexto de output apropriado para uso imediato. Deixar de codificar a output resultará numa aplicação injetável e insegura.

#	Descrição	L1	L2	L3	CWE
5.3.1	Verifique se a codificação de output é relevante para o interpretador e o contexto necessários. Por exemplo, use codificadores especificamente para valores HTML, atributos HTML, JavaScript, parâmetros de URL, cabeçalhos HTTP, SMTP e outros conforme o contexto exigir, especialmente de inputs não confiáveis (por exemplo, nomes com Unicode ou apóstrofes, como ねこ ou O'Hara) . (C4)	✓	✓	✓	116

#	Descrição	L1	L2	L3	CWE
5.3.2	Verifique se a codificação de output preserva o conjunto de caracteres e localidade escolhidos pelo usuário, de forma que qualquer ponto de caractere Unicode seja válido e manipulado com segurança. (C4)	✓	✓	✓	176
5.3.3	Verifique se o escape de output sensível ao contexto, de preferência automatizado - ou, na pior das hipóteses, manual - protege contra XSS refletido, armazenado e baseado em DOM. (C4)	✓	✓	✓	79
5.3.4	Verifique se a seleção de dados ou consultas de banco de dados (por exemplo, SQL, HQL, ORM, NoSQL) usam consultas parametrizadas, ORMs, estruturas de entidade ou estão protegidas contra ataques de injeção de banco de dados. (C3)	✓	✓	✓	89
5.3.5	Verifique se onde mecanismos parametrizados ou mais seguros não estão presentes, a codificação de output específica do contexto é usada para proteger contra ataques de injeção, como o uso de escape SQL para proteger contra injeção SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Verifique se a aplicação protege contra ataques de injeção JSON, ataques de avaliação JSON e avaliação de expressão JavaScript. (C4)	✓	✓	✓	830
5.3.7	Verifique se a aplicação protege contra vulnerabilidades de injeção de LDAP ou se foram implementados controles de segurança específicos para impedir a injeção de LDAP. (C4)	✓	✓	✓	90
5.3.8	Verifique se a aplicação protege contra injeção de comando do SO e se as chamadas do sistema operacional usam consultas de SO parametrizadas ou usam codificação de output de linha de comando contextual. (C4)	✓	✓	✓	78
5.3.9	Verifique se a aplicação protege contra ataques de inclusão de arquivo local (LFI) ou inclusão de arquivo remoto (RFI).	✓	✓	✓	829
5.3.10	Verifique se a aplicação protege contra ataques de injeção XPath ou injeção XML. (C4)	✓	✓	✓	643

Observação: usar consultas parametrizadas ou escapar do SQL nem sempre é suficiente; nomes de tabelas e colunas, ORDER BY e assim por diante, não podem ser ignorados. A inclusão de dados fornecidos pelo usuário com escape nesses campos resulta em consultas com falha ou injeção de SQL.

Observação: o formato SVG permite explicitamente o script ECMA em quase todos os contextos, portanto, pode não ser possível bloquear completamente todos os vetores SVG XSS. Se o upload de SVG for necessário, é altamente recomendável servir esses arquivos carregados como texto/sem formatação ou usar um domínio de conteúdo fornecido pelo usuário separado para evitar que o XSS bem-sucedido assuma o controle da aplicação.

V5.4 Memória, String e Código Não Gerenciado

Os requisitos a seguir serão aplicados apenas quando a aplicação usar uma linguagem de sistema ou código não gerenciado.

#	Descrição	L1	L2	L3	CWE
5.4.1	Verifique se a aplicação usa cadeia de memória segura, cópia de memória mais segura e aritmética de ponteiro para detectar ou evitar estouros de pilha, buffer ou heap.		✓	✓	120
5.4.2	Verifique se as strings de formato não aceitam inputs potencialmente hostis e são constantes.		✓	✓	134

#	Descrição	L1	L2	L3	CWE
5.4.3	Verifique se as técnicas de validação de sinal, intervalo e input são usadas para evitar estouros de número inteiro.		✓	✓	190

V5.5 Prevenção de desserialização

#	Descrição	L1	L2	L3	CWE
5.5.1	Verifique se os objetos serializados usam verificações de integridade ou são criptografados para impedir a criação de objetos hostis ou adulteração de dados. (C5)	✓	✓	✓	502
5.5.2	Verifique se a aplicação restringe corretamente os analisadores de XML para usar apenas a configuração mais restritiva possível e para garantir que recursos não seguros, como a resolução de entidades externas, sejam desabilitados para evitar ataques de XML eXternal Entity (XXE).	✓	✓	✓	611
5.5.3	Verifique se a desserialização de dados não confiáveis é evitada ou protegida em código personalizado e bibliotecas de terceiros (como analisadores JSON, XML e YAML).	✓	✓	✓	502
5.5.4	Verifique se, ao analisar JSON em navegadores ou back-ends baseados em JavaScript, JSON.parse é usado para analisar o documento JSON. Não use eval() para analisar JSON.	✓	✓	✓	95

Referências

Para mais informações, consulte também:

- OWASP Testing Guide 4.0: Input Validation Testing
- OWASP Cheat Sheet: Input Validation
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution
- OWASP LDAP Injection Cheat Sheet
- OWASP Testing Guide 4.0: Client Side Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP Java Encoding Project
- OWASP Mass Assignment Prevention Cheat Sheet
- DOMPurify - Client-side HTML Sanitization Library
- XML External Entity (XXE) Prevention Cheat Sheet

Para obter mais informações sobre escape automático, consulte:

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems
- AngularJS Strict Contextual Escaping
- AngularJS ngBind
- Angular Sanitization
- Angular Security
- ReactJS Escaping
- Improperly Controlled Modification of Dynamically-Determined Object Attributes

Para obter mais informações sobre desserialização, consulte:

- OWASP Deserialization Cheat Sheet
- OWASP Deserialization of Untrusted Data Guide

V6 Criptografia armazenada

Objetivo de controle

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- Todos os módulos criptográficos falham de maneira segura e os erros são tratados corretamente.
- Um gerador de números aleatórios adequado é usado.
- O acesso às chaves é gerenciado com segurança.

V6.1 Classificação de dados

O ativo mais importante são os dados processados, armazenados ou transmitidos por uma aplicação. Sempre realize uma avaliação de impacto na privacidade para classificar corretamente as necessidades de proteção de dados de quaisquer dados armazenados.

#	Descrição	L1	L2	L3	CWE
6.1.1	Verifique se os dados privados regulamentados são armazenados criptografados enquanto estão em repouso, como informações de identificação pessoal (PII), informações pessoais confidenciais ou dados avaliados que provavelmente estão sujeitos ao GDPR da UE.		✓	✓	311
6.1.2	Verifique se os dados de saúde regulamentados são armazenados criptografados enquanto estão em repouso, como registros médicos, detalhes de dispositivos médicos ou registros de pesquisa anonimizados.		✓	✓	311
6.1.3	Verifique se os dados financeiros regulamentados são armazenados criptografados enquanto estão inativos, como contas financeiras, inadimplência ou histórico de crédito, registros fiscais, histórico de pagamentos, beneficiários ou registros de pesquisa ou mercado anonimizados.		✓	✓	311

V6.2 Algoritmos

Avanços recentes em criptografia significam que algoritmos e comprimentos de chave anteriormente seguros não são mais seguros ou suficientes para proteger os dados. Portanto, deve ser possível alterar os algoritmos.

Embora esta seção não seja facilmente testada quanto à penetração, os desenvolvedores devem considerá-la como obrigatória, mesmo que L1 esteja faltando na maioria dos itens.

#	Descrição	L1	L2	L3	CWE
6.2.1	Verifique se todos os módulos criptográficos falham com segurança e se os erros são tratados de forma a não permitir ataques de Padding Oracle.	✓	✓	✓	310
6.2.2	Verifique se são usados algoritmos, modos e bibliotecas criptográficos comprovados pelo setor ou aprovados pelo governo, em vez de criptografia codificada personalizada. (C8)		✓	✓	327
6.2.3	Verifique se o vetor de inicialização de criptografia, a configuração de cifra e os modos de bloqueio estão configurados com segurança usando as recomendações mais recentes.		✓	✓	326
6.2.4	Verifique se o número aleatório, algoritmos de criptografia ou hash, comprimentos de chave, rodadas, cifras ou modos podem ser reconfigurados, atualizados ou trocados a qualquer momento, para proteger contra quebras criptográficas. (C8)		✓	✓	326

#	Descrição	L1	L2	L3	CWE
6.2.5	Verifique se os modos de bloco inseguros conhecidos (ou seja, ECB, etc.), modos de preenchimento (ou seja, PKCS#1 v1.5, etc.), cifras com tamanhos de bloco pequenos (ou seja, Triple-DES, Blowfish, etc.) e algoritmos de hash fracos (ou seja, MD5, SHA1, etc.) não são usados, a menos que sejam necessários para compatibilidade com versões anteriores.		✓	✓	326
6.2.6	Verifique se nonces, vetores de inicialização e outros números de uso único não devem ser usados mais de uma vez com uma determinada chave de criptografia. O método de geração deve ser apropriado para o algoritmo que está sendo usado.		✓	✓	326
6.2.7	Verifique se os dados criptografados são autenticados por meio de assinaturas, modos de cifra autenticados ou HMAC para garantir que o texto cifrado não seja alterado por uma parte não autorizada.			✓	326
6.2.8	Verifique se todas as operações criptográficas são de tempo constante, sem operações de 'curto-circuito' em comparações, cálculos ou retornos, para evitar vazamento de informações.			✓	385

V6.3 Valores aleatórios

A verdadeira geração de números pseudoaleatórios (PRNG) é incrivelmente difícil de acertar. Geralmente, boas fontes de entropia dentro de um sistema serão rapidamente esgotadas se usadas em excesso, mas fontes com menos aleatoriedade podem levar a chaves e segredos previsíveis.

