UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

ENGENHARIA DE SEGURANÇA

Projeto 1

Managing Security Risks Inherent in the Use of Third-party Components

---

Grupo 4

Autores:

Joel Gama (A82202)

Tiago Pinheiro (A82491)

23 de Março de 2020

Conteúdo

Introdução

O presente relatório surge no âmbito da Unidade Curricular de Engenharia de Segurança integrada no perfil de Criptografia e Segurança da Informação.

Este projeto é o primeiro de três projetos que ainda irão ser abordados nesta UC. O objetivo do projeto é investigar sobre o desenvolvimento seguro de *software*. Devido à existência de vários grupos de trabalho nesta UC foram introduzidos diferentes temas que iriam ser abordados por diferentes grupos. Ao nosso grupo de trabalho foi-nos atribuído o seguinte tema: *Managing Security Risks Inherent in the Use of Third-party Components*. Ou seja, os riscos do uso de *3rd party Software/Code* no próprio código.

Sobre este tema, nós decidimos abordar diversos aspetos. Vamos começar pela definição de *3rd party software* e falar das principais dificuldades causadas pelo uso de *TPCs*, com ênfase na segurança. Por outro lado, iremos falar das soluções existentes para a prevenção/eliminação do risco assim como enumerar as práticas seguras que se devem seguir quando se trabalha com *3rd party software*. Por fim, apresentámos algumas ideias de como irá ser o futuro nesta área.

O que é *Third Party Software* ?

Third Party Software é qualquer *software* (incluindo *object code, binary code, source code*, bibliotecas ou outro código, e incluindo Software comercial, *open-source* ou de uso gratuito), qualquer documentação ou material relacionado com o software em questão ou até qualquer derivação do que foi dito anteriormente que: não é unicamente propriedade da companhia ou suas subsidiárias e é incorporado em, distribuído com, requerido, necessário ou uma dependência para o desenvolvimento, uso ou comercialização de qualquer produto da Empresa em questão.

Third Party Software inclui:

- *Software* que é fornecido de alguma forma aos utilizadores finais da companhia, quer seja aplicando uma taxa ou sem custo, também se for distribuído ou *hosted* pela empresa ou até se estiver embebido ou incorporado no produto da empresa ou numa base *standalone*.
- Software que é utilizado para desenvolvimento, manutenção e/ou suporte de qualquer produto de companhias. Isto inclui, ferramentas de desenvolvimento como: compiladores, conversores, *debugers* ou *parsers*, ferramentas de *tracking* e bases de dados, etc.
- *Software* que é usado para gerar código

Principais dificuldades causadas pelo uso de *third party components*

Utilizando a definição anterior de *3rd party software components*, vamos agora identificar e exemplificar quais são as maiores dificuldades que aparecem depois de usar *third party software*.

Começando por um exemplo, imaginemos que uma determina empresa **Alpha** usa vários *TPCs* no seu produto para, desta forma, terem as funcionalidades que necessitam para o seu produto. Para este caso também se assume que a empresa não tem qualquer restrição ao uso de *3rd party components*. Sem que haja uma restrição ao uso destes surge um problema que irá afetar tudo o que vem a seguir, este problema é o controlo de *TPCs*.

3.1 Quais são os *TPCs* incluídos ?

Quando algum *3rd party component* é adicionado à lista de vulnerabilidades conhecidas como, por exemplo, a lista CVE, a empresa *Alpha* deve perceber se o *TPC* existe no seu produto, para depois conseguir corrigir o problema e evitar que os seus clientes sejam afetados pela vulnerabilidade. Com isto, surge a pergunta "Quais são os *TPCs* incluídos no meu produto ?".

Como, anteriormente, não existiu controlo algum no uso de *3rd party components*, a empresa não tem informação sobre quais e quantos *TPCs* está a usar. Para resolver este problema, a empresa tenta correr uma **ferramenta automática** de forma a detetar quais são os *TPCs* que estão a usar, assim como as suas versões, obtendo assim uma lista dos *TPCs* que estão a usar.

