#	Descrição	L1	L2	L3	CWE	NIST §
1.1.1	Verifique o uso de um ciclo de vida seguro de desenvolvimento de software que aborda a segurança em todas as fases de desenvolvimento. (C1)		✓	✓		
1.1.2	Verifique o uso de modelagem de ameaças para cada mudança de design ou planejamento de sprint para identificar ameaças, planejar contramedidas, facilitar respostas de risco apropriadas 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, eu deveria ser capaz de visualizar e editar meu perfil. Eu não deveria ser capaz de visualizar ou editar o perfil de outra pessoa"		✓	✓	1110	
1.1.4	Verifique a documentação e a justificativa de todos os limites de confiança do aplicativo, componentes e fluxos de dados significativos.		✓	✓	1059	
1.1.5	Verifique a definição e a análise de segurança da arquitetura de alto nível do aplicativo e de todos os serviços remotos conectados. (C1)		✓	✓	1059	
1.1.6	Verifique a implementação de controles de segurança centralizados, simples (econômicos de design), controlados, seguros e reutilizáveis para evitar controles de segurança 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	
1.2.1	Verifique o uso de contas únicas ou especiais de sistema operacional de baixo privilégio para todos os componentes, serviços e servidores do aplicativo. (C3)		✓	✓	250	
1.2.2	Verifique se as comunicações entre componentes do aplicativo, incluindo APIs, middleware e camadas de dados, são autenticadas. Os componentes devem ter os privilégios menos necessários. (C3)		✓	✓	306	
1.2.3	Verifique se o aplicativo usa um único mecanismo de autenticação controlado que é conhecido por ser seguro, pode ser estendido para incluir autenticação forte e tem registro e monitoramento suficientes para detectar abusos ou violações de contas.		✓	✓	306	
1.2.4	Verifique se todas as vias de autenticação e APIs de gerenciamento de identidade implementam uma força consistente de controle de segurança de autenticação, de tal forma que não há alternativas mais fracas pelo risco do aplicativo.		✓	✓	306	
1.4.1	Verifique se pontos de execução confiáveis, como gateways de controle de acesso, servidores e funções sem servidor, aplicam controles de acesso. Nunca imponha controles de acesso ao cliente.		✓	✓	602	
1.4.2	Verifique se a solução de controle de acesso escolhida é flexível o suficiente para atender às necessidades do aplicativo.		✓	✓	284	
1.4.3	Verifique a aplicação do princípio do menor privilégio em funções, arquivos de dados, URLs, controladores, serviços e outros recursos. Isso implica proteção contra falsificação e elevação de privilégios.		✓	✓	272	
1.4.4	Verifique se o aplicativo usa um mecanismo de controle de acesso único e bem controlado para acessar dados e recursos protegidos. Todas as solicitações devem passar por este mecanismo único para evitar copiar e colar ou caminhos alternativos inseguros. (C7)		✓	✓	284	
1.4.5	Verifique se o atributo ou o controle de acesso baseado em recursos são usados pelo qual o código verifica a autorização do usuário para um item de recurso/dados, em vez de apenas sua função. As permissões ainda devem ser alocadas usando funções. (C7)		✓	✓	275	
1.5.1	Verifique se os requisitos de entrada e saída definem claramente como lidar e processar dados com base em tipo, conteúdo e leis, regulamentos e outras conformidades de políticas aplicáveis.		✓	✓	1029	
1.5.2	Verifique se a serialização não é usada quando se comunica com clientes não confiáveis. Se isso não for possível, certifique-se de que controles de integridade adequados (e possivelmente criptografia se dados confidenciais forem enviados) sejam aplicados para evitar ataques de desercionalização, incluindo injeção de objeto.		✓	✓	502	
1.5.3	Verifique se a validação de entrada é aplicada em uma camada de serviço confiável. (C5)		✓	✓	602	
1.5.4	Verifique se a codificação de saída ocorre perto ou pelo intérprete para o qual se destina. (C4)		✓	✓	116	
1.6.1	Verifique se existe uma política explícita de gerenciamento de chaves criptográficas e que um ciclo de vida de chave criptográfica segue um padrão de gerenciamento chave, como o NIST SP 800-57.		✓	✓	320	
1.6.2	Verifique se os consumidores de serviços criptográficos protegem material-chave e outros segredos usando cofres-chave 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 reripriptando dados confidenciais.		✓	✓	320	
1.6.4	Verifique se chaves simétricas, senhas ou segredos de API gerados ou compartilhados com clientes são usados apenas na proteção de segredos de baixo risco, como criptografar armazenamento local ou usos efêmeros temporários, como ofuscação de parâmetros. Compartilhar segredos com clientes é equivalente a texto claro e arquitetônicamente deve ser tratado como tal.		✓	✓	320	
1.7.1	Verifique se um formato e abordagem de registro comum é usado em todo o sistema. (C9)		✓	✓	1009	
1.7.2	Verifique se os registros são transmitidos com segurança para um sistema preferencialmente remoto para análise, detecção, alerta e escalada. (C9)		✓	✓		
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 possuem um conjunto associado de requisitos de proteção, como requisitos de criptografia, requisitos de integridade, retenção, privacidade e outros requisitos de confidencialidade, e que estes são aplicados na arquitetura.		✓	✓		
1.9.1	Verifique se o aplicativo criptografa as comunicações entre os componentes, especialmente quando esses componentes estão em diferentes contêineres, sistemas, sites ou provedores de nuvem. (C3)		✓	✓	319	
1.9.2	Verifique se os componentes do aplicativo verificam a autenticidade de cada lado em um link de comunicação para evitar ataques de pessoas no meio. Por exemplo, os componentes do aplicativo devem validar certificados e cadeias TLS.		✓	✓	295	
1.10.1	Verifique se um sistema de controle de código-fonte está em uso, com procedimentos para garantir que os check-ins estejam acompanhados de problemas ou alterações de bilhetes. O sistema de controle de código-fonte deve ter controle de acesso e usuários identificáveis para permitir a rastreabilidade de quaisquer alterações.		✓	✓	284	
1.11.1	Verifique a definição e documentação de todos os componentes do aplicativo em termos das funções de negócios ou de segurança que eles fornecem.		✓	✓	1059	
1.11.2	Verifique se todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gerenciamento de sessões e controle de acesso, não compartilham estado dessincronizado.		✓	✓	362	
1.11.3	Verifique se todos os fluxos de lógica de negócios de alto valor, incluindo autenticação, gerenciamento de sessão e controle de acesso, são seguros e resistentes a condições de corrida de tempo de verificação e tempo de uso.			✓	367	