#	Descrição	L1	L2	L3	CWE
6.3.1	Verifique se todos os números aleatórios, nomes de arquivos aleatórios, GUIDs aleatórios e strings aleatórias são gerados usando o gerador de números aleatórios criptograficamente seguro aprovado do módulo criptográfico quando esses valores aleatórios não devem ser adivinhados por um invasor.		✓	✓	338
6.3.2	Verifique se os GUIDs aleatórios são criados usando o algoritmo GUID v4 e um gerador de números pseudo-aleatórios criptograficamente seguro (CSPRNG). GUIDs criados usando outros geradores de números pseudo-aleatórios podem ser previsíveis.		✓	✓	338
6.3.3	Verifique se os números aleatórios são criados com a entropia adequada mesmo quando a aplicação está sob carga pesada ou se a aplicação se degrada normalmente nessas circunstâncias.			✓	338

V6.4 Gestão de segredo

Embora esta seção não seja facilmente testada quanto à penetração, os desenvolvedores devem considerá-la como obrigatória, mesmo que L1 esteja faltando na maioria dos itens.

#	Descrição	L1	L2	L3	CWE
6.4.1	Verifique se uma solução de gestão de segredos, como um cofre de chaves, é usada para criar, armazenar, controlar o acesso e destruir segredos com segurança. (C8)		✓	✓	798
6.4.2	Verifique se o material da chave não está exposto à aplicação, mas usa um módulo de segurança isolado como um cofre para operações criptográficas. (C8)		✓	✓	320

Referências

Para mais informações, consulte também:

- OWASP Testing Guide 4.0: Testing for weak Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
- FIPS 140-2

V7 Tratamento e registro de erros

Objetivo de controle

O principal objetivo do tratamento e registro de erros é fornecer informações úteis para o usuário, administradores e equipes de resposta a incidentes. O objetivo não é criar grandes quantidades de logs, mas logs de alta qualidade, com mais sinal do que ruído descartado.

Logs de alta qualidade geralmente contêm dados confidenciais e devem ser protegidos conforme as leis ou diretivas locais de privacidade de dados. Isso deve incluir:

- Não coletar ou registrar informações confidenciais, a menos que seja especificamente necessário.
- Garantir que todas as informações registradas sejam tratadas com segurança e protegidas conforme a sua classificação de dados.
- Garantir que os logs não sejam armazenados para sempre, mas tenham um tempo de vida absoluto o mais curto possível.

Se os logs contiverem dados privados ou confidenciais, cuja definição varia de país para país, os logs se tornam algumas das informações mais confidenciais mantidas pela aplicação e, assim, muito atraentes para os invasores.

Também é importante garantir que a aplicação falhe com segurança e que erros não divulguem informações desnecessárias.

V7.1 Conteúdo do registro

Registrar informações confidenciais é perigoso - os próprios logs se tornam classificados, o que significa que precisam ser criptografados, sujeitos a políticas de retenção e devem ser divulgados em auditorias de segurança. Certifique-se de que apenas as informações necessárias sejam mantidas em logs e, certamente, nenhum pagamento, credenciais (incluindo tokens de sessão), informações confidenciais ou de identificação pessoal.

V7.1 abrange OWASP Top 10 2017:A10. Como 2017:A10 e esta seção não são passíveis de teste de penetração, é importante para:

- Desenvolvedores devem garantir total conformidade com esta seção, como se todos os itens fossem marcados como L1
- Penetration testers para validar a conformidade total de todos os itens na V7.1 por meio de entrevista, capturas de tela ou afirmação

#	Descrição	L1	L2	L3	CWE
7.1.1	Verifique se a aplicação não registra credenciais ou detalhes de pagamento. Os tokens de sessão devem ser armazenados apenas em logs num formato de hash irreversível. (C9, C10)	✓	✓	✓	532
7.1.2	Verifique se a aplicação não registra outros dados confidenciais conforme definido nas leis de privacidade locais ou na política de segurança relevante. (C9)	✓	✓	✓	532
7.1.3	Verifique se a aplicação registra eventos relevantes de segurança, incluindo eventos de autenticação bem-sucedidos e com falha, falhas de controle de acesso, falhas de desserialização e falhas de validação de input. (C5, C7)		✓	✓	778
7.1.4	Verifique se cada evento de log inclui as informações necessárias que permitem uma investigação detalhada da linha do tempo quando um evento ocorre. (C9)		✓	✓	778

V7.2 Processamento de Log

O registro oportuno é crítico para eventos de auditoria, triagem e escalonamento. Certifique-se de que os logs da aplicação estejam claros e possam ser facilmente monitorados e analisados localmente ou enviados para um sistema de monitoramento remoto.

V7.2 abrange OWASP Top 10 2017:A10. Como 2017:A10 e esta seção não são passíveis de teste de penetração, é importante para:

- Desenvolvedores devem garantir total conformidade com esta seção, como se todos os itens fossem marcados como L1
- Penetration testers para validar a conformidade total de todos os itens na V7.2 por entrevista, capturas de tela ou afirmação

#	Descrição	L1	L2	L3	CWE
7.2.1	Verifique se todas as decisões de autenticação são registradas, sem armazenar senhas ou tokens de sessão confidenciais. Isso deve incluir solicitações com metadados relevantes necessários para investigações de segurança.		✓	✓	778
7.2.2	Verifique se todas as decisões de controle de acesso podem ser registradas e todas as decisões com falha são registradas. Isso deve incluir solicitações com metadados relevantes necessários para investigações de segurança.		✓	✓	285

V7.3 Proteção de Log

Logs que podem ser modificados ou excluídos trivialmente são inúteis para investigações e processos. A divulgação de logs pode expor detalhes internos sobre a aplicação ou os dados que ele contém. Deve-se tomar cuidado ao proteger os logs contra divulgação, modificação ou exclusão não autorizada.

#	Descrição	L1	L2	L3	CWE
7.3.1	Verifique se todos os componentes de log codificam os dados adequadamente para evitar a injeção de log. (C9)		✓	✓	117
7.3.2	[EXCLUÍDO, DUPLICADO DE 7.3.1]				
7.3.3	Verifique se os logs de segurança estão protegidos contra acesso e modificação não autorizados. (C9)		✓	✓	200
7.3.4	Verifique se as fontes de horário estão sincronizadas com a hora e o fuso horário corretos. Considere fortemente o registro apenas em UTC se os sistemas forem globais para auxiliar na análise forense pós-incidente. (C9)		✓	✓	

Observação: a codificação de log (7.3.1) é difícil de testar e revisar usando ferramentas dinâmicas automatizadas e testes de penetração, mas arquitetos, desenvolvedores e revisores de código-fonte devem considerá-la um requisito L1.

V7.4 Tratamento de Erros

O objetivo do tratamento de erros é permitir que a aplicação forneça eventos relevantes de segurança para monitoramento, triagem e escalonamento. O objetivo não é criar logs. Ao registrar eventos relacionados à segurança, certifique-se de que haja uma finalidade para o registro e que ele possa ser distinguido pelo SIEM ou software de análise.

#	Descrição	L1	L2	L3	CWE
7.4.1	Verifique se uma mensagem genérica é exibida quando ocorre um erro inesperado ou sensível à segurança, possivelmente com uma ID exclusiva que a equipe de suporte pode usar para investigar. (C10)	✓	✓	✓	210
7.4.2	Verifique se o tratamento de exceção (ou um equivalente funcional) é usado na base de código para contabilizar as condições de erro esperadas e inesperadas. (C10)		✓	✓	544
7.4.3	Verifique se um manipulador de erro de "último recurso" está definido para capturar todas as exceções não tratadas. (C10)		✓	✓	431

Nota: Certas linguagens, como Swift e Go - e através da prática de design comum - muitas linguagens funcionais, não suportam exceções ou manipuladores de eventos de último recurso. Nesse caso, arquitetos e desenvolvedores devem usar um padrão, linguagem ou estrutura amigável para garantir que as aplicações possam manipular com segurança eventos excepcionais, inesperados ou relacionados à segurança.

Referências

Para mais informações, consulte também:

- OWASP Testing Guide 4.0 content: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages

V8 Proteção de Dados

Objetivo de controle

Existem três elementos-chave para uma boa proteção de dados: Confidencialidade, Integridade e Disponibilidade (CIA). Este padrão assume que a proteção de dados é aplicada num sistema confiável, como um servidor, reforçado e possui proteções suficientes.

As aplicações devem assumir que todos os dispositivos do usuário estão comprometidos de alguma forma. Quando uma aplicação transmite ou armazena informações confidenciais em dispositivos inseguros, como computadores, telefones e tablets compartilhados, a aplicação é responsável por garantir que os dados armazenados nesses dispositivos sejam criptografados e não possam ser facilmente obtidos, alterados ou divulgados de forma ilícita.

