Normas de orientação para terceiros do sector sobre "Codificação Segura"

Edição final 1.0

Data: 19/10/2016,

Segurança da BT

Índice

1. Introdução

1.1. Enquadramento

Este documento estabelece uma orientação geral com base em boas práticas para o desenvolvimento de software realizado por terceiros que forneçam aplicações ou sistemas à BT.

As vulnerabilidades ao nível da aplicação são uma forma extremamente eficaz de obter acesso não autorizado a dados ou sistemas. Por isso, é cada vez mais importante que os métodos de desenvolvimento de software incluam considerações de segurança na sua essência.

1.2. Objetivos

Este documento procura fornecer um conjunto de diretrizes sobre como desenvolver software de uma forma que garanta que o código resultante seja fiável e seguro. Um projeto de desenvolvimento de software engloba vários aspetos, e trata-se de uma área em rápida mutação, portanto, dentro do possível, delineámos um conjunto de princípios de alto nível para as diferentes fases de um projeto de software. O documento está dividido em três secções principais:

Conceção do sistema

Esta secção fornece orientação para o início de um projeto de software e descreve as várias áreas que devem ser consideradas antes de escrever qualquer código. Seguindo estes passos, será mais fácil garantir que se programa um código seguro numa fase posterior do projeto.

Entrega do sistema

Esta secção diz respeito ao desenvolvimento do código propriamente dito. Abrange alguns princípios gerais (por exemplo, a gestão de código fonte), bem como a forma de evitar alguns erros/problemas com que nos deparamos habitualmente.

Garantia do sistema:

Esta secção abrange a implementação e a gestão durante o ciclo de vida do código programado. Isto nem sempre será relevante, dependendo do contrato em particular, mas deverá garantir que a sua aplicação ou sistema é capaz de operar dentro destas diretrizes.

Incluímos ainda vários anexos que descrevem as especificações de Controlo de acessos e Criptografia da BT. É importante assegurar-se de que a sua aplicação atende a estes controlos.

1.3. Materiais adicionais e aconselhamento

Conforme as ligações no documento.

2. Conceção do sistema

Ref.	Controlo	Razão
1	Os dados devem ser classificados de acordo com as Normas de classificação de dados, conforme estipulado pela sua organização. Isto inclui informações de conceção e de desenvolvimento (por ex.: conceção, problemas, requisitos), bem como dados da aplicação. Os dados devem ser mantidos em conformidade com a Política de conservação de informações	Ao compreender o tipo de dados que o seu sistema irá conter, poderá implementar as proteções adequadas. É importante tornar seguras as informações de conceção e desenvolvimento (fatores particularmente sensíveis), uma vez que o acesso não autorizado poderia ajudar alguém a atacar a aplicação.
2	Devem ser identificados todos os requisitos legais e regulamentares.	Ao escrever aplicações que lidam com determinados tipos de dados (por ex.: informações de cartões de crédito), há requisitos legais e regulamentares específicos que afetam a forma como se tratam os dados.
3	O processo de conceção deve considerar se existem ou não requisitos de segurança para qualquer pessoa que trabalhe no projeto. (Por ex.: apenas no Reino Unido ou se será necessária uma autorização de segurança)	Projetos que lidam com informação governamental ou sensível podem comportar requisitos sobre quais os funcionários que devem ser usados para trabalhar nesses projetos.
4	Use técnicas de análise de riscos para identificar e quantificar riscos de segurança detalhados. Documente os riscos e as resoluções propostas.	Cada aplicação tem um conjunto de riscos associados diferente (por ex.: qual o ambiente em que se insere, que nível de dados gere). É importante garantir que estes riscos foram considerados e que foram criadas medidas de proteção contra os mesmos. A análise de riscos pode depender do ambiente, mas existem muitas ferramentas para ajudar na