Infelizmente, a ferramenta pode não ser particularmente útil se o produto da empresa usar variadas linguagens de programação e *frameworks*, uma vez que é praticamente impossível a ferramenta abranger a totalidade das linguagens e *frameworks*. Para além disso, supondo que a empresa usa um número reduzido de linguagens e *frameworks* e, nesse caso, a ferramenta consiga executar com sucesso e obter uma lista com a totalidade dos *TPCs* presentes, a empresa irá ter de enfrentar mais uma problema: o **nome dos *TPCs***.

3.2 Nomes dos *third party components*

O problema do nome dos *TPCs* não é originado pela empresa *Alpha* , mas sim pela ausência de identificadores únicos para estes. Os seus nomes variam de empresa para empresa e são conhecidos por vários nomes: como pro exemplo o *Apache Xerces*, conhecido também por *Apache Xerces/J*.

Assim, não é propriamente fácil de determinar se o *TPC* está ou não em uso no projeto. Exigindo perder horas de trabalho e, portanto, custos monetários. Imaginando que a empresa consegue ultrapassar o problema do nome e perceber que o *TPC* existe no produto. Desta forma, surge um novo problema: as dependências do *TPC*.

3.3 Dependências dos *third party components*

O problema das dependências não tem que vir nesta ordem, ou seja, não existe uma ordem definida de quando os problemas aparecem. A questão das dependências é que um *TPC* pode usar vários *TPCs*, e esses usarem outros, e assim sucessivamente. Esta "árvore"de dependências é um obstáculo claro à detenção de *TPCs* num produto porque para além da empresa ter que se preocupar com os *TPCs* em si também tem que se preocupar com as dependências deste para que não existam vulnerabilidades nos seus projetos.

3.4 O produto foi afetado pela vulnerabilidade no *TPC* ?

Voltando à ideia inicial: existe uma vulnerabilidade num *TPC*, a empresa esta a averiguar se este *TPC* existe e a tentar corrigir o problema.

Neste caso, vamos pensar que a empresa **Alpha** conseguiu ultrapassar todas as dificuldades perceber que o *TPC* que foi identificado como tendo uma vulnerabilidade encontra-se no seu produto. Agora, é preciso verificar se esta vulnerabilidade afeta realmente o produto.

Por vezes, a informação no **CVE** não é detalhada ou clara o suficiente para a empresa conseguir perceber logo se o seu produto é afetado, sendo isto mais um obstáculo. Caso a informação seja pouco detalhada, será preciso realizar mais testes e perder mais horas a tentar solucionar o problema do que se a informação fosse mais detalhada. Esta fase, uma vez que é realiza manualmente esta sujeita a erros.

O que se pode fazer para prevenir o risco ?

As equipas de produção e desenvolvedores muitas vezes selecionam *TPCs* baseados apenas nas funcionalidades que eles procuram entregar aos utilizados, ignorando na maioria os riscos de segurança, suporte ou mantimento dos componentes. Desta forma, fazendo uma seleção tendo a segurança em consideração pode salvar bastante tempo posteriormente, para além de não colocar os utilizadores em risco.

4.1 Quais os *TPCs* não devemos utilizar e os riscos associados a eles?

Alguns *TPCs* normalmente não são implementados considerando a segurança, resultando em riscos que podem afetar produtos ou serviços que os utilizem. Considerando isso, alguns exemplos de *TPCs* que podem carregar maiores níveis de riscos de segurança são:

- Componentes criados por estudantes em graduação, desenvolvidos sem considerações de segurança de software. Pois na maioria das vezes, depois de terminado, os componentes são abandonados pelo autor e nunca mais são alterados.
- Código fonte disponível no *GitHub* com instruções de compilação que utilizam compiladores *outdated* e com a última atualização feita a mais de 4 anos.
- Código desenvolvido como *hobby*. Normalmente este código foi desenvolvido em experiências de programação ou numa fase de aprendizagem e está disponível no site ou autor.