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de proteção de dados de alto nível:

- Confidencialidade: Os dados devem ser protegidos contra observação ou divulgação não autorizada, tanto em trânsito quanto quando armazenados.
- Integridade: os dados devem ser protegidos contra criação, alteração ou exclusão maliciosa por invasores não autorizados.
- Disponibilidade: Os dados devem estar disponíveis para usuários autorizados conforme necessário.

V8.1 Proteção Geral de Dados

#	Descrição	L1	L2	L3	CWE
8.1.1	Verifique se a aplicação protege os dados confidenciais de serem armazenados em cache nos componentes do servidor, como balanceadores de carga e caches de aplicações.		✓	✓	524
8.1.2	Verifique se todas as cópias em cache ou temporárias de dados confidenciais armazenados no servidor estão protegidas contra acesso não autorizado ou eliminadas/invalidadas depois que o usuário autorizado acessa os dados confidenciais.		✓	✓	524
8.1.3	Verifique se a aplicação minimiza o número de parâmetros numa solicitação, como campos ocultos, variáveis Ajax, cookies e valores de cabeçalho.		✓	✓	233
8.1.4	Verifique se a aplicação pode detectar e alertar sobre números anormais de solicitações, como por IP, usuário, total por hora ou dia ou o que fizer sentido para a aplicação.		✓	✓	770
8.1.5	Verifique se os backups regulares de dados importantes são executados e se a restauração de teste de dados é realizada.			✓	19
8.1.6	Verifique se os backups são armazenados com segurança para evitar que os dados sejam roubados ou corrompidos.			✓	19

V8.2 Proteção de dados do lado do cliente

#	Descrição	L1	L2	L3	CWE
8.2.1	Verifique se a aplicação define cabeçalhos anti-cache suficientes para que dados confidenciais não sejam armazenados em cache em navegadores modernos.	✓	✓	✓	525

#	Descrição	L1	L2	L3	CWE
8.2.2	Verifique se os dados armazenados no armazenamento do navegador (como localStorage, sessionStorage, IndexedDB ou cookies) não contêm dados confidenciais.	✓	✓	✓	922
8.2.3	Verifique se os dados autenticados foram apagados do armazenamento do cliente, como o DOM do navegador, após o término do cliente ou da sessão.	✓	✓	✓	922

V8.3 Dados privados confidenciais

Esta seção ajuda a proteger dados confidenciais de serem criados, lidos, atualizados ou excluídos sem autorização, especialmente em grandes quantidades.

A conformidade com esta seção implica conformidade com o Controle de acesso V4 e, em particular, V4.2. Por exemplo, para proteger contra atualizações não autorizadas ou divulgação de informações pessoais confidenciais, é necessário aderir à V4.2.1. Por favor, cumpra esta seção e V4 para cobertura completa.

Observação: os regulamentos e as leis de privacidade, como os princípios de privacidade australianos APP-11 ou GDPR, afetam diretamente como as aplicações devem abordar a implementação de armazenamento, uso e transmissão de informações pessoais confidenciais. Isso varia de penalidades severas a conselhos simples. Consulte as leis e regulamentos locais e consulte um especialista em privacidade ou advogado qualificado, conforme necessário.

#	Descrição	L1	L2	L3	CWE
8.3.1	Verifique se os dados confidenciais são enviados ao servidor no corpo ou nos cabeçalhos da mensagem HTTP e se os parâmetros da string de consulta de qualquer verbo HTTP não contêm dados confidenciais.	✓	✓	✓	319
8.3.2	Verifique se os usuários têm um método para remover ou exportar os seus dados sob demanda.	✓	✓	✓	212
8.3.3	Verifique se os usuários recebem uma linguagem clara sobre a coleta e o uso das informações pessoais fornecidas e se os usuários forneceram consentimento para o uso desses dados antes de serem usados de qualquer forma.	✓	✓	✓	285
8.3.4	Verifique se todos os dados confidenciais criados e processados pela aplicação foram identificados e certifique-se de que haja uma política sobre como lidar com dados confidenciais. (C8)	✓	✓	✓	200
8.3.5	Verifique se o acesso a dados confidenciais é auditado (sem registrar os próprios dados confidenciais), se os dados forem coletados conforme as diretivas de proteção de dados relevantes ou se o registro de acesso for necessário.		✓	✓	532
8.3.6	Verifique se as informações confidenciais contidas na memória são substituídas assim que não são mais necessárias para mitigar ataques de despejo de memória, usando zeros ou dados aleatórios.		✓	✓	226
8.3.7	Verifique se as informações confidenciais ou privadas que devem ser criptografadas estão criptografadas usando algoritmos aprovados que fornecem confidencialidade e integridade. (C8)		✓	✓	327
8.3.8	Verifique se as informações pessoais confidenciais estão sujeitas à classificação de retenção de dados, de modo que os dados antigos ou desatualizados sejam excluídos automaticamente, de acordo com a programação ou conforme a situação exigir.		✓	✓	285

Ao considerar a proteção de dados, uma consideração primária deve ser a extração em massa ou modificação, ou uso excessivo. Por exemplo, muitos sistemas de mídia social permitem apenas que os usuários adicionem 100 novos amigos por dia, mas de qual sistema essas solicitações vieram não é importante. Uma plataforma bancária pode querer bloquear mais de 5 transações por hora, transferindo mais de 1000 euros de fundos para instituições externas. É provável que os requisitos de cada sistema sejam muito diferentes, portanto, decidir sobre "anormal" deve considerar o modelo de ameaça e o risco comercial. Critérios importantes são a capacidade de detectar, impedir ou, preferencialmente, bloquear tais ações em massa anormais.

Referências

Para mais informações, consulte também:

- Consider using Security Headers website to check security and anti-caching headers
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network

V9 Comunicação

Objetivo de controle

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- Requer TLS ou criptografia forte, independente da sensibilidade do conteúdo.
- Siga as orientações mais recentes, incluindo:
  - Conselho de configuração
  - Algoritmos e cifras preferidos
- Evite algoritmos e cifras fracos ou prestes a serem obsoletos, exceto como último recurso
- Desabilite algoritmos e cifras obsoletos ou inseguros conhecidos.

Dentro desses requisitos:

- Mantenha-se atualizado com os conselhos recomendados do setor sobre a configuração segura do TLS, pois ele muda com frequência (geralmente devido a quebras catastróficas nos algoritmos e cifras existentes).
- Use as versões mais recentes das ferramentas de revisão de configuração TLS para configurar a ordem preferida e a seleção do algoritmo.
- Verifique a sua configuração periodicamente para garantir que a comunicação segura esteja sempre presente e eficaz.

V9.1 Segurança de comunicação do cliente

Certifique-se de que todas as mensagens do cliente sejam enviadas por redes criptografadas, usando TLS 1.2 ou posterior. Use ferramentas atualizadas para revisar a configuração do cliente regularmente.

#	Descrição	L1	L2	L3	CWE
9.1.1	Verifique se o TLS é usado para toda a conectividade do cliente e não retrocede para comunicações inseguras ou não criptografadas. (C8)	✓	✓	✓	319
9.1.2	Verifique, usando ferramentas de teste de TLS atualizadas, que somente conjuntos de cifras fortes estão ativados, com os conjuntos de cifras mais fortes definidos como preferenciais.	✓	✓	✓	326
9.1.3	Verifique se apenas as versões recomendadas mais recentes do protocolo TLS estão habilitadas, como TLS 1.2 e TLS 1.3. A versão mais recente do protocolo TLS deve ser a opção preferida.	✓	✓	✓	326

V9.2 Segurança de comunicação do servidor

As comunicações do servidor são mais que apenas HTTP. Conexões seguras de e para outros sistemas, como sistemas de monitoramento, ferramentas de gestão, acesso remoto e ssh, middleware, banco de dados, mainframes, parceiros ou sistemas de origem externa - devem estar em vigor. Tudo isso deve ser criptografado para evitar "difícil por fora, trivialmente fácil de interceptar por dentro".

#	Descrição	L1	L2	L3	CWE
9.2.1	Verifique se as conexões de e para o servidor usam certificados TLS confiáveis. Onde certificados gerados internamente ou autoassinados são usados, o servidor deve ser configurado para confiar apenas em CAs internas específicas e certificados autoassinados específicos. Todos os outros devem ser rejeitados.		✓	✓	295

#	Descrição	L1	L2	L3	CWE
9.2.2	Verifique se as comunicações criptografadas, como TLS, são usadas para todas as conexões de input e output, incluindo portas de gestão, monitoramento, autenticação, API ou chamadas de Web Service, banco de dados, nuvem, serverless, mainframe, externas e conexões de parceiros. O servidor não deve recorrer a protocolos inseguros ou não criptografados.		✓	✓	319
9.2.3	Verifique se todas as conexões criptografadas com sistemas externos que envolvem informações ou funções confidenciais são autenticadas.		✓	✓	287
9.2.4	Verifique se a revogação de certificação adequada, como o grampeamento do protocolo de status de certificado on-line (OCSP), está habilitada e configurada.		✓	✓	299
9.2.5	Verifique se as falhas de conexão TLS de back-end são registradas.			✓	544

Referências

Para mais informações, consulte também:

- OWASP – TLS Cheat Sheet
- OWASP - Pinning Guide
- Notas sobre “Modos aprovados de TLS”:
  - No passado, a ASVS referia-se ao padrão americano FIPS 140-2, mas como um padrão global, a aplicação dos padrões americanos pode ser difícil, contraditória ou confusa.
  - Um método melhor para obter conformidade com a seção 9.1 seria revisar guias como Mozilla's Server Side TLS ou generate known good configurations e use ferramentas de avaliação TLS conhecidas e atualizadas para obter o nível de segurança desejado.