		tarefa. A OWASP elaborou um guia com vista a identificar e avaliar os riscos, onde se inclui uma secção útil sobre como avaliá-los. Alternativamente, a Microsoft desenvolveu uma ferramenta para ajudar a identificar ameaças, e outra para compreender a sua superfície de ataque antes e depois da implementação de novas aplicações. Se tiver quaisquer dúvidas ou necessitar de conselhos sobre metodologias, contacte o proprietário da norma. Nota: a técnica a usar pode ser específica do projeto, por exemplo, IS1 para trabalho certificado pelo governo britânico.
5	Defina e documente um cronograma de avaliação dos riscos.	Isto assegurará que os riscos são mantidos atualizados relativamente a quaisquer alterações na aplicação ou no respetivo ambiente. Podem ser adicionados novos riscos e as ações de resolução levadas a cabo.
6	Documente a Arquitetura de segurança para abordar cada risco identificado acima. Forneça diretrizes de segurança para conceções de baixo nível/implementação; bibliotecas fiáveis cifras aceitáveis: algoritmos hash aceitáveis; comprimentos de chave mínimos; procedimentos de gestão de chaves; segregação de armazenamento/encriptação; acesso/transmissão de credenciais; protocolos inaceitáveis.	Isto garantirá que são aplicadas resoluções adequadas para os riscos identificados e que tais resoluções são aplicadas de uma forma coerente. Ajudará também a conceber os testes de aceitação.

7	Defina e documente o ciclo de vida de um desenvolvimento.	Ter um ciclo de vida definido para um desenvolvimento significa que existe uma abordagem consistente e repetível para passar dos requisitos/questões a uma solução em produção. Sem isto, poderá ser difícil e demorado corrigir problemas de segurança. A Microsoft possui um modelo que pode ser usado como ponto de partida.
8	Documente e implemente os seguintes procedimentos de conceção: Controlo de acessos (quem pode aceder a que partes da aplicação, e de que forma o acesso é pedido, mantido e revogado) — se isto não for gerido, é provável que as pessoas acabem por ter acessos inadequados (por exemplo, os utilizadores serem capazes de executar tarefas de administração, ou manterem o acesso mesmo após terem mudado de emprego); Sincronização do relógio — se os relógios não forem precisos, torna-se muito difícil usar ficheiros de registo para identificar problemas, especialmente onde for necessário estabelecer correlações com atividade em outros sistemas; Transferências de dados — os dados em trânsito devem estar rodeados de proteções adequadas à sua classificação. A forma como as ferramentas e técnicas podem ser usadas para atacar o sistema — isto permitir-lhe-á antecipar-se a	A segurança das aplicações depende fortemente da forma como é gerida durante a sua vida útil. Definindo e implementando os processos sugeridos, é possível garantir que o trabalho realizado nas fases de conceção e implementação não é invalidado durante a execução da aplicação. Muitos destes pontos são do “senso comum” e serão seguidos naturalmente. Porém, é importante explicitar os processos para garantir que são seguidos ao longo de toda a vida da aplicação (por exemplo, caso a equipa de assistência seja alterada).

	tentativas e implementar resoluções desde o primeiro dia; Recuperação após desastre/contingência — deverá ser claramente definida e testada, para que o sistema possa reagir a eventos inesperados. Documente e implemente os seguintes procedimentos de desenvolvimento: Inclua ferramentas de teste de Integração Contínua — um ambiente de compilação automatizada permite que cada compilação seja automaticamente testada no que toca a problemas comuns antes do lançamento da aplicação; Use ferramentas de análise de segurança estáticas e dinâmicas como parte do processo de desenvolvimento — isto ajudará a encontrar vulnerabilidades numa fase inicial do processo de desenvolvimento; Identifique problemas de segurança no seu repositório de origem e controlador de problemas — identificando erros de segurança é possível avaliar a eficácia das resoluções/correções. Podem também contribuir para a análise de risco do projeto permitindo partilhar os problemas comuns com outros programadores para melhorar o conhecimento geral; Use processos de revisão entre pares — o processo de revisão de código entre pares permite identificar problemas que não podem ser detetados automaticamente e evitar vulnerabilidades causadas por erros de	

	codificação — por exemplo erros de lógica; Efetue a implementação a partir de um ambiente de compilação limpo — usar um ambiente de compilação limpo e fiável reduz o risco de resultados corrompidos com malware, bibliotecas indesejadas ou componentes que possam produzir falhas que comprometam a segurança; Partilhe as melhores práticas e experiências — isto pode ajudar a comunidade a evitar problemas comuns suscetíveis de conduzir a vulnerabilidades. Documente e implemente os seguintes procedimentos de gestão durante o ciclo de vida: Aplicação oportuna de correções de segurança, incluindo correções de segurança para componentes de terceiros — isto irá corrigir problemas de segurança que poderiam comprometer a sua aplicação e os respetivos dados; Manutenção de hardware e software — é importante planear como manter a assistência do hardware e software para poder corrigir problemas e receber atualizações de segurança; Arranque/encerramento da aplicação — um encerramento incompleto das aplicações pode conduzir a comportamentos inesperados, ou a artefactos não limpos; Aplicação e gestão de tarefas — a forma como a aplicação gere tarefas deve ser definida, especialmente sob cargas muito	