Apesar destes componentes fornecerem as funcionalidades desejadas, eles podem trazer um nível de risco inaceitável para a organização. Especialmente quando os produtos desenvolvidos são integrados em funções críticas das empresas ou infraestruturas. Assim, selecionar os componentes tendo em mente a segurança para além das funcionalidades vai ajudar a diminuir o risco de segurança da organização. Em suma, as organizações que integram *TPCs* têm de ter noção dos desafios e implementar um ponto de vista balanceado entre segurança e as funcionalidades que pretendem oferecer para uma melhor gestão do risco.

4.2 O que devemos fazer para manter os *TPCs* no nosso produto?

A manutenção de *TPCs* usados ou incorporados num produto é uma tarefa importante durante todo o ciclo de vida de um produto. Esta manutenção inclui a resposta a vulnerabilidades que existam ou surjam em *TPCs* usados no produto. Porém, esta tarefa não é tão fácil de ser realizada, uma vez que é necessário alguma preparação para que seja executada, pois sem um plano ou estratégia para alguns problemas, podem surgir riscos.

Algumas estratégias ou planos para os problemas são:

- Fazer rastreamento das fraquezas e vulnerabilidades de segurança - verificar quais as vulnerabilidades e fraquezas existentes para todos os *TPCs* como forma de reconhecer possíveis pontos de risco.
- Fazer monitorização e *updates* - manter a informação sobre novos riscos atualizada e corrigir os riscos existentes ou que sejam descobertos.
- Verificar *TPCs* não utilizados - listar quais são os *TPCs* não utilizados que estão presentes no produto (quer tenham sido adicionado por erro no desenvolvimento ou porque deixaram de ser utilizados) e remove-los.
- Verificar os *TPCs* descontinuados e que não recebem mais suporte - remover as versões dos *TPCs* que tenham vulnerabilidades e que estejam ou descontinuados ou sem suporte.

4.3 O que é um clico de vida de um produto?

Um ciclo de vida de um produto são o conjunto de passos pelos quais o produto passa durante o seu desenvolvimento.