1.12.1	Verifique se os arquivos carregados pelo usuário estão armazenados fora da raiz da web		✓	✓	552	
1.12.2	Verifique se os arquivos carregados pelo usuário - se necessário para serem exibidos ou baixados do aplicativo - são servidos por downloads de fluxo de octeto ou por um domínio não relacionado, como um balde de armazenamento de arquivos em nuvem. Implemente uma política de segurança de conteúdo adequada para reduzir o risco de vetores XSS ou outros ataques do arquivo carregado.		✓	✓	646	

1.14.1	Verifique a segregação de componentes de diferentes níveis de confiança através 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 a implantação de binários em dispositivos não confiáveis faz uso de assinaturas binárias, conexões confiáveis e pontos finais verificados.		✓	✓	494	
1.14.3	Verifique se o pipeline de construção alerta para componentes desatualizados ou inseguros e tome as medidas apropriadas.		✓	✓	1104	
1.14.4	Verifique se o pipeline de compilação contém uma etapa de compilação para construir e verificar automaticamente a implantação segura do aplicativo, particularmente se a infraestrutura do aplicativo for definida por software, como scripts de compilação de ambiente em nuvem.		✓	✓		
1.14.5	Verifique se as implantações do aplicativo são adequadamente sandbox, contêiner e/ou isolam no nível da rede para atrasar e impedir que os invasores ataquem outras aplicações, especialmente quando eles estão realizando ações sensíveis ou perigosas, como a deserialização. (C5)		✓	✓	265	
1.14.6	Verifique se o aplicativo não usa tecnologias não suportadas, inseguras ou preteridas do lado do cliente, como plugins NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL ou javatas do lado do cliente.		✓	✓	477	
2.2.1	Verifique se os controles anti-automação são eficazes na mitigação de testes de credenciais violados, força bruta e ataques de bloqueio de contas. Tais controles incluem o bloqueio das senhas violadas mais comuns, bloqueios suaves, limitação de taxas, CAPTCHA, atrasos cada vez maiores entre tentativas, restrições de endereço IP ou restrições baseadas em riscos, como localização, primeiro login em um dispositivo, tentativas recentes de desbloquear a conta	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2
2.2.2	Verifique se o uso de autenticadores fracos (como SMS e e-mail) está limitado à verificação secundária e aprovação de transações e não como um substituto para métodos de autenticação mais seguros. Verifique se métodos mais fortes são oferecidos antes de métodos fracos, os usuários estão cientes dos riscos ou que 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 para detalhes de autenticação, como redefinições de credenciais, alterações de e-mail ou endereço, login em locais desconhecidos ou arriscados. O uso de notificações push - em vez de SMS ou e-mail - é preferido, mas na ausência de notificações push, SMS ou e-mail é aceitável desde que nenhuma informação sensível seja divulgada na notificação.	✓	✓	✓	620	
2.2.4	Verifique a resistência à representação contra phishing, como o uso de autenticação de vários fatores, dispositivos criptográficos com intenção (como chaves conectadas com um empurrão para autenticar) ou em níveis AAL mais altos, certificados do lado do cliente.			✓	308	5.2.5
2.2.5	Verifique se quando um provedor de serviços de credencial (CSP) e o aplicativo que verifica a autenticação estão separados, o TLS autenticado mutuamente está em vigor entre os dois pontos finais.			✓	319	5.2.6
2.2.6	Verifique a resistência ao replay através do uso obrigatório de dispositivos OTP, autenticadores criptográficos ou códigos de pesquisa.			✓	308	5.2.8
2.2.7	Verifique a intenção de autenticar exigindo a entrada de um token OTP ou ação iniciada pelo usuário, como um botão pressionar uma tecla de hardware FIDO.			✓	308	5.2.9
2.3.1	Verifique as senhas iniciais ou códigos de ativação gerados pelo sistema DEVE ser gerado aleatoriamente, deve ter pelo menos 6 caracteres e PODE conter letras e números e expirar 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
2.3.2	Verifique se a inscrição e o uso de dispositivos de autenticação fornecidos por assinantes são suportados, como um token U2F ou FIDO.		✓	✓	308	6.1.3
2.3.3	Verifique se as instruções de renovação são enviadas com tempo suficiente para renovar autenticadores vinculados ao tempo.		✓	✓	287	6.1.4
2.4.1	Verifique se as senhas são armazenadas em um formulário resistente a ataques offline. As senhas devem ser salgadas e hashed usando uma função de derivação de chave unidiretiva aprovada ou hashing de senha. As funções de derivação e hashing de senhas da chave tomam uma senha, um sal e um fator de custo como entradas 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 será escolhido arbitrariamente para minimizar colisões de valor de sal entre hashes armazenados. Para cada credencial, um valor de sal único e o hash resultante devem ser armazenados. (C6)		✓	✓	916	5.1.1.2
2.4.3	Verifique se se o PBKDF2 for usado, a contagem de iteração deve ser tão grande quanto o desempenho do servidor de verificação permitirá, normalmente pelo menos 100.000 iterações. (C6)		✓	✓	916	5.1.1.2
2.4.4	Verifique se se o bcrypt for usado, o fator de trabalho deve ser tão grande quanto o desempenho do servidor de verificação permitirá, normalmente pelo menos 13. (C6)		✓	✓	916	5.1.1.2
2.4.5	Verifique se uma iteração adicional de uma função de derivação chave é realizada, usando um valor de sal que é secreto e conhecido apenas pelo verificador. Gere o valor de sal usando um gerador de bits aleatório aprovado [SP 800-90Ar1] e forneça pelo menos a força mínima de segurança especificada na última revisão do SP 800-131A. O valor secreto do sal deve ser armazenado separadamente das senhas hashed (por exemplo, em um dispositivo especializado		✓	✓	916	5.1.1.2
2.5.1	Verifique se um sistema gerou ativação inicial ou segredo de recuperação não é enviado em texto claro ao usuário. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Verifique as dicas de senha ou autenticação baseada em conhecimento (as chamadas "perguntas secretas") não estão presentes.	✓	✓	✓	640	5.1.1.2