	pesadas; Monitorização e alarmes — a forma como os registos e outros resultados são monitorizados e como se lhes responde deve ser definida e implementada; Arquivo/limpeza de dados — pode ser necessário gerir os dados de forma que sejam removidos ou arquivados quando já não forem necessários. Este pode ser um requisito da legislação de proteção de dados; Migração — serve para permitir a continuação do serviço após uma atualização, por exemplo, para corrigir problemas de segurança; Controlo de alterações — é importante que as alterações à aplicação sejam controladas para permitir compreender se são legítimas e, se suscitarem problemas, o que pode tê-los causado.	
9	A arquitetura de segurança deve ser revista e atualizada a cada ciclo ou iteração. As alterações planeadas à aplicação devem ser revistas para garantir que não comprometem os requisitos de segurança	Assim, é possível garantir que se mantém atualizada relativamente a alterações aos requisitos ou à implementação do projeto.
10	Use kits de ferramentas para as normas de segurança do sector sujeitos a manutenção ativa, em vez de os conceber.	Os kits de ferramentas mais usados estão bastante testados, são menos suscetíveis de conter falhas relevantes e serão rapidamente alvo de correções em caso de problemas. Isto poupar-lhe-á tempo! Por exemplo: OpenSSL, OpenSSH, Solaris KCF, Windows Cryptography API, Active Directory,

		OpenLDAP.
11	Use apenas componentes e serviços necessários para oferecer a funcionalidade dos produtos. Não exponha quaisquer interfaces para além do necessário	Ao minimizar funcionalidades desnecessárias, é possível reduzir a exposição da aplicação e, por conseguinte, reduzir o impacto dos erros eventualmente encontrados.
12	Use apenas bibliotecas ou componentes de terceiros sujeitos a manutenção ativa e com assistência — no caso do software com assistência comercial, isto significa que deverá manter um contrato de assistência vitalício com o fornecedor da aplicação.	Isto implica que, caso sejam encontrados problemas de segurança, seja capaz de atualizar os componentes usados pela sua aplicação. No caso de software de código aberto, isto pode ser difícil de avaliar, já que a manutenção se baseia no envolvimento da comunidade. Em caso de dúvida, apenas deverá usar um componente se for capaz de assumir a respetiva manutenção.
13	As funcionalidades de controlo de segurança devem ser definidas num conjunto mínimo de locais, claramente definidos e documentados no código. Documente os locais das funcionalidades de controlo de segurança.	O impacto das alterações pode ser avaliado mais facilmente e é mais fácil corrigir falhas de segurança. Mantendo as funcionalidades de segurança num pequeno número de locais é mais fácil compreender e testá-las do que quando se encontram dispersas numa extensa base de código.
14	Assegure-se de que utiliza e mantém versões atuais e testadas de componentes de terceiros.	Versões de software mais antigas podem conter problemas ou falhas de segurança que podem conduzir a erros ou acessos não autorizados à aplicação.
15	O acesso ao sistema deve seguir os controlos indicados na secção Controlo e gestão de acessos abaixo.	Isto garantirá que apenas pessoas autorizadas têm acesso aos dados da aplicação.
16	A implementação predefinida do sistema deve resultar em privilégios mínimos.	Isto reduz o risco de uma implementação predefinida resultar desnecessariamente num acesso com altos privilégios. Caso seja