V10 Código malicioso

Objetivo de controle

Certifique-se de que o código satisfaça os seguintes requisitos de alto nível:

- A atividade maliciosa é tratada de forma segura e adequada para não afetar o restante da aplicação.
- Não possui bombas-relógio ou outros ataques baseados no tempo.
- Não "telefone para casa" para destinos maliciosos ou não autorizados.
- Não possui portas traseiras, ovos de Páscoa, ataques de salame, rootkits ou código não autorizado que possa ser controlado por um invasor.

Encontrar um código malicioso é a prova da negativa, impossível de validar completamente. Devem ser envidados os melhores esforços para garantir que o código não tenha código malicioso inerente ou funcionalidade indesejada.

V10.1 Integridade do código

A melhor defesa contra códigos maliciosos é "confiar, mas verificar". A introdução de código não autorizado ou malicioso no código costuma ser uma ofensa criminal em muitas jurisdições. Políticas e procedimentos devem tornar claras as sanções relativas a códigos maliciosos.

Os principais desenvolvedores devem revisar regularmente os check-ins de código, especialmente aqueles que podem acessar funções de tempo, E/S ou rede.

#	Descrição	L1	L2	L3	CWE
10.1.1	Verifique se uma ferramenta de análise de código está em uso e pode detectar códigos potencialmente maliciosos, como funções de tempo, operações de arquivo inseguras e conexões de rede.			✓	749

V10.2 Pesquisa de código malicioso

O código malicioso é extremamente raro e difícil de detectar. A revisão manual do código linha por linha pode ajudar a procurar por bombas lógicas, mas mesmo o revisor de código mais experiente terá dificuldade em encontrar código malicioso, mesmo que saiba que ele existe.

A conformidade com esta seção não é possível sem acesso completo ao código-fonte, incluindo bibliotecas de terceiros.

#	Descrição	L1	L2	L3	CWE
10.2.1	Verifique se o código-fonte da aplicação e as bibliotecas de terceiros não contêm recursos não autorizados de telefone residencial ou coleta de dados. Onde tal funcionalidade existir, obtenha a permissão do usuário para que ela opere antes de coletar quaisquer dados.		✓	✓	359
10.2.2	Verifique se a aplicação não solicita permissões desnecessárias ou excessivas para recursos ou sensores relacionados à privacidade, como contatos, câmeras, microfones ou localização.		✓	✓	272
10.2.3	Verifique se o código-fonte da aplicação e as bibliotecas de terceiros não contêm backdoors, como contas ou chaves codificadas ou adicionais não documentadas, ofuscação de código, blobs binários não documentados, rootkits ou antidepuração, recursos de depuração inseguros ou de outra forma fora de data, funcionalidade insegura ou oculta que pode ser usada maliciosamente se descoberta.			✓	507

#	Descrição	L1	L2	L3	CWE
10.2.4	Verifique se o código-fonte da aplicação e as bibliotecas de terceiros não contêm bombas-relógio procurando por funções relacionadas a data e hora.			✓	511
10.2.5	Verifique se o código-fonte da aplicação e as bibliotecas de terceiros não contêm código mal-intencionado, como ataques de salame, desvios lógicos ou bombas lógicas.			✓	511
10.2.6	Verifique se o código-fonte da aplicação e as bibliotecas de terceiros não contêm ovos de Páscoa ou qualquer outra funcionalidade potencialmente indesejada.			✓	507

V10.3 Integridade da aplicação

Depois que uma aplicação é implantado, o código malicioso ainda pode ser inserido. as aplicações precisam se proteger contra-ataques comuns, como a execução de código não assinado de fontes não confiáveis e invasões de subdomínio.

O cumprimento desta seção provavelmente será operacional e contínuo.

#	Descrição	L1	L2	L3	CWE
10.3.1	Verifique se a aplicação possui um recurso de atualização automática de cliente ou servidor, as atualizações devem ser obtidas por canais seguros e assinadas digitalmente. O código de atualização deve validar a assinatura digital da atualização antes de instalar ou executar a atualização.	✓	✓	✓	16
10.3.2	Verifique se a aplicação emprega proteções de integridade, como assinatura de código ou integridade de sub-recurso. A aplicação não deve carregar ou executar código de fontes não confiáveis, como carregamento de inclusões, módulos, plug-ins, códigos ou bibliotecas de fontes não confiáveis ou da Internet.	✓	✓	✓	353
10.3.3	Verifique se a aplicação tem proteção contra invasões de subdomínio se a aplicação depender de inputs DNS ou subdomínios DNS, como nomes de domínio expirados, ponteiros DNS ou CNAMEs desatualizados, projetos expirados em repositórios públicos de código-fonte ou APIs de nuvem temporárias, funções serverless, ou depósitos de armazenamento (*autogen-bucket-id*.cloud.example.com) ou similar. As proteções podem incluir a garantia de que os nomes DNS usados pelas aplicações sejam verificados regularmente quanto à expiração ou alteração.	✓	✓	✓	350

Referências

Para mais informações, consulte também:

- Hostile Subdomain Takeover, Detectify Labs
- Hijacking of abandoned subdomains part 2, Detectify Labs

V11 Lógica de Negócios

Objetivo de controle

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- O fluxo da lógica de negócios é sequencial, processado em ordem e não pode ser ignorado.
- A lógica de negócios inclui limites para detectar e prevenir ataques automatizados, como pequenas transferências contínuas de fundos ou adição de um milhão de amigos, um por vez, e assim por diante.
- Os fluxos de lógica de negócios de alto valor consideraram casos de abuso e atores mal-intencionados com proteções contra falsificação, adulteração, divulgação de informações e ataques de elevação de privilégio.

V11.1 Segurança da Lógica de Negócios

A segurança da lógica de negócios é tão individual para cada aplicação que nenhuma lista de verificação será aplicada. A segurança da lógica de negócios deve ser projetada para proteger contra prováveis ameaças externas - ela não pode ser adicionada usando firewalls de aplicações da Web ou comunicações seguras. Recomendamos o uso de modelagem de ameaças durante os sprints de design, por exemplo, usando o OWASP Cornucopia ou ferramentas semelhantes.

#	Descrição	L1	L2	L3	CWE
11.1.1	Verifique se a aplicação processará apenas fluxos de lógica de negócios para o mesmo usuário em ordem sequencial de etapas e sem pular etapas.	✓	✓	✓	841
11.1.2	Verifique se a aplicação processará apenas fluxos de lógica de negócios com todas as etapas sendo processadas em tempo humano realista, ou seja, as transações não são enviadas muito rapidamente.	✓	✓	✓	799
11.1.3	Verifique se a aplicação tem limites apropriados para ações ou transações de negócios específicas que são aplicadas corretamente por usuário.	✓	✓	✓	770
11.1.4	Verifique se a aplicação possui controles antiautomação para proteção contra chamadas excessivas, como exfiltração de dados em massa, solicitações de lógica de negócios, uploads de arquivos ou ataques de negação de serviço.	✓	✓	✓	770
11.1.5	Verifique se a aplicação tem limites de lógica de negócios ou validação para proteger contra prováveis riscos ou ameaças de negócios, identificados usando modelagem de ameaças ou metodologias semelhantes.	✓	✓	✓	841
11.1.6	Verifique se a aplicação não sofre de problemas de "Time Of Check to Time Of Use" (TOCTOU) ou outras condições de corrida para operações confidenciais.		✓	✓	367
11.1.7	Verifique se a aplicação monitora eventos ou atividades incomuns de uma perspectiva de lógica de negócios. Por exemplo, tentativas de executar ações fora de ordem ou ações que um usuário normal nunca tentaria. (C9)		✓	✓	754
11.1.8	Verifique se a aplicação possui alertas configuráveis quando ataques automatizados ou atividades incomuns são detectados.		✓	✓	390

Referências

Para mais informações, consulte também:

- OWASP Web Security Testing Guide 4.1: Business Logic Testing

- A antiautomação pode ser alcançada de várias maneiras, incluindo o uso de OWASP AppSensor e OWASP Automated Threats to Web Applications
- OWASP AppSensor pode também ajudar com Detecção e Respostas de Ataques.
- OWASP Cornucopia

V12 Arquivos e Recursos

Objetivo de controle

Certifique-se de que uma aplicação verificado atenda aos seguintes requisitos de alto nível:

- Dados de arquivos não confiáveis devem ser tratados adequadamente e de maneira segura.
- Dados de arquivos não confiáveis obtidos de fontes não confiáveis são armazenados fora da raiz da web e com permissões limitadas.