2.5.3	Verifique a recuperação da credencial por senha não revele a senha atual de forma alguma. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Verifique se as contas compartilhadas ou padrão não estão presentes (por exemplo, "raiz", "administração" ou "sa").	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Verifique se um fator de autenticação for alterado ou substituído, o usuário será notificado deste 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 TOTP ou outro token macio, 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 multifatorial são perdidos, essa evidência de prova de identidade é realizada no mesmo nível que durante a inscrição.		✓	✓	308	6.1.2.3
2.6.1	Verifique se os segredos de busca 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 menos de 112 bits de entropia, salgados com um sal único e aleatório de 32 bits e hashed com um hash de mão única aprovado.		✓	✓	330	5.1.2.2
2.6.3	Verifique se os segredos de procura são resistentes a ataques offline, como valores previsíveis.		✓	✓	310	5.1.2.2
2.7.1	Verifique se autenticadores de texto claro fora da banda (NIST "restrito"), como SMS ou PSTN, não são oferecidos por padrão, e alternativas mais fortes, como notificações push, são oferecidas primeiro.	✓	✓	✓	287	5.1.3.2
2.7.2	Verifique se o verificador fora da banda expira fora de solicitações de autenticação de banda, códigos ou tokens após 10 minutos.	✓	✓	✓	287	5.1.3.2
2.7.3	Verifique se as solicitações de autenticação, códigos ou tokens fora da banda só são utilizáveis uma vez e apenas para a solicitação de autenticação original.	✓	✓	✓	287	5.1.3.2
2.7.4	Verifique se o autenticador e verificador fora da banda se comunica em um canal independente seguro.	✓	✓	✓	523	5.1.3.2
2.7.5	Verifique se o verificador fora da banda retém apenas uma versão hashed 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 números aleatórios seguro, contendo pelo menos 20 bits de entropia (normalmente um número aleatório de seis dígitos é suficiente).		✓	✓	310	5.1.3.2

2.8.1	Verifique se os OTPs baseados no tempo têm uma vida útil definida antes de expirar.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Verifique se as chaves simétricas usadas para verificar os OTPs submetidos são altamente protegidas, como usar um módulo de segurança de hardware ou armazenamento de chaves 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, semeadura e verificação.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Verifique se o OTP baseado no tempo só pode ser usado uma vez dentro do prazo de validade.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Verifique se um token OTP multifatorial baseado no tempo for reutilizado durante o período de validade, ele será registrado e rejeitado com notificações seguras sendo enviadas ao titular do dispositivo.		✓	✓	287	5.1.5.2
2.8.6	Verifique se o gerador OTP de fator único físico pode ser revogado em caso de roubo ou outra perda. Certifique-se de que a revogação seja imediatamente eficaz em sessões registradas, independentemente da localização.		✓	✓	613	5.2.1
2.8.7	Verifique se os autenticadores biométricos estão limitados a usar apenas como fatores secundários em conjunto com algo que você tem e algo que você conhece.		o	✓	308	5.2.3
2.9.1	Verifique se as chaves criptográficas usadas na verificação são armazenadas com segurança e protegidas contra divulgação, como o uso de um TPM ou HSM, ou um serviço de SO que pode usar esse armazenamento seguro.		✓	✓	320	5.1.7.2
2.9.2	Verifique se o desafio nonce tem pelo menos 64 bits de comprimento, e estatisticamente único ou único ao longo da vida útil do dispositivo criptográfico.		✓	✓	330	5.1.7.2
2.9.3	Verifique se algoritmos criptográficos aprovados são usados na geração, semeadura e verificação.		✓	✓	327	5.1.7.2
2.10.1	Verifique se os segredos de integração não dependem de senhas imutáveis, como chaves de API ou contas privilegiadas compartilhadas.		**OS assisted**	**HSM**	287	5.1.1.1
2.10.2	Verifique se se forem necessárias senhas, as credenciais não são uma conta padrão.		**OS assisted**	**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.		**OS assisted**	**HSM**	522	5.1.1.1
2.10.4	Verifique senhas, integrações com bancos de dados e sistemas de terceiros, sementes e segredos internos e as chaves de API são gerenciadas com segurança e não incluídas no código-fonte ou armazenadas nos repositórios de código-fonte. Esse armazenamento deve resistir a ataques offline. O uso de uma loja de chaves de software segura (L1), módulo de plataforma confiável de hardware (TPM) ou um módulo de segurança de hardware (L3) é recomendado para		**OS assisted**	**HSM**	798	
3.1.1	Verifique se o aplicativo nunca revela tokens de sessão em parâmetros de URL ou mensagens de erro.	✓	✓	✓	598	
3.2.1	Verifique se o aplicativo 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 o aplicativo armazena apenas tokens de sessão no navegador usando métodos seguros, como cookies devidamente protegidos (ver seção 3.4) ou armazenamento de sessão HTML 5.	✓	✓	✓	539	7,1
3.2.4	Verifique se o token de sessão é gerado usando algoritmos criptográficos aprovados. (C6)		✓	✓	331	7,1
3.3.1	Verifique se o logout e a expiração invalidam o token de sessão, de tal forma que o botão de trás ou uma parte que depende do rio down não retome uma sessão autenticada, inclusive entre as partes que dependem. (C6)	✓	✓	✓	613	7,1
3.3.2	Se os autenticadores permitirem que os usuários permaneçam logados, verifique se a re autenticação ocorre periodicamente, tanto quando usada ativamente ou após um período ocioso. (C6)	**30 days**	**12 hours or 30 minutes of inactivity, 2FA optional**	**12 hours or 15 minutes of inactivity, with 2FA**	613	7,2
3.3.3	Verifique se o aplicativo encerra todas as outras sessões ativas após uma alteração de senha bem sucedida, e que isso é eficaz em todo o aplicativo, login federado (se presente) e quaisquer partes que dependem.		✓	✓	613	
3.3.4	Verifique se os usuários são capazes de visualizar e sair de qualquer ou todas as sessões e dispositivos atualmente ativos.		✓	✓	613	7,1
3.4.1	Verifique se os tokens de sessão baseados em cookies têm o conjunto de atributos 'Secure'. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Verifique se os tokens de sessão baseados em cookies têm o conjunto de atributos 'HttpOnly'. (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 solicitações entre sites. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Verifique se os tokens de sessão baseados em cookies usam prefixo "__Host- "ver referências) para fornecer confidencialidade de cookies de sessão.	✓	✓	✓	16	7.1.1
3.4.5	Verifique se o aplicativo for publicado em um nome de domínio com outros aplicativos que definem ou usam cookies de sessão que podem substituir ou divulgar os cookies da sessão, defina o atributo path em tokens de sessão baseados em cookies usando o caminho mais preciso possível. (C6)	✓	✓	✓	16	7.1.1