		necessário, deve ser explicitamente ativado para impedir o esquecimento durante a implementação.
17	O software deve ser executado apenas com os privilégios que necessita para executar a tarefa indicada. Nos casos em que sejam necessários privilégios mais elevados, estes deverão ser obtidos da forma mais conhecida, conforme documentado na arquitetura de segurança, e renunciados logo que possível.	Ao executar a aplicação com privilégios mínimos, o impacto de erros e falhas de segurança pode ser reduzido, já que explorá-los apenas concederia acesso mínimo.
18	As tarefas que exigem privilégios elevados persistentes devem ser separadas do software da aplicação principal e sujeitas ao máximo escrutínio durante a revisão por pares. Preste especial atenção às tarefas que requeiram recorrer/voltar a recorrer a software não fiável.	Fazê-lo significa que quaisquer erros ou falhas de segurança na aplicação principal não resultarão num acesso de alto privilégio. Reduzindo a quantidade de código executada com privilégios elevados, o risco de falhas ou erros de segurança que resultem em acessos de alto privilégio diminui.
19	Crie um plano para testar a sua aplicação, incluindo testes centrados em controlos de segurança ou resoluções que tenha implementado.	
20	Defina limites de segurança que permitam controlar: direitos de acesso aos dados; autenticidade e integridade das mensagens; validação de entradas e saídas.	Isto garante que os controlos são aplicados no local correto para proteger os dados na aplicação. A incapacidade para implementar os controlos no local correto pode levar ao acesso ou modificação não autorizada dos dados.
21	Os sistemas devem garantir a manutenção e preservação da integridade dos dados.	Um controlo deficiente sobre a integridade dos dados pode resultar em fraude e não conformidade com os requisitos normativos, como Sarbanes-Oxley, ou com normas do sector, como PCI DSS.

3. Entrega do sistema

Ref.	Política	Razão
22	Armazene o seu código fonte num ambiente que restrinja o acesso de forma a que apenas pessoas autorizadas possam fazer alterações.	Isto reduzirá o risco de alterações não autorizadas ou maliciosas ao código que poderiam comprometer a segurança da aplicação. Usar um sistema de controlo de revisões, como Git ou SVN com acesso restringido a um grupo de indivíduos designados é a forma mais fácil de manter esse controlo.
23	Garanta que todas as alterações de código estão associadas a um indivíduo designado. Todas as alterações devem ser auditáveis para identificar quem fez o quê, quando e como.	Isto significa que quaisquer alterações de código (por ex., que introduzam funcionalidades indesejadas) podem ser associadas a um indivíduo. Usar um sistema de controlo de revisões, como Git ou SVN com acesso restringido a um grupo de indivíduos designados é a forma mais fácil de manter esse controlo.
24	Siga as diretrizes de melhores práticas relevantes para o seu ambiente de desenvolvimento ou tipo de aplicação. Assegure-se de que conhece sempre os problemas comuns da plataforma ou linguagem que está a usar.	Estes são frequentemente publicados pelos fornecedores dos conjuntos de ferramentas ou outros grupos terceiros. Compreender e acompanhar estas questões ajudará a evitar problemas comuns. Abaixo, encontra-se uma seleção de guias habitualmente usados: Aplicações Web: o top 10 da OWASP descreve as vulnerabilidades comuns nas aplicações web. Aplicações para Android: CERT norma de codificação segura para android: Windows/Windows Phone: iOS/Mac OS X: Guia de codificação segura da Apple: Linux: Para mais informações ou aconselhamento, contacte o

		proprietário da norma.
25	Use um estilo de codificação consistente e documentado.	Se os estilos de codificação forem inconsistentes na base de código da aplicação, poderá ser difícil entender a lógica da aplicação e, assim, detetar problemas que poderiam conduzir a comportamentos indesejados. A maioria dos fornecedores elabora um guia de estilo para a sua linguagem/ambiente cuja utilização se recomenda na ausência de qualquer outro requisito. Alguns guias de estilo sugeridos: Java — Normas de codificação SEI CERT Oracle para Java C/C++ — Diretrizes de codificação segura CERT ou diretrizes MISRA-C C#: Python — PEP8 php — Guia de estilo do CodeIgniter
26	Use cadeias de ferramentas fiáveis e com assistência.	As ferramentas com assistência são as que estão prontamente disponíveis e sujeitas a manutenção ativa (quer por um único fornecedor, por ex.: Microsoft Visual Studio, ou por uma comunidade, por ex.: GNU Compilers Collection). As ferramentas fiáveis são as fornecidas (e assinadas) por um terceiro de confiança (por ex.: Microsoft, Ubuntu) ou amplamente disponíveis e verificáveis através de hashes bem conhecidos (por ex.: transferências de origem GCC). Usando ferramentas com assistência, receberá atualizações para corrigir erros e outros problemas de segurança. Verificando a fiabilidade das ferramentas usadas garantirá que os seus conjuntos de ferramentas não introduzem nenhuma funcionalidade indesejada.