V12.1 Carregamento de arquivo

Embora as bombas zip sejam eminentemente testáveis usando técnicas de teste de penetração, elas são consideradas L2 e acima para encorajar a consideração de design e desenvolvimento com testes manuais cuidadosos e para evitar testes de penetração manuais automatizados ou não qualificados de uma condição de negação de serviço.

#	Descrição	L1	L2	L3	CWE
12.1.1	Verifique se a aplicação não aceitará arquivos grandes que possam encher o armazenamento ou causar uma negação de serviço.	✓	✓	✓	400
12.1.2	Verifique se a aplicação verifica os arquivos compactados (por exemplo, zip, gz, docx, odt) em relação ao tamanho máximo descompactado permitido e ao número máximo de arquivos antes de descompactar o arquivo.		✓	✓	409
12.1.3	Verifique se uma cota de tamanho de arquivo e um número máximo de arquivos por usuário são aplicados para garantir que um único usuário não possa preencher o armazenamento com muitos arquivos ou arquivos excessivamente grandes.		✓	✓	770

V12.2 Integridade do arquivo

#	Descrição	L1	L2	L3	CWE
12.2.1	Verifique se os arquivos obtidos de fontes não confiáveis são validados como sendo do tipo esperado com base no conteúdo do arquivo.		✓	✓	434

V12.3 Execução de Arquivo

#	Descrição	L1	L2	L3	CWE
12.3.1	Verifique se os metadados de nome de arquivo enviados pelo usuário não são usados diretamente pelo sistema ou sistemas de arquivos de estrutura e se uma API de URL é usada para proteção contra travessia de caminho.	✓	✓	✓	22
12.3.2	Verifique se os metadados de nome de arquivo enviados pelo usuário são validados ou ignorados para impedir a divulgação, criação, atualização ou remoção de arquivos locais (LFI).	✓	✓	✓	73
12.3.3	Verifique se os metadados de nome de arquivo enviados pelo usuário são validados ou ignorados para evitar a divulgação ou execução de arquivos remotos por ataques de inclusão de arquivo remoto (RFI) ou falsificação de solicitação do lado do servidor (SSRF).	✓	✓	✓	98

#	Descrição	L1	L2	L3	CWE
12.3.4	Verifique se a aplicação protege contra download de arquivo reflexivo (RFD) validando ou ignorando nomes de arquivo enviados pelo usuário num parâmetro JSON, JSONP ou URL, o cabeçalho Content-Type de resposta deve ser definido como text/plain e o cabeçalho Content-Disposition deve ter um nome de arquivo fixo.	✓	✓	✓	641
12.3.5	Verifique se os metadados de arquivos não confiáveis não são usados diretamente com API ou bibliotecas do sistema, para proteger contra a injeção de comandos do sistema operacional.	✓	✓	✓	78
12.3.6	Verifique se a aplicação não inclui e executa funcionalidades de fontes não confiáveis, como redes de distribuição de conteúdos não verificados, bibliotecas JavaScript, bibliotecas de nó npm ou DLLs do lado do servidor.		✓	✓	829

V12.4 Armazenamento de Arquivos

#	Descrição	L1	L2	L3	CWE
12.4.1	Verifique se os arquivos obtidos de fontes não confiáveis são armazenados fora da raiz da web, com permissões limitadas.	✓	✓	✓	552
12.4.2	Verifique se os arquivos obtidos de fontes não confiáveis são verificados por scanners antivírus para impedir o upload e a exibição de conteúdo malicioso conhecido.	✓	✓	✓	509

V12.5 Download do arquivo

#	Descrição	L1	L2	L3	CWE
12.5.1	Verifique se a camada da web está configurada para servir apenas arquivos com extensões de arquivo específicas para evitar informações não intencionais e vazamento de código-fonte. Por exemplo, arquivos de backup (por exemplo, .bak), arquivos de trabalho temporários (por exemplo, .swp), arquivos compactados (.zip, .tar.gz, etc) e outras extensões comumente usadas por editores devem ser bloqueados, a menos que necessário.	✓	✓	✓	552
12.5.2	Verifique se as solicitações diretas para arquivos carregados nunca serão executadas como conteúdo HTML/JavaScript.	✓	✓	✓	434

V12.6 Proteção SSRF

#	Descrição	L1	L2	L3	CWE
12.6.1	Verifique se o servidor da web ou de aplicações está configurado com uma lista de permissões de recursos ou sistemas para os quais o servidor pode enviar solicitações ou carregar dados/arquivos.	✓	✓	✓	918

Referências

Para mais informações, consulte também:

- File Extension Handling for Sensitive Information
- Reflective file download by Oren Hafif
- OWASP Third Party JavaScript Management Cheat Sheet

V13 API e Web Service

Objetivo de controle

Certifique-se de que uma aplicação verificado que usa APIs de camada de serviço confiável (geralmente usando JSON, XML ou GraphQL) tenha:

- Autenticação adequada, gestão de sessão e autorização de todos os serviços da web.
- Validação de input de todos os parâmetros que transitam de um nível de confiança inferior para superior.
- Controles de segurança eficazes para todos os tipos de API, incluindo nuvem e API serverless

Por favor, leia este capítulo em combinação com todos os outros capítulos neste mesmo nível; não duplicamos mais questões de autenticação ou gestão de sessões de API.

V13.1 Segurança genérica de Web Service

#	Descrição	L1	L2	L3	CWE
13.1.1	Verifique se todos os componentes da aplicação usam as mesmas codificações e analisadores para evitar ataques de análise que exploram URI diferente ou comportamento de análise de arquivo que pode ser usado em ataques SSRF e RFI.	✓	✓	✓	116
13.1.2	[EXCLUÍDO, DUPLICADO DE 4.3.1]				
13.1.3	Verifique se os URLs da API não expõem informações confidenciais, como a chave da API, tokens de sessão, etc.	✓	✓	✓	598
13.1.4	Verifique se as decisões de autorização são feitas no URI, impostas por segurança programática ou declarativa no controlador ou roteador, e no nível do recurso, impostas por permissões baseadas em modelo.		✓	✓	285
13.1.5	Verifique se as solicitações que contêm tipos de conteúdo inesperados ou ausentes são rejeitadas com cabeçalhos apropriados (status de resposta HTTP 406 inaceitável ou 415 tipo de mídia não compatível).		✓	✓	434

V13.2 Serviço Web RESTful

A validação do esquema JSON está em fase de rascunho de padronização (consulte as referências). Ao considerar o uso da validação do esquema JSON, que é a prática recomendada para serviços da Web RESTful, considere o uso dessas estratégias adicionais de validação de dados em combinação com a validação do esquema JSON:

- Validação de análise do objeto JSON, como se houver elementos ausentes ou extras.
- Validação dos valores do objeto JSON usando métodos de validação de input padrão, como tipo de dados, formato de dados, comprimento, etc.
- e validação de esquema JSON formal.

Assim que o padrão de validação do esquema JSON for formalizado, a ASVS atualizará os seus conselhos nessa área. Monitore cuidadosamente todas as bibliotecas de validação de esquema JSON em uso, já que elas precisarão ser atualizadas regularmente até que o padrão seja formalizado e os bugs sejam eliminados das implementações de referência.

#	Descrição	L1	L2	L3	CWE
13.2.1	Verifique se os métodos HTTP RESTful ativados são uma escolha válida para o usuário ou ação, como impedir que usuários normais usem DELETE ou PUT em recursos ou API protegidos.	✓	✓	✓	650
13.2.2	Verifique se a validação do esquema JSON está em vigor e verificada antes de aceitar a input.	✓	✓	✓	20
13.2.3	Verifique se os serviços da Web RESTful que utilizam cookies estão protegidos contra falsificação de solicitação entre sites por meio do uso de pelo menos um ou mais dos seguintes: padrão de cookie de envio duplo, nonces CSRF ou verificações de cabeçalho de solicitação de origem.	✓	✓	✓	352
13.2.4	[EXCLUÍDO, DUPLICADO DE 11.1.4]				
13.2.5	Verifique se os serviços REST verificam explicitamente se o Content-Type de input é o esperado, como application/xml ou application/json.		✓	✓	436
13.2.6	Verifique se os cabeçalhos e a carga da mensagem são confiáveis e não foram modificados em trânsito. Exigir criptografia forte para transporte (somente TLS) pode ser suficiente em muitos casos, pois fornece proteção de confidencialidade e integridade. As assinaturas digitais por mensagem podem fornecer garantia adicional além das proteções de transporte para aplicações de alta segurança, mas trazem consigo complexidade e riscos adicionais para comparar com os benefícios.		✓	✓	345

V13.3 Serviço Web SOAP

#	Descrição	L1	L2	L3	CWE
13.3.1	Verifique se a validação do esquema XSD ocorre para garantir um documento XML formado corretamente, seguido pela validação de cada campo de input antes de qualquer processamento desses dados.	✓	✓	✓	20
13.3.2	Verifique se a carga útil da mensagem está assinada usando WS-Security para garantir o transporte confiável entre o cliente e o serviço.		✓	✓	345

Observação: devido a problemas com ataques XXE contra DTDs, a validação de DTD não deve ser usada e a avaliação de DTD de estrutura desativada de acordo com os requisitos estabelecidos na configuração V14.