3.5.1	Verifique se o aplicativo não trata o OAuth e atualiza tokens — por conta própria — como a presença do assinante e permite que os usuários encerrem relações de confiança com aplicativos vinculados.		✓	✓	290	7.1.2
3.5.2	Verifique se o aplicativo usa tokens de sessão em vez de segredos e chaves estáticas da API, exceto com implementações legados.		✓	✓	798	
3.5.3	Verifique se os tokens de sessão apátridas usam assinaturas digitais, criptografia e outras contramedidas para proteger contra ataques de adulteração, envolvemento, repetição, cifra nula e substituição de chaves.		✓	✓	345	
3.6.1	Verifique se as partes que dependem especificam o tempo máximo de autenticação para CSPs e que os CSPs re-autenticam o assinante se eles não usaram uma sessão dentro desse período.			✓	613	7.2.1
3.6.2	Verifique se os CSPs informam as partes que dependem do último evento de autenticação, para permitir que os RPs determinem se eles precisam re-autenticar o usuário.			✓	613	7.2.1
3.7.1	Verifique se o aplicativo garante uma sessão de login válida ou requer re autenticação ou verificação secundária antes de permitir quaisquer transações ou modificações de conta sensíveis.	✓	✓	✓	778	
4.1.1	Verifique se o aplicativo impõe regras de controle de acesso em uma camada de serviço confiável, especialmente se o controle de acesso do lado do cliente estiver presente e puder ser ignorado.	✓	✓	✓	602	
4.1.2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	✓	✓	✓	639	
4.1.3	Verifique se o princípio do menor privilégio existe - os usuários só devem ser capazes de acessar funções, arquivos de dados, URLs, controladores, serviços e outros recursos, para os quais possuem autorização específica. Isso implica proteção contra falsificação e elevação de privilégios. (C7)	✓	✓	✓	285	

4.1.4	Verifique se existe o princípio de negar por padrão pelo qual novos usuários/funções começam com permissões mínimas ou sem permissões e os usuários/funções não recebem acesso a novos recursos até que o acesso seja explicitamente atribuído. (C7)	✓	✓	✓	276
4.1.5	Verifique se os controles de acesso falham com segurança, inclusive quando ocorre uma exceção. (C10)	✓	✓	✓	285
4.2.1	Verifique se dados e APIs confidenciais estão protegidos contra ataques diretos de objetos visando a criação, leitura, atualização e exclusão de registros, como criar ou atualizar o registro de outra pessoa, visualizar registros de todos ou excluir todos os registros.	✓	✓	✓	639
4.2.2	Verifique se o aplicativo ou a estrutura impõe um forte mecanismo anti-CSRF para proteger a funcionalidade autenticada e o anti-automação eficaz ou anti-CSRF protege a funcionalidade não autenticada.	✓	✓	✓	352
4.3.1	Verifique se as interfaces administrativas usam autenticação multifatorial apropriada para evitar o uso não autorizado.	✓	✓	✓	419
4.3.2	Verifique se a navegação do diretório está desativada, a menos que seja deliberadamente desejada. Além disso, os aplicativos não devem permitir a descoberta ou divulgação de metadados de arquivos ou diretórios, como pastas Thumbs.db, .DS_Store, .git ou .svn.	✓	✓	✓	548
4.3.3	Verifique se o aplicativo tem autorização adicional (como intensificação ou autenticação adaptativa) para sistemas de menor valor e/ou segregação de direitos para aplicações de alto valor para impor controles antifraude de acordo com o risco de aplicação e fraudes passadas.		✓	✓	732
5.1.1	Verifique se o aplicativo tem defesas contra ataques de poluição por parâmetro HTTP, especialmente se o quadro de aplicação não faz distinção sobre a fonte de parâmetros de solicitação (GET, POST, cookies, cabeçalhos ou variáveis ambientais).	✓	✓	✓	235
5.1.2	Verifique se as estruturas protegem contra ataques de atribuição de parâmetros em massa ou que o aplicativo tem contramedidas para proteger contra atribuição de parâmetros inseguros, como marcar campos privados ou similares. (C5)	✓	✓	✓	915
5.1.3	Verifique se todas as entradas (campos de formulário HTML, solicitações REST, parâmetros de URL, cabeçalhos HTTP, cookies, arquivos em lote, feeds RSS, etc) são validadas usando validação positiva (whitelisting). (C5)	✓	✓	✓	20
5.1.4	(por exemplo, números de cartão de crédito ou telefone, ou validar que dois campos relacionados são razoáveis, como verificar aquele subúrbio e zip/postal match). (C5)	✓	✓	✓	20
5.1.5	Verifique se os redirecionamentos e encaminhamentos de URL só permitem destinos na lista branca ou mostram um aviso ao redirecionar para conteúdo potencialmente não confiável.	✓	✓	✓	601
5.2.1	Verifique se todas as entradas HTML não confiáveis de editores WYSIWYG ou similares são adequadamente higienizadas com uma biblioteca ou recurso de estrutura desinfetante HTML. (C5)	✓	✓	✓	116
5.2.2	Verifique se os dados não estruturados são higienizados para aplicar medidas de segurança, como caracteres e comprimento permitidos.	✓	✓	✓	138
5.2.3	Verifique se o aplicativo higieniza a entrada do usuário antes de passar para sistemas de e-mail para proteger contra a injeção de SMTP ou IMAP.	✓	✓	✓	147
5.2.4	Verifique se o aplicativo evita o uso de eval() ou outros recursos de execução de código dinâmico. Quando não houver alternativa, qualquer entrada do usuário que esteja sendo incluída deve ser higienizada ou sandboxed antes de ser executada.	✓	✓	✓	95
5.2.5	Verifique se o aplicativo protege contra ataques de injeção de modelo, garantindo que qualquer entrada do usuário que esteja sendo incluída seja higienizada ou sandboxed.	✓	✓	✓	94
5.2.6	Verifique se o aplicativo protege contra ataques SSRF, validando ou higienizando dados não confiáveis ou metadados de arquivos HTTP, como nomes de arquivos e campos de entrada de URL, use whitelisting de protocolos, domínios, caminhos e portas.	✓	✓	✓	918
5.2.7	Verifique se o aplicativo higieniza, desativa ou sandboxes de conteúdo scriptável SVG fornecido pelo usuário, especialmente quando eles se relacionam com XSS resultantes de scripts inline e foreignObject.	✓	✓	✓	159
5.2.8	Verifique se o aplicativo higieniza, desativa ou sandboxes fornecido pelo usuário ao conteúdo de linguagem de modelo de script ou de expressão, como folhas de estilo Markdown, CSS ou XSL, BBCode ou similares.	✓	✓	✓	94
5.3.1	atributos HTML, JavaScript, Parâmetros de URL, cabeçalhos HTTP, SMTP e outros como o contexto exige, especialmente de entradas não confiáveis (por exemplo, nomes com Unicode ou apóstrofes, como ねこ ou O'Hara). (C4)	✓	✓	✓	116
5.3.2	Verifique se a codificação de saída preserva o conjunto de caracteres e a localização escolhidos pelo usuário, de modo que qualquer ponto de caractere Unicode seja válido e manuseado com segurança. (C4)	✓	✓	✓	176