27	Efetue a manutenção das cadeias de ferramentas.	Certifique-se de que todas as atualizações de segurança são aplicadas às ferramentas de compilação antes da utilização. Isto destina-se a corrigir problemas da cadeia de ferramentas que poderiam introduzir comportamentos indesejados nos binários de saída.
28	Use linguagem e funcionalidades da cadeia de ferramentas que ajudem a identificar ou solucionar problemas simples.	Ao ativar avisos do compilador ou runtime poderá detetar preventivamente problemas que poderiam provocar falhas ou erros de segurança (por ex.: -fstack-protector no gcc avisa sobre possíveis condições de capacidade da memória intermédia excedida; ou /GS ou /SafeSEH no VisualC++ para dificultar o exploit de transbordos nas memórias intermédias baseadas em pilha).
29	Use funcionalidades do sistema operativo que mitiguem vetores de ataque comuns.	Isto é específico do sistema operativo, mas deverá ativar as funcionalidades de segurança disponíveis para aumentar a proteção da sua aplicação, por exemplo, ASLR ou DEP. É disponibilizada assistência para os sistemas operativos Microsoft. No caso do linux, ASLR e DEP estão novamente ativos por predefinição (para kernels superiores a 2.6.12). O SELinux é uma forma de aplicar políticas de controlo de acessos e pode ser usado para reduzir o impacto de um exploit. A Redhat possui um bom guia sobre o desenvolvimento de políticas.
30	Limpe todas as entradas de acordo com a arquitetura de segurança antes do processamento. Converta as entradas para uma forma canónica antes de limpar.	Se uma entrada do utilizador for usada como base para comandos ou operações da base de dados, essa entrada poderá ser usada para criar operações indesejadas, por ex.: injeção de SQL. As entradas podem adquirir formas invulgares, por exemplo objetos java serializados, por isso, é importante pensar em todos os casos em que um utilizador/intermediário seja capaz

		de modificar ou injetar conteúdo. As entradas que não sejam convertidas podem ignorar a limpeza usando conjuntos de caracteres que a limpeza não foi projetada para gerir.
31	Limpe todas as saídas para outros sistemas de acordo com a arquitetura de segurança.	Nos casos em que é usada uma entrada do utilizador como saída (ou para outra aplicação ou de volta ao utilizador), a entrada poderá ser manipulada para causar comportamentos indesejados no sistema recetor da saída (por ex.: scripts entre sites).
32	Implemente encriptações de acordo com os controlos da secção criptografia abaixo.	A criptografia pode ser complexa e difícil de executar. Seguindo os controlos apropriados, é possível garantir que o nível de criptografia usado é apropriado para a informação que pretende proteger.
33	Implemente mecanismos para evitar operações de memória não seguras. Consulte a OWASP	Algumas operações de memória podem fazer com que um utilizador sem privilégios tenha a capacidade de manipular o conteúdo da memória. Em alguns contextos, isto poderá conduzir à manipulação do fluxo do programa (por ex.: ignorar verificações) ou à execução de código arbitrário.
34	Use ferramentas antiadulteração onde indicado pela avaliação de risco.	Permitir alterações ao firmware ou ao código da aplicação poderá fazer com que as funções de segurança sejam ignoradas. Isto poderá aplicar-se aos dispositivos móveis, onde pode optar por impedir que a sua aplicação seja usada em dispositivos desbloqueados por “rooting” ou “jailbreak”. Poderá optar por dificultar o processo de análise runtime/depuração por parte de utilizadores que pretendam efetuar a engenharia inversa de funcionalidades da aplicação.
35	Use técnicas anti-inversão onde indicado pela avaliação de risco.	A engenharia inversa poderá revelar funções de segurança. No entanto, mantenha presente que a maioria das técnicas anti-inversão só pode aumentar o tempo necessário para efetuar a engenharia inversa e