V13.4 GraphQL

#	Descrição	L1	L2	L3	CWE
13.4.1	Verifique se uma lista de permissões de consulta ou uma combinação de limitação de profundidade e limitação de quantidade é usada para impedir o GraphQL ou a negação de serviço (DoS) da expressão da camada de dados como resultado de consultas aninhadas caras. Para cenários mais avançados, a análise de custo de consulta deve ser usada.		✓	✓	770
13.4.2	Verifique se GraphQL ou outra lógica de autorização da camada de dados deve ser implementada na camada de lógica de negócios em vez da camada GraphQL.		✓	✓	285

Referências

Para mais informações, consulte também:

- OWASP Serverless Top 10

- OWASP Serverless Project
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- OWASP Cross-Site Request Forgery cheat sheet
- OWASP XML External Entity Prevention Cheat Sheet - General Guidance
- JSON Web Tokens (and Signing)
- REST Security Cheat Sheet
- JSON Schema
- XML DTD Entity Attacks
- Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages

V14 Configuração

Objetivo de controle

Certifique-se de que uma aplicação verificado tenha:

- Um ambiente de construção seguro, repetível e automatizável.
- Biblioteca reforçada de terceiros, dependência e gestão de configuração de forma que componentes desatualizados ou inseguros não sejam incluídos pela aplicação.

A configuração da aplicação pronta para uso deve ser segura para estar na Internet, significando uma configuração segura pronta para uso.

V14.1 Construir e Implantar

Os pipelines de compilação são a base para a segurança repetível - toda vez que algo inseguro é descoberto, ele pode ser resolvido no código-fonte, scripts de compilação ou implantação e testado automaticamente. Incentivamos fortemente o uso de pipelines de compilação com segurança automática e verificações de dependência que avisam ou interrompem a compilação para evitar que problemas de segurança conhecidos sejam implantados na produção. Etapas manuais executadas de forma irregular levam diretamente a erros de segurança evitáveis.

À medida que o setor se move para um modelo DevSecOps, é importante garantir a disponibilidade e a integridade contínuas da implantação e da configuração para atingir um estado "conhecido como bom". No passado, se um sistema fosse hackeado, levaria de dias a meses para provar que nenhuma outra intrusão havia ocorrido. Hoje, com o advento da infraestrutura definida por software, implantações rápidas de A/B com tempo de inatividade zero e compilações automatizadas em contêineres, é possível criar, fortalecer e implantar de forma automática e contínua uma substituição "boa" para qualquer sistema comprometido.

Se os modelos tradicionais ainda estiverem em vigor, devem ser tomadas medidas manuais para fortalecer e fazer backup dessa configuração para permitir que os sistemas comprometidos sejam rapidamente substituídos por sistemas não comprometidos de alta integridade em tempo hábil.

A conformidade com esta seção requer um sistema de compilação automatizado e acesso a scripts de compilação e implantação.

#	Descrição	L1	L2	L3	CWE
14.1.1	Verifique se os processos de criação e implantação da aplicação são executados de maneira segura e repetível, como automação de CI/CD, gestão de configuração automatizada e scripts de implantação automatizada.		✓	✓	
14.1.2	Verifique se os sinalizadores do compilador estão configurados para habilitar todas as proteções e avisos de buffer overflow disponíveis, incluindo randomização de pilha, prevenção de execução de dados e para interromper a compilação se um ponteiro inseguro, memória, string de formato, número inteiro ou operações de string forem encontrados.		✓	✓	120
14.1.3	Verifique se a configuração do servidor está protegida de acordo com as recomendações do servidor de aplicações e estruturas em uso.		✓	✓	16
14.1.4	Verifique se a aplicação, a configuração e todas as dependências podem ser reimplantados usando scripts de implantação automatizados, criados a partir de um runbook documentado e testado num tempo razoável ou restaurados de backups em tempo hábil.		✓	✓	
14.1.5	Verifique se os administradores autorizados podem verificar a integridade de todas as configurações relevantes para a segurança para detectar adulterações.			✓	

V14.2 Dependência

O gestão de dependência é crítico para a operação segura de qualquer aplicação de qualquer tipo. A falha em manter-se atualizado com dependências desatualizadas ou inseguras é a causa raiz dos maiores e mais caros ataques até hoje.

Nota: No Nível 1, a conformidade com 14.2.1 refere-se a observações ou detecções do lado do cliente e outras bibliotecas e componentes, em vez da análise de código estático ou análise de dependência mais precisa em tempo de compilação. Essas técnicas mais precisas podem ser descobertas por entrevistas, conforme necessário.

#	Descrição	L1	L2	L3	CWE
14.2.1	Verifique se todos os componentes estão atualizados, de preferência usando um verificador de dependência durante o tempo de construção ou compilação. (C2)	✓	✓	✓	1026
14.2.2	Verifique se todos os recursos, documentação, aplicações de amostra e configurações desnecessários foram removidos.	✓	✓	✓	1002
14.2.3	Verifique se os ativos da aplicação, como bibliotecas JavaScript, CSS ou fontes da Web, são hospedados externamente numa rede de entrega de conteúdo (CDN) ou provedor externo, a integridade do subrecurso (SRI) é usada para validar a integridade do ativo.	✓	✓	✓	829
14.2.4	Verifique se os componentes de terceiros vêm de repositórios predefinidos, confiáveis e mantidos continuamente. (C2)		✓	✓	829
14.2.5	Verifique se uma lista de materiais de software (SBOM) é mantida para todas as bibliotecas de terceiros em uso. (C2)		✓	✓	
14.2.6	Verifique se a superfície de ataque é reduzida por sandbox ou encapsulamento de bibliotecas de terceiros para expor apenas o comportamento necessário na aplicação. (C2)		✓	✓	265

V14.3 Divulgação de segurança não intencional

As configurações para produção devem ser fortalecidas para proteger contra-ataques comuns, como consoles de depuração, aumentar o nível de ataques Cross-site Scripting (XSS) e Remote File Inclusion (RFI) e eliminar "vulnerabilidades" triviais de descoberta de informações que são as indesejadas marcas registradas de muitos relatórios de teste de penetração. Muitos desses problemas são raramente classificados como um risco significativo, mas estão encadeados com outras vulnerabilidades. Se esses problemas não estiverem presentes por padrão, ele eleva a fasquia antes que a maioria dos ataques seja bem-sucedida.

#	Descrição	L1	L2	L3	CWE
14.3.1	[EXCLUÍDO, DUPLICADO DE 7.4.1]				
14.3.2	Verifique se os modos de depuração da estrutura da aplicação ou servidor da Web estão desativados na produção para eliminar recursos de depuração, consoles de desenvolvedor e divulgações de segurança não intencionais.	✓	✓	✓	497
14.3.3	Verifique se os cabeçalhos HTTP ou qualquer parte da resposta HTTP não expõe informações detalhadas sobre a versão dos componentes do sistema.	✓	✓	✓	200

V14.4 Cabeçalhos de segurança HTTP

#	Descrição	L1	L2	L3	CWE
14.4.1	Verifique se cada resposta HTTP contém um cabeçalho Content-Type. Especifique também um conjunto de caracteres seguro (por exemplo, UTF-8, ISO-8859-1) se os tipos de conteúdo forem text/*, /+xml e application/xml. O conteúdo deve corresponder ao cabeçalho Content-Type fornecido.	✓	✓	✓	173
14.4.2	Verifique se todas as respostas da API contêm um anexo Content-Disposition:; cabeçalho filename="api.json" (ou outro nome de arquivo apropriado para o tipo de conteúdo).	✓	✓	✓	116
14.4.3	Verifique se um cabeçalho de resposta da política de segurança de conteúdo (CSP) está em vigor para ajudar a atenuar o impacto de ataques XSS, como HTML, DOM, JSON e vulnerabilidades de injeção de JavaScript.	✓	✓	✓	1021
14.4.4	Verifique se todas as respostas contêm um cabeçalho X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Verifique se um cabeçalho Strict-Transport-Security está incluído em todas as respostas e para todos os subdomínios, como Strict-Transport-Security: max-age=15724800; includeSubdomains.	✓	✓	✓	523
14.4.6	Verifique se um cabeçalho Referrer-Policy adequado está incluído para evitar a exposição de informações confidenciais na URL por meio do cabeçalho Referer para partes não confiáveis.	✓	✓	✓	116
14.4.7	Verifique se o conteúdo de uma aplicação da Web não pode ser incorporado num site de terceiros por padrão e se a incorporação dos recursos exatos só é permitida quando necessária usando a política de segurança de conteúdo adequada: resposta de ancestrais de quadro e opções de X-Frame cabeçalhos.	✓	✓	✓	1021

V14.5 Validação de cabeçalho de solicitação HTTP

#	Descrição	L1	L2	L3	CWE
14.5.1	Verifique se o servidor de aplicações aceita apenas os métodos HTTP em uso pela aplicação/API, incluindo OPÇÕES pré-voo e logs/alertas em quaisquer solicitações que não sejam válidas para o contexto da aplicação.	✓	✓	✓	749
14.5.2	Verifique se o cabeçalho Origin fornecido não é usado para autenticação ou decisões de controle de acesso, pois o cabeçalho Origin pode ser facilmente alterado por um invasor.	✓	✓	✓	346
14.5.3	Verifique se o cabeçalho Access-Control-Allow-Origin de compartilhamento de recursos de origem cruzada (CORS) usa uma lista de permissão estrita de domínios e subdomínios confiáveis para correspondência e não oferece suporte à origem "nula".	✓	✓	✓	346
14.5.4	Verifique se os cabeçalhos HTTP adicionados por um proxy confiável ou dispositivos SSO, como um token de portador, são autenticados pela aplicação.		✓	✓	306