5.3.3	Verifique se a saída de saída de saída refletida, armazenada e baseada em DOM, de preferência automatizada - ou na pior das hipóteses, manual - protege contra XSS refletidos, armazenados e baseados em DOM. (C4)	✓	✓	✓	79
5.3.4	Verifique se as consultas de seleção de dados ou banco de dados (por exemplo,SQL, HQL, ORM, NoSQL) usam consultas parametrizadas, ORMs, frameworks de entidades ou estão protegidas de 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 saída específica do contexto é usada para proteger contra ataques de injeção, como o uso de SQL escapando para proteger contra a injeção SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Verifique se o aplicativo protege contra ataques de injeção JavaScript ou JSON, inclusive para ataques de eval, JavaScript remoto inclui, desvios CSP, avaliação de expressão DOM XSS e JavaScript. (C4)	✓	✓	✓	830
5.3.7	Verifique se o aplicativo protege contra vulnerabilidades de injeção LDAP ou que controles de segurança específicos para evitar a injeção de LDAP foram implementados. (C4)	✓	✓	✓	943
5.3.8	Verifique se o aplicativo protege contra a injeção de comando do SISTEMA OPERACIONAL e que as chamadas do sistema operacional usam consultas parametrizadas do SISTEMA OU usam codificação de saída de linha de comando contextual. (C4)	✓	✓	✓	78
5.3.9	Verifique se o aplicativo protege contra ataques LFI (Local File Inclusion, inclusão de arquivos locais) ou de inclusão remota de arquivos (RFI).	✓	✓	✓	829
5.3.10	Verifique se o aplicativo protege contra ataques de injeção XPath ou injeção XML. (C4)	✓	✓	✓	643
5.4.1	Verifique se o aplicativo usa sequência segura de memória, cópia de memória mais segura e aritmética do ponteiro para detectar ou evitar estouros de pilha, buffer ou pilha.		✓	✓	120
5.4.2	Verifique se as sequências de formato não tomam entrada potencialmente hostil e são constantes.		✓	✓	134
5.4.3	Verifique se as técnicas de validação de sinal, intervalo e entrada são usadas para evitar estouros inteiros.		✓	✓	190
5.5.1	Verifique se os objetos serializados usam verificações de integridade ou são criptografados para evitar a criação de objetos hostis ou adulteração de dados. (C5)	✓	✓	✓	502
5.5.2	Verifique se o aplicativo restringe corretamente os analisadores XML para usar apenas a configuração mais restritiva possível e para garantir que recursos inseguros, como a resolução de entidades externas, sejam desativados para evitar o XXE.	✓	✓	✓	611
5.5.3	Verifique se a desserialização de dados não confiáveis é evitada ou está protegida tanto em códigos personalizados quanto em bibliotecas de terceiros (como parsers JSON, XML e YAML).	✓	✓	✓	502
5.5.4	Verifique se ao analisar json em navegadores ou backends baseados em JavaScript, o JSON.parse é usado para analisar o documento JSON. Não use eval() para analisar JSON.	✓	✓	✓	95

6.1.1	Verifique se os dados privados regulamentados são armazenados criptografados enquanto estão em repouso, como informações pessoalmente identificáveis (PII), informações pessoais confidenciais ou dados avaliados que podem estar 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 não anonimizados.		✓	✓	311
6.1.3	Verifique se os dados financeiros regulamentados são armazenados criptografados enquanto estão em repouso, como contas financeiras, inadimplência ou histórico de crédito, registros fiscais, histórico de pagamento, beneficiários ou registros de mercado ou pesquisa não anonimizados.		✓	✓	311
6.2.1	Verifique se todos os módulos criptográficos falham com segurança e os erros são tratados de forma que não habilite ataques Oracle de preenchimento.	✓	✓	✓	310
6.2.2	Verifique se algoritmos, modos e bibliotecas criptográficas comprovados pela indústria ou pelo governo são usados, em vez de criptografia codificada personalizada. (C8)		✓	✓	327
6.2.3	Verifique se os modos vetor, cipher de inicialização e bloqueio de inicialização de criptografia estão configurados com segurança usando os últimos conselhos.		✓	✓	326
6.2.4	Verifique se o número aleatório, os algoritmos de criptografia ou hashing, os comprimentos das chaves, as rodadas, cifras ou modos, podem ser reconfigurados, atualizados ou trocados a qualquer momento, para proteger contra quebras criptográficas. (C8)		✓	✓	326
6.2.5	Verifique se os modos de bloco inseguro conhecidos (ou seja, ECB, etc.), modos de preenchimento (ou seja, PKCS#1 v1.5, etc.), cifras com tamanhos de blocos pequenos (ou seja, Triple-DES, Blowfish, etc.) e algoritmos fracos de hashing (ou seja, MD5, SHA1, etc.) não são usados a menos que sejam necessários para compatibilidade invertida.		✓	✓	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 através 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
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 criptográficos aprovado pelo módulo criptográfico aprovado pelo módulo criptográfico quando esses valores aleatórios são destinados a não 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 pseudoaleatórios (CSPRNG) com proteção criptográfica. GuiDs criados usando outros geradores de números pseudoaleatórios podem ser previsíveis.		✓	✓	338
6.3.3	Verifique se os números aleatórios são criados com entropia adequada mesmo quando o aplicativo está sob carga pesada, ou que o aplicativo se degrada graciosamente em tais circunstâncias.			✓	338
6.4.1	Verifique se uma solução de gerenciamento 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-chave não está exposto ao aplicativo, mas usa um módulo de segurança isolado como um cofre para operações criptográficas. (C8)		✓	✓	320
7.1.1	Verifique se o aplicativo não registra credenciais ou detalhes de pagamento. Os tokens de sessão só devem ser armazenados em logs de forma irreversível e hashed. (C9, C10)	✓	✓	✓	532
7.1.2	Verifique se o aplicativo não registra outros dados confidenciais conforme definido pelas leis de privacidade locais ou pela política de segurança relevante. (C9)	✓	✓	✓	532
7.1.3	Verifique se o aplicativo registra eventos relevantes de segurança, incluindo eventos de autenticação bem-sucedidos e com falha, falhas de controle de acesso, falhas de deserialização e falhas de validação de entrada. (C5, C7)		✓	✓	778
7.1.4	Verifique se cada evento de registro inclui informações necessárias que permitam uma investigação detalhada da linha do tempo quando um evento acontecer. (C9)		✓	✓	778
7.2.1	Verifique se todas as decisões de autenticação estão registradas, sem armazenar identificadores de sessão ou senhas sensíveis. 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 estão registradas. Isso deve incluir solicitações com metadados relevantes necessários para investigações de segurança.		✓	✓	285