		não pode evitá-la completamente, por isso, esta não deve ser a única forma de defesa de segurança.
36	Evite ficheiros temporários. Não use a função tmpnam() ou semelhante para gerar nomes de ficheiro.	O uso de ficheiros temporários pode resultar em dados distribuídos amplamente num disco, os quais podem ser recuperáveis caso seja obtido acesso não autorizado ao sistema. A função tmpnam() deverá ser evitada porque entre a criação do nome e a abertura do ficheiro é possível que outro processo tenha criado um ficheiro com o mesmo nome usando tmpnam. Isto poderia fazer com que outro processo fosse capaz de afetar a integridade dos dados armazenados ou de ler dados aos quais não deveria ter acesso.
37	Não utilize palavras-passe predefinidas (hard coded).	As palavras-passe predefinidas na aplicação costumam tornar-se bastante conhecidas e partilhadas pelos utilizadores. Isto poderá dar acesso a utilizadores não autorizados, ou fazer com que seja difícil comprovar quem tem acesso por não estarem atribuídas a um único utilizador. É ainda importante atualizar e gerir as palavras-passe predefinidas em conformidade com as especificações da BT sobre a gestão de contas.
38	Implemente mecanismos que impeçam o acesso não autorizado aos dados na memória	Dependendo da sua plataforma e avaliação de risco, poderá ter de tomar medidas para impedir que alguém com acesso à máquina que executa a sua aplicação recupere dados sensíveis da memória. Poderá fazê-lo: • substituindo partes da memória quando já não forem necessárias • afixando a memória • usando construções de framework concebidas para proteger a informação, por ex.: a classe SecureString em .Net

39	Os relatórios de erro devem conter informações suficientes para identificar a causa dos problemas. Quando os erros são apresentados aos utilizadores, não devem revelar informações sobre questões internas da aplicação. Por exemplo, não use tipos de exceção genérica para produzir condições de erro específicas.	Relatórios de erro incompletos podem frustrar as investigações aos problemas e esconder atividade não autorizada. Normalmente, um exploit gera erros numa fase inicial enquanto o acesso está a ser obtido e, se devidamente registados, estes erros podem ajudar na investigação do problema. É difícil compreender o que causou a condição de erro se apenas for registada uma exceção genérica. Se a condição de erro foi causada por um ataque é mais difícil deduzir que operações estavam a ser realizadas.
40	Todo o software deve ser testado antes da migração para o ambiente ao vivo. Estabeleça o plano de teste criado como parte da conceção da aplicação.	Software não testado poderá: introduzir riscos de segurança inaceitáveis; não funcionar de acordo com os requisitos, especialmente os requisitos de segurança detalhados no Registo de Segurança; afetar negativamente as operações existentes.
41	Todos os controlos de segurança devem ser testados especificamente e não podem ser contornados.	Vulnerabilidades não identificadas em controlos de segurança devem ser exploradas no ambiente ao vivo.
42	Os dados de teste devem ser eliminados após um período determinado pelo proprietário dos dados.	A divulgação dos dados de teste pode fornecer uma perspetiva do sistema dador.

4. Garantia do sistema

Ref.	Controlo	Razão
43	As vulnerabilidades de segurança identificadas dentro do código ou quaisquer	Vulnerabilidades não corrigidas podem ser utilizadas como parte de um ataque à aplicação ou ao sistema.

	componentes usados pelo código serão categorizadas de acordo com os critérios de avaliação de vulnerabilidades da BT. As vulnerabilidades devem ser corrigidas em conformidade com o cronograma associado a cada uma das categorias. Este poderá ser ajustado com base na situação de risco da plataforma onde a aplicação se encontra.	
44	Rastreie e identifique alterações para corrigir problemas de segurança explicitamente no controlo de alterações.	Ao rastrear as correções de segurança será capaz de examinar rapidamente a sua aplicação/ambiente e provar que as correções foram aplicadas.
45	Instale versões da aplicação que tenham sido compiladas de origem por meio de integração contínua.	A implementação a partir de um ambiente de compilação limpo significa que a aplicação começa a sua vida operacional a partir de um conjunto de código conhecido e que quaisquer problemas podem ser rastreados até ao código fonte que os introduziu.
46	Certifique-se de que é usada a mais recente versão da aplicação e de quaisquer componentes de terceiros a que esta recorra.	Atualizando os componentes de terceiros, reduz-se o risco de um problema de segurança num componente explorado.
47	Aplique correções ao ambiente que aloja a aplicação em conformidade com a política de aplicação de correções.	É importante aplicar regularmente correções de segurança, já que as versões de software antigas se tornam alvo de exploração frequente. Um problema de segurança no ambiente de alojamento poderá colocar os dados da aplicação em risco.
48	As seguintes áreas de processo devem ser definidas e documentadas:	A falta de procedimentos operacionais documentados muitas vezes implica negligenciar processos importantes,

	aplicação de correções de segurança; controlo de acessos; arranque/encerramento; sincronização de relógio, aplicação e gestão de tarefas; transferências de dados; monitorização e alarmes; gestão de problemas e escalamentos; cópias de segurança e recuperação; arquivo; migração; controlo de alterações; recuperação após desastre/contingência; manutenção de hardware e software; registo, análise e ações sobre eventos invulgares.	ou o desconhecimento da equipa sobre como executá-los.