Referências

Para mais informações, consulte também:

- OWASP Web Security Testing Guide 4.1: Testing for HTTP Verb Tampering
- Adicionar disposição de conteúdo às respostas da API ajuda a evitar muitos ataques baseados em mal-entendidos no tipo MIME entre cliente e servidor, e a opção "nome do arquivo" especificamente ajuda a evitar Reflected File Download attacks.
- Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- OWASP Web Security Testing Guide 4.1: Configuration and Deployment Management Testing
- Sandboxing third party components

Apêndice A: Glossário

- **Address Space Layout Randomization** (ASLR) - Uma técnica para dificultar a exploração de bugs de corrupção de memória.
- **Lista de permissões** - Uma lista de dados ou operações permitidas, por exemplo, uma lista de caracteres com permissão para executar a validação de input.
- **Segurança de aplicações** - a segurança em nível de aplicação se concentra na análise de componentes que compõem a camada de aplicação do Modelo de referência de interconexão de sistemas abertos (modelo OSI), em vez de se concentrar, por exemplo, no sistema operacional subjacente ou nas redes conectadas.
- **Application Security Verification** - A avaliação técnica de uma aplicação contra o OWASP ASVS.
- **Relatório de verificação de segurança da aplicação** - Um relatório que documenta os resultados gerais e a análise de suporte produzida pelo verificador para uma aplicação específico.
- **Autenticação** - A verificação da identidade reivindicada de um usuário da aplicação.
- **Verificação automatizada** - O uso de ferramentas automatizadas (ferramentas de análise dinâmica, ferramentas de análise estática ou ambas) que usam assinaturas de vulnerabilidade para encontrar problemas.
- **teste black box** - É um método de teste de software que examina a funcionalidade de uma aplicação sem examinar as suas estruturas ou funcionamento interno.
- **Componente** - uma unidade de código independente, com disco associado e interfaces de rede que se comunica com outros componentes.
- **Cross-Site Scripting** (XSS) - Uma vulnerabilidade de segurança normalmente encontrada ema aplicaçãos da Web que permite a injeção de scripts do lado do cliente no conteúdo.
- **Módulo criptográfico** - Hardware, software e/ou firmware que implementa algoritmos criptográficos e/ou gera chaves criptográficas.
- **Common Weakness Enumeration** (CWE) - Uma lista desenvolvida pela comunidade de vulnerabilidades comuns de segurança de software. Ele serve como uma linguagem comum, uma medida para ferramentas de segurança de software e como uma linha de base para esforços de identificação, mitigação e prevenção de fraquezas.
- **Verificação de projeto** - A avaliação técnica da arquitetura de segurança de uma aplicação.
- **Dynamic Application Security Testing** (DAST) - As tecnologias são projetadas para detectar condições indicativas de uma vulnerabilidade de segurança numa aplicação em seu estado de execução.
- **Verificação Dinâmica** - O uso de ferramentas automatizadas que usam assinaturas de vulnerabilidade para encontrar problemas durante a execução de uma aplicação.
- **Fast IDentity Online** (FIDO) - Um conjunto de padrões de autenticação que permite o uso de vários métodos de autenticação diferentes, incluindo biometria, Trusted Platform Modules (TPMs), tokens de segurança USB, etc.
- **Globally Unique Identifier** (GUID) - um número de referência exclusivo usado como um identificador no software.
- **Hyper Text Transfer Protocol** (HTTPS) - Um protocolo de aplicação para sistemas de informação de hipermídia distribuídos e colaborativos. É a base da comunicação de dados para a World Wide Web.
- **Chaves codificadas** - Chaves criptográficas armazenadas no sistema de arquivos, seja em código, comentários ou arquivos.
- **Módulo de Segurança de Hardware** (HSM) - Componente de hardware capaz de armazenar chaves criptográficas e outros segredos de forma protegida.

- **Hibernate Query Language** (HQL) - Uma linguagem de consulta semelhante em aparência ao SQL usado pela biblioteca Hibernate ORM.
- **Validação de input** - A canonização e validação de input de usuário não confiável.
- **Código malicioso** - Código introduzido numa aplicação durante o seu desenvolvimento sem o conhecimento do proprietário da aplicação, que contorna a política de segurança pretendida da aplicação. Não é o mesmo que malware, como um vírus ou worm!
- **Malware** - Código executável introduzido numa aplicação durante o tempo de execução sem o conhecimento do usuário ou administrador da aplicação.
- **Open Web Application Security Project** (OWASP) - O Open Web Application Security Project (OWASP) é uma comunidade mundial gratuita e aberta focada em melhorar a segurança do software de aplicações. A nossa missão é tornar a segurança de aplicações "visível", para que pessoas e organizações possam tomar decisões informadas sobre os riscos de segurança de aplicações. Veja: https://www.owasp.org/
- **One-time Password** (OTP) - Uma senha gerada exclusivamente para ser usada numa única ocasião.
- **Mapeamento relacional de objeto** (ORM) - Um sistema usado para permitir que um banco de dados relacional/baseado em tabela seja referenciado e consultado dentro de um programa aplicação usando um modelo de objeto compatível com a aplicação.
- **Função de derivação de chave baseada em senha 2** (PBKDF2) - Um algoritmo unidirecional especial usado para criar uma chave criptográfica forte a partir de um texto de input (como uma senha) e um valor de sal aleatório adicional e, assim, pode ser usado dificulte quebrar uma senha offline se o valor resultante for armazenado em vez da senha original.
- **Informações de identificação pessoal** (PII) - são informações que podem ser usadas sozinhas ou com outras informações para identificar, contatar ou localizar uma única pessoa ou para identificar um indivíduo no contexto.
- **Executável independente de posição** (PIE) - Um corpo de código de máquina que, sendo colocado em algum lugar na memória primária, é executado adequadamente independentemente do seu endereço absoluto.
- **Public Key Infrastructure** (PKI) - Um arranjo que vincula as chaves públicas às respectivas identidades das entidades. A vinculação é estabelecida por meio de um processo de registro e emissão de certificados em e por uma autoridade de certificação (CA).
- **Rede telefônica pública comutada** (PSTN) - A rede telefônica tradicional, incluindo telefones fixos e celulares.
- **Relying Party** (RP) - Geralmente, uma aplicação que depende de um usuário autenticado num provedor de autenticação separado. a aplicação conta com algum tipo de token ou conjunto de asserções assinadas fornecidas por esse provedor de autenticação para confiar que o usuário é quem diz ser.
- **Teste de segurança de aplicações estáticos** (SAST) - Um conjunto de tecnologias projetadas para analisar o código-fonte da aplicação, código de bytes e binários para condições de codificação e design que são indicativas de vulnerabilidades de segurança. As soluções SAST analisam uma aplicação de “dentro para fora” num estado sem execução.
- **Ciclo de vida de desenvolvimento de software** (SDLC) - O processo passo a passo pelo qual o software é desenvolvido desde os requisitos iniciais até a implantação e manutenção.
- **Arquitetura de segurança** - Uma abstração do design de uma aplicação que identifica e descreve onde e como os controles de segurança são usados e também identifica e descreve a localização e a confidencialidade dos dados do usuário e da aplicação.
- **Configuração de segurança** - A configuração de tempo de execução de uma aplicação que afeta como os controles de segurança são usados.

- **Controle de segurança** - Uma função ou componente que executa uma verificação de segurança (por exemplo, uma verificação de controle de acesso) ou quando chamado resulta num efeito de segurança (por exemplo, gerando um registro de auditoria).
- **Server-side Request Forgery** (SSRF) - Um ataque que abusa da funcionalidade do servidor para ler ou atualizar recursos internos, fornecendo ou modificando uma URL que o código em execução no servidor irá ler ou enviar dados.
- **Single Sign-on Authentication** (SSO) - Isso ocorre quando um usuário faz login numa aplicação e é automaticamente conectado a outras aplicações sem precisar se autenticar novamente. Por exemplo, ao fazer login no Google, ao acessar outros serviços do Google, como YouTube, Google Docs e Gmail, você será automaticamente conectado.
- **SQL Injection** (SQLi) - Uma técnica de injeção de código usada para atacar aplicações baseados em dados, nos quais instruções SQL maliciosas são inseridas num ponto de input.
- **SVG** - Gráficos vetoriais escaláveis
- **OTP baseado em tempo** - Um método de geração de um OTP onde a hora atual atua como parte do algoritmo para gerar a senha.
- **Modelagem de ameaças** - Uma técnica que consiste no desenvolvimento de arquiteturas de segurança cada vez mais refinadas para identificar agentes de ameaças, zonas de segurança, controles de segurança e importantes ativos técnicos e de negócios.
- **Transport Layer Security** (TLS) - protocolos criptográficos que fornecem segurança de comunicação numa conexão de rede
- **Módulo de plataforma confiável** (TPM) - Um tipo de HSM que é geralmente conectado a um componente de hardware maior, como uma placa-mãe, e atua como a "raiz de confiança" desse sistema.
- **Autenticação de dois fatores** (2FA) - Adiciona um segundo nível de autenticação a um login de conta.
- **Universal 2nd Factor** (U2F) - Um dos padrões criados pela FIDO especificamente para permitir que uma chave de segurança USB ou NFC seja usada como um 2.º fator de autenticação.
- **Fragmentos de URL/URL/URL** - Um identificador de recurso uniforme é uma cadeia de caracteres usada para identificar um nome ou um recurso da web. Um localizador uniforme de recursos é geralmente usado como referência a um recurso.
- **Verificador** - A pessoa ou equipe que está revisando uma aplicação em relação aos requisitos OWASP ASVS.
- **O que você vê é o que obtém** (WYSIWYG) - Um tipo de editor de conteúdo avançado que mostra como o conteúdo realmente ficará quando renderizado, em vez de mostrar a codificação usada para controlar a renderização.
- **Certificado X.509** - Um certificado X.509 é um certificado digital que usa o padrão internacional amplamente aceito de infraestrutura de chave pública (PKI) X.509 para verificar se uma chave pública pertence à identidade do usuário, computador ou serviço contida no certificado.
- **XML eXternal Entity** (XXE) - Um tipo de entidade XML que pode acessar conteúdo local ou remoto por meio de um identificador de sistema declarado. Isso pode carregar vários ataques de injeção.