7.3.1	Verifique se o aplicativo codifica adequadamente os dados fornecidos pelo usuário para evitar a injeção de log. (C9)		✓	✓	117
7.3.2	Verifique se todos os eventos estão protegidos contra injeção quando visualizados no software de visualização de log. (C9)		✓	✓	117
7.3.3	Verify that security logs are protected from unauthorized access and modification. (C9)		✓	✓	200
7.3.4	Verifique se as fontes de tempo estão sincronizadas com o fuso horário e horário corretos. Considere fortemente o registro apenas na UTC se os sistemas forem globais para ajudar na análise forense pós-incidente. (C9)		✓	✓	
7.4.1	Verifique se uma mensagem genérica é mostrada quando ocorre um erro inesperado ou sensível à segurança, potencialmente com uma ID única que o pessoal de suporte pode usar para investigar. (C10)	✓	✓	✓	210
7.4.2	Verifique se o manuseio de exceção (ou um equivalente funcional) é usado em toda a base de código para explicar as condições de erro esperadas e inesperadas. (C10)		✓	✓	544
7.4.3	Verifique se é definido um manipulador de erros de "último recurso", que irá capturar todas as exceções não manuseadas. (C10)		✓	✓	460
8.1.1	Verifique se o aplicativo protege os dados confidenciais de serem armazenados em cache em componentes do servidor, como balanceadores de carga e caches de aplicativos.		✓	✓	524
8.1.2	Verifique se todas as cópias armazenadas em cache ou temporárias de dados confidenciais armazenados no servidor estão protegidas contra acesso não autorizado ou expurgados/invalidados após o usuário autorizado acessar os dados confidenciais.		✓	✓	524
8.1.3	Verifique se o aplicativo minimiza o número de parâmetros em uma solicitação, como campos ocultos, variáveis Ajax, cookies e valores de cabeçalho.		✓	✓	233
8.1.4	Verifique se o aplicativo pode detectar e alertar em números anormais de solicitações, como por IP, usuário, total por hora ou dia, ou o que faz sentido para o aplicativo.		✓	✓	770
8.1.5	Verifique se backups regulares de dados importantes são realizados e que a restauração de dados do teste é realizada.			✓	19
8.1.6	Verifique se os backups são armazenados com segurança para evitar que os dados sejam roubados ou corrompidos.			✓	19
8.2.1	Verifique se o aplicativo define cabeçalhos anti-cache suficientes para que os dados confidenciais não sejam armazenados em cache em navegadores modernos.	✓	✓	✓	525
8.2.2	Verifique se os dados armazenados no armazenamento lateral do cliente (como armazenamento local HTML5, armazenamento de sessão,DB indexado, cookies regulares ou cookies Flash) não contêm dados confidenciais ou PII.	✓	✓	✓	922

8.2.3	Verifique se os dados autenticados são liberados do armazenamento do cliente, como o DOM do navegador, após o cliente ou sessão ser encerrada.	✓	✓	✓	922
8.3.1	Verifique se os dados confidenciais são enviados ao servidor no corpo de mensagens HTTP ou nos cabeçalhos, e que os parâmetros de sequência 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 seus dados sob demanda.	✓	✓	✓	212
8.3.3	Verifique se os usuários são fornecidos linguagem clara em relação à coleta e uso de informações pessoais fornecidas e que os usuários forneceram consentimento opt-in 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 pelo aplicativo foram identificados e certifique-se de que uma política está em vigor sobre como lidar com dados confidenciais. (C8)	✓	✓	✓	200
8.3.5	Verificar o acesso a dados confidenciais é auditado (sem registrar os dados confidenciais em si), se os dados são coletados sob diretivas relevantes de proteção de dados ou quando o registro de acesso é necessário.		✓	✓	532
8.3.6	se as informações confidenciais contidas na memória são substituídas assim que não for mais necessária para mitigar ataques de dumping de memória, usando zeros ou dados aleatórios.		✓	✓	226
8.3.7	se informações confidenciais ou privadas que são necessárias para serem criptografadas, são criptografadas usando algoritmos aprovados que fornecem confidencialidade e integridade. (C8)		✓	✓	327
8.3.8	Verifique se informações pessoais confidenciais estão sujeitas à classificação de retenção de dados, de modo que dados antigos ou desatualizados são excluídos automaticamente, em um cronograma ou conforme a situação exige.		✓	✓	285
9.1.1	Verifique se o TLS protegido é usado para toda a conectividade do cliente e não recua para protocolos inseguros ou não criptografados. (C8)	✓	✓	✓	319
9.1.2	Verifique usando ferramentas de teste TLS on-line ou atualizadas que apenas algoritmos, cifras e protocolos fortes estão habilitados, com os algoritmos e cifras mais fortes definidos como preferidos.	✓	✓	✓	326
9.1.3	Verifique se versões antigas de protocolos SSL e TLS, algoritmos, cifras e configuração estão desativadas, como SSLv2, SSLv3 ou TLS 1.0 e TLS 1.1. A versão mais recente do TLS deve ser a suíte cifrada preferida.	✓	✓	✓	326
9.2.1	o servidor deve ser configurado apenas para confiar apenas em CAs internos específicos e certificados específicos auto-assinados. Todos os outros devem ser rejeitados.		✓	✓	295
9.2.2	monitoramento, autenticação, API ou chamadas de serviço web, banco de dados, nuvem, sem servidor, mainframe, conexões externas e parceiras. O servidor não deve recuar para protocolos inseguros ou não criptografados.		✓	✓	319
9.2.3	Verifique se todas as conexões criptografadas a sistemas externos que envolvem informações ou funções confidenciais são autenticadas.		✓	✓	287
9.2.4	Verifique se a revogação adequada da certificação, como o Stapling do Protocolo de Status de Certificado Online (OCSP), está ativada e configurada.		✓	✓	299
9.2.5	Verifique se as falhas de conexão TLS de backend estão registradas.			✓	544
10.1.1	Verifique se uma ferramenta de análise de código está em uso que pode detectar códigos potencialmente maliciosos, como funções de tempo, operações de arquivos inseguras e conexões de rede.			✓	749
10.2.1	Verifique se o código-fonte do aplicativo e as bibliotecas de terceiros não contêm recursos de coleta de telefone ou coleta de dados não autorizados. Quando essa funcionalidade existir, obtenha a permissão do usuário para que ele opere antes de coletar quaisquer dados.		✓	✓	359
10.2.2	Verifique se o aplicativo não pede permissões desnecessárias ou excessivas para recursos ou sensores relacionados a privacidade, como contatos, câmeras, microfones ou localização.		✓	✓	272