5. Anexos:

5.1.controlo de acessos;

Ref.	Política	Razão
49	Defina os direitos de acesso ao sistema de utilizadores e outros sistemas com os quais interage. Os direitos de acesso devem basear-se em:	Uma definição incorreta das funções poderá resultar na execução de operações acidentais ou maliciosas no sistema.

	operações a ser levadas a cabo pelo sistema; interação sistema a sistema; política de acessos dos utilizadores a informação e sistemas. Os utilizadores devem ter privilégios mínimos para desempenhar as suas tarefas.	
50	Contas privilegiadas partilhadas* devem ser geridas conforme definido.	Sem controlo formal é difícil localizar os responsáveis pelas alterações que afetam a segurança.
51	Os processos da aplicação devem ser concebidos de forma a serem executados com os direitos de acesso mínimos exigidos para a operação correta. operações a ser levadas a cabo pelo sistema; interação sistema a sistema; Não use contas com privilégios para os processos da aplicação. Documente todos os acessos privilegiados. O escalamento de privilégios deve usar apenas API conhecidas. O escalamento de privilégios deve durar	Os processos da aplicação executados com privilégios excessivos podem ser explorados para obter acesso a dados ou privilégios do sistema.

	apenas o tempo mínimo necessário.	
52	As sessões de utilizador devem ser terminadas após um período máximo de inatividade de 30 minutos. Quando o tempo limite é excedido, toda a informação apresentada no ecrã deverá ser eliminada.	Os períodos de tempo excedido limitam a possibilidade de acesso não autorizado caso o sistema seja deixado sem vigilância.
53	Uma sessão de utilizador não deve ultrapassar as 12 horas.	Garanta que todos os utilizadores iniciam sessão e voltam a autenticar-se a cada 12 horas.
54	Deve ser fornecido controlo de acessos baseado em privilégios e funções em todas as portas de gestão, quer locais (porta série/terminal de consola ou servidor de terminal) quer remotas (via EMS). Devem ser aplicadas ACL a todas as interfaces de gestão (IP), restringindo o endereço e a porta de origem, e o protocolo invocado; particularmente, capacidade para restringir o acesso a ICMP, HTTP, SNMP, FTP, TFTP, SSH, Telnet e protocolos de encaminhamento avançado, como OSPF.	Interfaces de gestão expostas podem ser exploradas para obter acessos não autorizados.
55	Apenas podem ser usados protocolos de gestão autenticados de forma segura, por ex.:	Protocolos de gestão inseguros podem ser explorados para acessos não autorizados.

	SNMP v3 (AuthnNoPriv no mínimo) SSHv2 — ver secção de criptografia HTTPS — ver secção de criptografia	

5.2 Criptografia

Ref.	Política	Razão
56	Devem ser usadas bibliotecas criptográficas atuais.	As bibliotecas criptográficas são atualizadas regularmente. Além de atualizar pacotes de software em linha com a orientação do fornecedor, os pacotes criptográficos devem ser revistos e atualizados regularmente.
57	Use apenas conjuntos de cifras aprovados pelas normas do sector sobre encriptação. Por ex.: TLS SSLv2. Observe o aviso acima. Em caso de dúvida, obtenha aconselhamento do ITSAC.	Cifras não aprovadas podem introduzir vulnerabilidades.
58	Em novas implantações deverá ser usada a mais recente versão do TLS. Não devem ser usadas as versões 1, 2 e 3 de SSL.	Versões anteriores, até e incluindo a TLS1.0 já não são consideradas seguras.
59	O Perfect Forward Secrecy deve ser ativado.	Algoritmos com Perfect Forward Secrecy evitam que as mensagens capturadas sejam desencriptadas, mesmo que a chave privada de autenticação seja comprometida no futuro.
60	Não devem ser usados certificados	Os certificados autoassinados invalidam o benefício da

	autoassinados.	autenticação de ponto final e também diminuem significativamente a capacidade de um indivíduo para detetar um ataque “man-in-the-middle”.
61	As palavras-passe devem ser protegidas usando uma função matemática unidirecional não reversível (por exemplo, algoritmo de hashing) com um fator aleatório exclusivo (Salt) por palavra-passe.	Os ficheiros de palavras-passe armazenados podem ser extraídos e, como tal, todas as entradas devem ser protegidas para impedir a recuperação de palavras-passe em texto não encriptado.
62	As palavras-passe protegidas devem ser armazenadas longe dos ficheiros de configuração de um sistema e com o controlo de acessos implementado de forma que apenas utilizadores com os privilégios apropriados possam ler ou copiar o conteúdo.	Nunca deve ser possível recuperar palavras-passe protegidas por travessia de diretórios, caminho SNMP, imagem de erro da configuração ou outro mecanismo que possa permitir tentativas de decifragem offline.