Apêndice B: Referências

Os seguintes projetos OWASP provavelmente serão úteis para usuários/adotantes deste padrão:

Projetos principais do OWASP

1. OWASP Top 10 Project: https://owasp.org/www-project-top-ten/
2. OWASP Web Security Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Security Knowledge Framework: https://owasp.org/www-project-security-knowledge-framework/
5. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/

Projeto OWASP Cheat Sheet Series

Este projeto possui várias sugestões que serão relevantes para diferentes tópicos no ASVS.

Existe um mapeamento para o ASVS que pode ser encontrado aqui:
https://cheatsheetseries.owasp.org/cheatsheets/IndexASVS.html

Projetos Relacionados à Segurança Mobile

1. OWASP Mobile Security Project: https://owasp.org/www-project-mobile-security/
2. OWASP Mobile Top 10 Risks: https://owasp.org/www-project-mobile-top-10/
3. OWASP Mobile Security Testing Guide and Mobile Application Security Verification Standard: https://owasp.org/www-project-mobile-security-testing-guide/

Projetos relacionados à Internet of Things OWASP

1. OWASP Internet of Things Project: https://owasp.org/www-project-internet-of-things/

Projetos OWASP Serverless

1. OWASP Serverless Project: https://owasp.org/www-project-serverless-top-10/

Outras

Da mesma forma, os seguintes sites provavelmente serão úteis para usuários/adotantes deste padrão.

1. SecLists Github: https://github.com/danielmiessler/SecLists
2. MITRE Common Weakness Enumeration: https://cwe.mitre.org/
3. PCI Security Standards Council: https://www.pcisecuritystandards.org
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf

Apêndice C: Requisitos de verificação da Internet of Things

Este capítulo estava originalmente no ramo principal, mas com o trabalho que a equipe OWASP IoT tem feito, não faz sentido manter dois tópicos diferentes sobre o assunto. Para a versão 4.0, estamos movendo isso para o Apêndice e instamos todos que precisam disso a usar o OWASP IoT Project principal.

Objetivo de controle

Os dispositivos integrados/IoT devem:

- Tenha o mesmo nível de controles de segurança no dispositivo encontrado no servidor, aplicando controles de segurança num ambiente confiável.
- Os dados confidenciais armazenados no dispositivo devem ser feitos de maneira segura, usando armazenamento com suporte de hardware, como elementos seguros.
- Todos os dados confidenciais transmitidos do dispositivo devem utilizar a segurança da camada de transporte.

Requisitos de verificação de segurança

#	Descrição	L1	L2	L3	Desde
C.1	Verifique se as interfaces de depuração da camada de aplicação, como USB, UART e outras variantes seriais, estão desativadas ou protegidas por uma senha complexa.	✓	✓	✓	4.0
C.2	Verifique se as chaves e certificados criptográficos são exclusivos para cada dispositivo individual.	✓	✓	✓	4.0
C.3	Verifique se os controles de proteção de memória, como ASLR e DEP, estão ativados pelo sistema operacional integrado/IoT, se aplicável.	✓	✓	✓	4.0
C.4	Verifique se as interfaces de depuração no chip, como JTAG ou SWD, estão desabilitadas ou se o mecanismo de proteção disponível está habilitado e configurado adequadamente.	✓	✓	✓	4.0
C.5	Verifique se a execução confiável está implementada e habilitada, se disponível no dispositivo SoC ou CPU.	✓	✓	✓	4.0
C.6	Verifique se dados confidenciais, chaves privadas e certificados estão armazenados com segurança num Elemento Seguro, TPM, TEE (Ambiente de Execução Confiável) ou protegidos por criptografia forte.	✓	✓	✓	4.0
C.7	Verifique se as aplicações de firmware protegem os dados em trânsito usando a segurança da camada de transporte.	✓	✓	✓	4.0
C.8	Verifique se as aplicações de firmware validam a assinatura digital das conexões do servidor.	✓	✓	✓	4.0
C.9	Verifique se as comunicações sem fio são mutuamente autenticadas.	✓	✓	✓	4.0
C.10	Verifique se as comunicações sem fio são enviadas por um canal criptografado.	✓	✓	✓	4.0
C.11	Verifique se qualquer uso de funções C proibidas é substituído pelas funções equivalentes seguras apropriadas.	✓	✓	✓	4.0
C.12	Verifique se cada firmware mantém uma lista de materiais de software catalogando componentes de terceiros, versões e vulnerabilidades publicadas.	✓	✓	✓	4.0

#	Descrição	L1	L2	L3	Desde
C.13	Verifique se todos os códigos, incluindo binários, bibliotecas e estruturas de terceiros, são revisados quanto a credenciais codificadas (backdoors).	✓	✓	✓	4.0
C.14	Verifique se os componentes da aplicação e do firmware não são suscetíveis à injeção de comando do sistema operacional, invocando wrappers de comando shell, scripts ou se os controles de segurança impedem a injeção de comando do sistema operacional.	✓	✓	✓	4.0
C.15	Verifique se as aplicações de firmware fixam a assinatura digital em servidores confiáveis.		✓	✓	4.0
C.16	Verifique a presença de recursos de resistência e/ou detecção de violação.		✓	✓	4.0
C.17	Verifique se todas as tecnologias de proteção de propriedade intelectual disponíveis fornecidas pelo fabricante do chip estão ativadas.		✓	✓	4.0
C.18	Verifique se os controles de segurança estão em vigor para impedir a engenharia reversa do firmware (por exemplo, remoção de símbolos de depuração detalhados).		✓	✓	4.0
C.19	Verifique se o dispositivo valida a assinatura da imagem de inicialização antes de carregar.		✓	✓	4.0
C.20	Verifique se o processo de atualização do firmware não é vulnerável a ataques de tempo de verificação versus tempo de uso.		✓	✓	4.0
C.21	Verifique se o dispositivo usa assinatura de código e valida os arquivos de atualização do firmware antes da instalação.		✓	✓	4.0
C.22	Verifique se o dispositivo não pode ser rebaixado para versões antigas (anti-reversão) de firmware válido.		✓	✓	4.0
C.23	Verifique o uso do gerador de números pseudoaleatórios criptograficamente seguro no dispositivo incorporado (por exemplo, usando geradores de números aleatórios fornecidos por chip).		✓	✓	4.0
C.24	Verifique se o firmware pode executar atualizações automáticas de firmware num cronograma predefinido.		✓	✓	4.0
C.25	Verifique se o dispositivo limpa o firmware e os dados confidenciais após a detecção de adulteração ou recebimento de mensagem inválida.			✓	4.0
C.26	Verifique se apenas os microcontroladores que suportam a desativação de interfaces de depuração (por exemplo, JTAG, SWD) são usados.			✓	4.0
C.27	Verifique se apenas os microcontroladores que fornecem proteção substancial contra-ataques de decapagem e de canal lateral são usados.			✓	4.0
C.28	Verifique se os traços sensíveis não estão expostos às camadas externas da placa de circuito impresso.			✓	4.0
C.29	Verifique se a comunicação entre chips está criptografada (por exemplo, comunicação da placa principal para a placa filha).			✓	4.0
C.30	Verifique se o dispositivo usa assinatura de código e valida o código antes da execução.			✓	4.0

#	Descrição	L1	L2	L3	Desde
C.31	Verifique se as informações confidenciais mantidas na memória são substituídas por zeros assim que não são mais necessárias.			✓	4.0
C.32	Verifique se as aplicações de firmware utilizam contêineres de kernel para isolamento entre aplicações.			✓	4.0
C.33	Verifique se os sinalizadores de compilador seguro, como -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap, estão configurados para compilações de firmware.			✓	4.0
C.34	Verifique se os microcontroladores estão configurados com proteção de código (se aplicável).			✓	4.0

Referências

Para mais informações, consulte também:

- OWASP Internet of Things Top 10
- Projeto OWASP Embedded Application Security
- Projeto Internet of Things OWASP
- Trudy TCP Proxy Tool