10.2.3	documentadas, ofuscação de código, blobs binários não documentados, rootkits ou anti-depuração, recursos de depuração inseguros ou de outra forma desatualizados, inseguros ou ocultos funcionalidades que poderiam ser usadas maliciosamente se descobertas.			✓	507
10.2.4	Verifique se o código-fonte do aplicativo e bibliotecas de terceiros não contêm bombas-relógio procurando por funções relacionadas à data e hora.			✓	511
10.2.5	Verifique se o código-fonte do aplicativo e bibliotecas de terceiros não contém código malicioso, como ataques de salame, desvios lógicos ou bombas lógicas.			✓	511
10.2.6	Verifique se o código-fonte do aplicativo e bibliotecas de terceiros não contêm ovos de Páscoa ou qualquer outra funcionalidade potencialmente indesejada.			✓	507
10.3.1	Verifique se o aplicativo tem um recurso de atualização automática de cliente ou servidor, as atualizações devem ser obtidas em 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 o aplicativo emprega proteções de integridade, como assinatura de código ou integridade de sub recurso. O aplicativo não deve carregar ou executar códigos de fontes não confiáveis, como o carregamento inclui, módulos, plugins, código ou bibliotecas de fontes não confiáveis ou da Internet.	✓	✓	✓	353
10.3.3	Verifique se o aplicativo tem proteção contra aquisições de sub-domínios se o aplicativo se baseia em entradas de DNS ou sub-domínios DNS, como nomes de domínio expirados, ponteiros DNS desatualizados ou CNAMEs, projetos vencidos em repositórios de código-fonte público ou APIs de código de nuvem transitórios, funções sem servidor ou baldes de armazenamento (autogen-bucket-id.cloud.example.com) ou similares. As proteções podem incluir garantir que	✓	✓	✓	350
11.1.1	Verifique se o aplicativo só processará fluxos de lógica de negócios para o mesmo usuário em ordem de etapa sequencial e sem pular etapas.	✓	✓	✓	841
11.1.2	Verifique se o aplicativo só processará fluxos de lógica de negócios com todas as etapas sendo processadas em tempo humano realista, ou seja, as transações não são submetidas muito rapidamente.	✓	✓	✓	779
11.1.3	Verificar se o aplicativo tem limites apropriados para ações ou transações comerciais específicas que são corretamente aplicadas por usuário.	✓	✓	✓	770
11.1.4	Verifique se o aplicativo tem controles anti-automação suficientes para detectar e proteger contra exfiltração de dados, solicitações excessivas de lógica empresarial, uploads excessivos de arquivos ou ataques de negação de serviço.	✓	✓	✓	770
11.1.5	se o aplicativo tem limites ou validação da lógica de negócios para proteger contra riscos ou ameaças prováveis dos negócios, identificados usando modelagem de ameaças ou metodologias semelhantes.	✓	✓	✓	841
11.1.6	se o aplicativo não sofre de problemas de "tempo de verificação para tempo de uso" (TOCTOU) ou outras condições de corrida para operações sensíveis.		✓	✓	367
11.1.7	Verifique os monitores de aplicativos para eventos ou atividades incomuns de uma perspectiva 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 o aplicativo tem alerta configurável quando ataques automatizados ou atividade incomum são detectados.		✓	✓	390
12.1.1	Verifique se o aplicativo não aceitará arquivos grandes que possam preencher o armazenamento ou causar um ataque de negação de serviço.	✓	✓	✓	400
12.1.2	Verifique se os arquivos compactados são verificados para "bombas zip" - pequenos arquivos de entrada que descomprimirão em arquivos enormes, esgotando assim os limites de armazenamento de arquivos.		✓	✓	409

12.1.3	Verifique se uma cota de tamanho de arquivo e o 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
12.2.1	Verifique se os arquivos obtidos a partir de fontes não confiáveis são validados como sendo do tipo esperado com base no conteúdo do arquivo.		✓	✓	434
12.3.1	Verifique se os metadados de nome de arquivo enviados pelo usuário não são usados diretamente com o sistema ou arquivo de estrutura e API de URL para proteger contra a travessia de caminho.	✓	✓	✓	22
12.3.2	Verifique se os metadados de nome de arquivo enviados pelo usuário são validados ou ignorados para evitar 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 (RFI), o que também pode levar ao SSRF.	✓	✓	✓	98
12.3.4	parâmetro JSON, JSONP ou URL, o cabeçalho tipo conteúdo de resposta deve ser definido como texto/planície e o cabeçalho de disposição de conteúdo deve ter um nome de arquivo fixo.	✓	✓	✓	641
12.3.5	Verifique se os metadados de arquivo não confiáveis não são usados diretamente com API do sistema ou bibliotecas, para proteger contra a injeção de comando do SISTEMA.	✓	✓	✓	78
12.3.6	Verifique se o aplicativo não inclui e executa a funcionalidade de fontes não confiáveis, como redes de distribuição de conteúdo não verificadas, bibliotecas JavaScript, bibliotecas npm de nó ou DLLs do lado do servidor.		✓	✓	829
12.4.1	Verifique se os arquivos obtidos a partir de fontes não confiáveis são armazenados fora da raiz da web, com permissões limitadas, de preferência com forte validação.	✓	✓	✓	922
12.4.2	Verifique se os arquivos obtidos a partir de fontes não confiáveis são digitalizados por scanners antivírus para evitar o upload de conteúdo malicioso conhecido.	✓	✓	✓	509
12.5.1	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 pelos editores devem ser bloqueadas, a menos que necessário.	✓	✓	✓	552
12.5.2	Verifique se as solicitações diretas aos arquivos carregados nunca serão executadas como conteúdo HTML/JavaScript.	✓	✓	✓	434
12.6.1	Verifique se o servidor web ou aplicativo está configurado com uma lista branca de recursos ou sistemas para os quais o servidor pode enviar solicitações ou carregar dados/arquivos.	✓	✓	✓	918
13.1.1	Verifique se todos os componentes do aplicativo usam as mesmas codificações e analisadores para evitar ataques de análise que exploram diferentes comportamentos de URI ou de análise de arquivos que poderiam ser usados em ataques SSRF e RFI.	✓	✓	✓	116
13.1.2	Verifique se o acesso às funções de administração e gerenciamento está limitado aos administradores autorizados.	✓	✓	✓	419
13.1.3	Verificar urls de API não expõem informações confidenciais, como a chave de API, tokens de sessão etc.	✓	✓	✓	598
13.1.4	Verifique se as decisões de autorização são tomadas tanto no URI, impostas por segurança programática ou declarativa no controlador ou roteador, quanto no nível de recurso, impostas por permissões baseadas em modelos.		✓	✓	285
13.1.5	Verifique se as solicitações que contenham tipos de conteúdo inesperados ou ausentes são rejeitadas com cabeçalhos apropriados (status de resposta HTTP 406 Inaceitável ou 415 Tipo de mídia sem suporte).		✓	✓	434