5.2.1 Controlos de criptografia web

Ref.	Política	Razão
63	Os cookies que armazenam ou são usados para aceder em confiança ou a dados de nível superior devem ser transportados com segurança.	Os cookies podem ser roubados através de diversos métodos, como XSS (scripts entre sites) e sniffing (intercetação de tráfego).
64	Conteúdos seguros e não seguros não devem ser misturados na mesma página.	Os conteúdos não seguros são potencialmente capazes de roubar informação segura do conteúdo.
65	Deverá usar o TLS para todas as páginas	A não utilização de TLS para início de sessão na página

	de início de sessão e todas as páginas autenticadas. A página de início de sessão e todas as páginas autenticadas subsequentes devem ser acedidas exclusivamente por TLS. A página de início de sessão e todas as páginas autenticadas subsequentes devem ser acedidas exclusivamente por TLS.	de destino permite que um atacante modifique a ação do formulário de início de sessão, fazendo com que as credenciais do utilizador sejam publicadas numa localização arbitrária. A não utilização do TLS para páginas autenticadas após o início de sessão permite a um atacante visualizar a ID de sessão sem encriptação, comprometendo a sessão autenticada do utilizador. A não utilização do TLS pode resultar num ataque “man-in-the-middle”.
66	Os clientes devem ser instruídos a não colocar dados sensíveis em cache.	O protocolo TLS fornece confidencialidade apenas para os dados em trânsito, mas não ajuda com possíveis problemas de fuga de dados do lado do cliente.
67	Use normas de assinatura digital aceites a nível nacional e internacional.	O uso de técnicas de assinatura digital não conformes com as normas poderá resultar numa confiança inapropriada na assinatura. A regulamentação nacional e internacional aplica controlos à assinatura digital, e resume a importação, exportação e controlos de criptografia domésticos em todo o mundo.
68	Não devem ser usados, em circunstância alguma, certificados de carácter universal.	Os certificados de carácter universal são um alvo muito apelativo. Podem certificar um servidor de forma não intencional e tornar um domínio de encriptação vulnerável. Caso um servidor ou subdomínio seja comprometido, todos os subdomínios podem ser comprometidos. Caso o certificado de carácter universal necessite de revogação, todos os subdomínios irão requerer um novo certificado.

5.2.2 Gestão de chaves geral

Ref.	Política	Razão
69	As chaves criptográficas para todas as novas implementações devem atender ou exceder as normas mínimas.	Chaves curtas ou fracas podem ser facilmente comprometidas durante o ciclo de vida dos dados.
70	Devem ser geradas chaves simétricas e assimétricas por meio de ferramentas e bibliotecas aprovadas.	Ferramentas e bibliotecas não aprovadas têm o potencial de gerar chaves fracas.
71	As palavras-passe que controlam a utilização de chaves criptográficas devem oferecer proteção equivalente à própria chave.	Palavras-passe fracas prejudicam a força das chaves criptográficas.
72	Um Gestor sénior da BT deverá ser responsável pela segurança das chaves.	Este deverá ter a antiguidade, a capacidade e a fiabilidade proporcionais à segurança dos dados que as chaves protegerão.
73	O Gestor de chaves sénior deverá certificar-se de que as responsabilidades da coluna de detalhes são atribuídas e levadas a cabo por pessoal devidamente formado e qualificado:	Responsabilidades Política de gestão de chaves Aquisição ou criação de chaves Conceção da distribuição de chaves, períodos das chaves, revogação e estrutura de responsabilização Criação de procedimentos de gestão de chaves Operação da gestão de chaves Proteção de chaves privadas e materiais relacionados

		Procedimentos de emergência, tais como, a revogação Auditoria de operações das chaves Recuperação de chaves Destruição de chaves

Controlo de documentos

Normas de orientação para terceiros do sector sobre “Codificação Segura”

Autor: Segurança da BT

Edição n.º 1, publicada em outubro de 2016