13.2.1	Verifique se os métodos HTTP RESTful ativados são uma escolha válida para o usuário ou ação, como impedir usuários normais que usam API ou recursos protegidos.	✓	✓	✓	650
13.2.2	Verifique se a validação do esquema JSON está no lugar e verificada antes de aceitar a entrada.	✓	✓	✓	20
13.2.3	Verifique se os serviços web RESTful que utilizam cookies estão protegidos da falsificação de solicitações entre sites através do uso de pelo menos um ou mais dos seguintes: padrão de cookies de envio triplo ou duplo (ver referências), nonces CSRF ou verificações de cabeçalho de solicitação ORIGIN.	✓	✓	✓	352
13.2.4	Verifique se os serviços REST têm controles anti-automação para proteger contra chamadas excessivas, especialmente se a API não for autenticada.		✓	✓	779
13.2.5	Verifique se os serviços REST verificam explicitamente o tipo de conteúdo recebido como sendo o esperado, como aplicativo/xml ou aplicativo/JSON.		✓	✓	436
13.2.6	Verifique se os cabeçalhos de mensagens e a carga são confiáveis e não modificados em trânsito. Exigir criptografia forte apenas para transporte (somente TLS) pode ser suficiente em muitos casos, pois fornece confidencialidade e proteção de integridade. As assinaturas digitais por mensagem podem fornecer garantia adicional em cima das proteções de transporte para aplicativos de alta segurança, mas trazem consigo complexidade adicional e riscos para pesar		✓	✓	345
13.3.1	Verifique se a validação do esquema XSD ocorre para garantir um documento XML devidamente formado, seguido de validação de cada campo de entrada antes que qualquer processamento desses dados ocorra.	✓	✓	✓	20
13.3.2	Verifique se a carga da mensagem está assinada usando o WS-Security para garantir um transporte confiável entre cliente e serviço.		✓	✓	345
13.4.1	de serviço de expressão de GraphQL ou camada de dados (DoS) como resultado de consultas caras e aninhadas. Para cenários mais avançados, deve-se utilizar a análise de custos de consulta.		✓	✓	770
13.4.2	Verifique se a lógica de autorização de GraphQL ou outra camada de dados deve ser implementada na camada lógica do negócio em vez da camada GraphQL.		✓	✓	285
14.1.1	Verifique se os processos de construção e implantação de aplicativos são realizados de forma segura e repetível, como automação de CI/CD, gerenciamento automatizado de configuração e scripts de implantação automatizados.		✓	✓	
14.1.2	randomização de pilha, prevenção de execução de dados e para quebrar a compilação se um ponteiro, memória, sequência de formato, inteiro ou string forem encontrados.		✓	✓	120
14.1.3	Verifique se a configuração do servidor está endurecida de acordo com as recomendações do servidor de aplicativos e as estruturas em uso.		✓	✓	16
14.1.4	Verifique se o aplicativo, a configuração e todas as dependências podem ser re-implantadas usando scripts de implantação automatizados, construídos a partir de um runbook documentado e testado em um tempo razoável ou restaurados a partir 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.			✓	
14.2.1	Verifique se todos os componentes estão atualizados, de preferência usando um verificador de dependência durante o tempo de compilação ou compilação. (C2)	✓	✓	✓	1026
14.2.2	Verifique se todos os recursos, documentação, amostras, configurações não solicitadas são removidos, como aplicativos de amostra, documentação da plataforma e usuários padrão ou exemplo.	✓	✓	✓	1002
14.2.3	Verifique se os ativos do aplicativo, como bibliotecas JavaScript, folhas de estilo CSS ou fontes web, estiverem hospedados externamente em uma rede de entrega de conteúdo (CDN) ou provedor externo, a Integridade Subresource (SRI) será usada para validar a integridade do ativo.	✓	✓	✓	714
14.2.4	Verifique se os componentes de terceiros vêm de repositórios pré-definidos, confiáveis e continuamente mantidos. (C2)		✓	✓	829

14.2.5	Verifique se um catálogo de inventário é mantido em todas as bibliotecas de terceiros em uso. (C2)		✓	✓	
14.2.6	Verifique se a superfície de ataque é reduzida pelo sandboxing ou encapsulando bibliotecas de terceiros para expor apenas o comportamento necessário no aplicativo. (C2)		✓	✓	265
14.3.1	Verifique se as mensagens de erro do servidor e da estrutura da web ou do aplicativo estão configuradas para fornecer respostas personalizadas e acionáveis pelo usuário para eliminar quaisquer divulgações de segurança não intencionais.	✓	✓	✓	209
14.3.2	Verifique se os modos de depuração do servidor web ou aplicativo e da estrutura de aplicativos 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 expõem informações detalhadas da versão dos componentes do sistema.	✓	✓	✓	200
14.4.1	Verifique se cada resposta HTTP contém um cabeçalho de tipo de conteúdo especificando um conjunto de caracteres seguro (por exemplo, UTF-8, ISO 8859-1).	✓	✓	✓	173
14.4.2	Verifique se todas as respostas da API contêm disposição de conteúdo: anexo; filename= "api.json" (ou outro nome de arquivo apropriado para o tipo de conteúdo).	✓	✓	✓	116
14.4.3	Verifique se uma política de segurança de conteúdo (CSPv2) está em vigor que ajuda a mitigar o impacto para ataques XSS como vulnerabilidades de injeção HTML, DOM, JSON e JavaScript.	✓	✓	✓	1021
14.4.4	Verifique se todas as respostas contêm opções tipo X-Content: nosniff.	✓	✓	✓	116
14.4.5	Verifique se os cabeçalhos HTTP Strict Transport Security estão incluídos em todas as respostas e em todas as subdomínios, como Strict-Transport-Security: max-age=15724800; incluiSubdomains.	✓	✓	✓	523
14.4.6	Verifique se está incluído um cabeçalho "Referrer-Policy" adequado, como "não-referrer" ou "mesma origem".	✓	✓	✓	116
14.4.7	Verifique se um quadro-opções ou política de segurança de conteúdo adequada: o cabeçalho de ancestrais de quadros está em uso para sites onde o conteúdo não deve ser incorporado em um site de terceiros.	✓	✓	✓	346
14.5.1	Verifique se o servidor de aplicativo só aceita os métodos HTTP em uso pelo aplicativo ou API, incluindo OPTIONS de pré-vôo.	✓	✓	✓	749
14.5.2	Verifique se o cabeçalho Origin fornecido não é usado para decisões de autenticação ou controle de acesso, pois o cabeçalho Origin pode ser facilmente alterado por um invasor.	✓	✓	✓	346
14.5.3	Verifique se o cabeçalho de compartilhamento de recursos entre domínios (CORS) Access-Control-Allow-Origin usa uma rigorosa lista branca de domínios confiáveis para corresponder e não suporta a origem "nula".	✓	✓	✓	346
14.5.4	Verifique se os cabeçalhos HTTP adicionados por um proxy confiável ou dispositivos SSO, como um token portador, são autenticados pelo aplicativo.		✓	✓	306
		119	**255**	**274**	