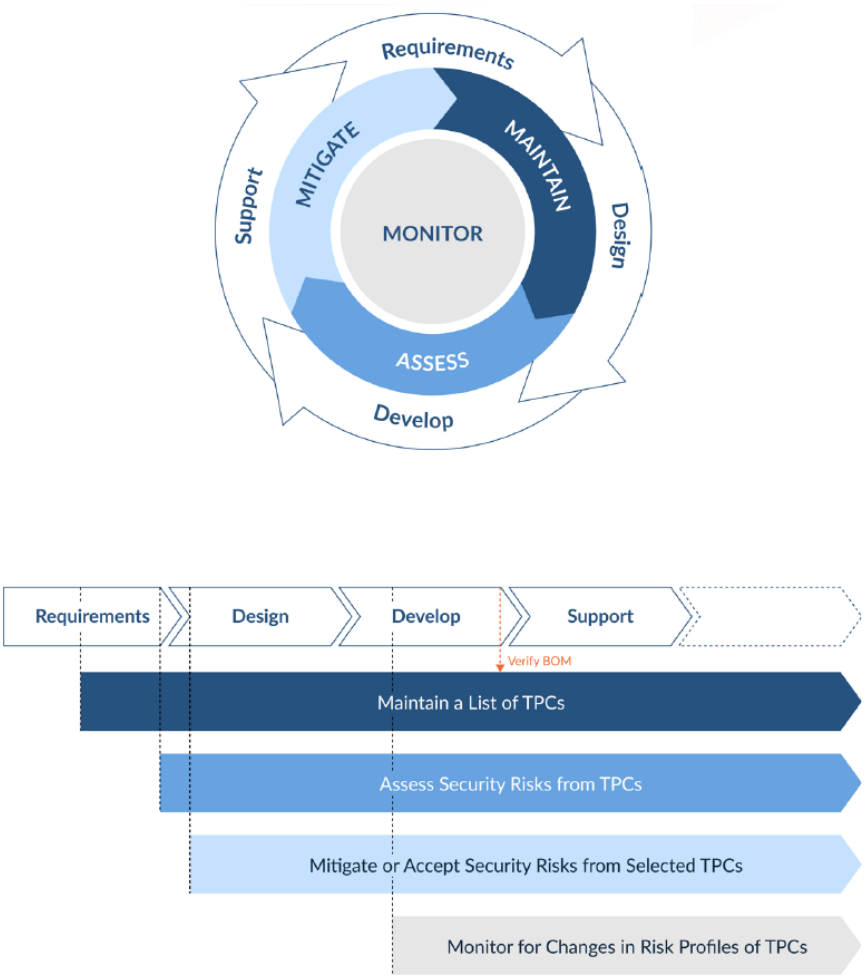

Figura 4.1: Esquema de um Ciclo de vida de um produto.

O primeiro passo é a definição de requisitos. Neste são definidos os requisitos e é quando deve ser definida uma lista dos *TPCs* que possam ser utilizados em função dos requisitos. O passo seguinte é o design. Aqui são feitas escolhas do que deve ser implementado. As escolhas para os *TPCs* deve passar pela análise dos riscos associados a cada *TPC*. Depois do design vem o desenvolvimento. É no desenvolvimento que são implementadas as decisões tomadas nos passos anteriores e é também no desenvolvimento que se inicia a monitorização dos perfis de risco dos *TPCs*. A monitorização dos perfis de risco dos *TPCs* consiste em verificar alterações nas vulnerabilidades e possíveis falhas que os *TPCs* possam ter e quais as "áreas"do produto que são afetadas. Por fim, o suporte é onde é feita a verificação dos perfis de risco dos *TPCs*. É verificado se foram reportadas novas vulnerabilidades ou falhas e são corrigidas em versões posteriores do produto.

Considerações Futuras

Atendendo aos riscos e desafios que foram enunciados anteriormente é evidente que existem mudanças a ser feitas para melhorar o panorama global dos *third party components*. O esforço requerido para esta mudança tem que ser coletivo, isto é, não pode ser realizado por apenas uma entidade ou companhia. Para obter sucesso neste aspeto também é necessário que as medidas tomadas sejam aceites e usados pela maior parte (ou totalidade) das empresas.

Nos próximos capítulos irão ser apresentadas as medidas que deviam ser tomadas.

5.1 Nome dos *third party components*

Como já foi dito e explicado anteriormente, a pluralidade de nomes de *TPCs* é um grande problema. A primeira medida passa por aí: criar uma base de dados onde estivessem contidos os *TPCs* existindo um único nome para cada um. Obviamente, isto é muito difícil de conseguir, mesmo envolvendo toda a gente devido à quantidade enorme de *TPCs* que existem. Tal que, primeiro se deviam começar com os *TPCs* mais conhecidos e usados e, só depois, ir adicionando os restantes.

Para além do nome, também seria necessário incluir alguma informação sobre os *TPCs*. Quanto maior a quantidade de informação sobre o *TPC* mais fácil é prevenir os riscos e encontrar soluções para os problemas que possam surgir.

5.2 *End-of-life Repository*

Assim como deve ser criada uma base de dados para os nomes dos *TPCs* essa mesma base de dados, ou outra, também deve ser usada para ter as datas de *validade* de um *third party component*, isto é, quando o *TPC* vai deixar de ser suportado pelo seu vendedor. Com este tipo de informação online e atualizada, a vida das empresas fica muito mais facilitada, evitando a perda de horas e mão-de-obra a procurar estas informações.

5.3 *Vulnerability Source Listing*

Mais uma vez, a forma como as vulnerabilidades são expostas e onde são expostas também é importante. Se existiram muitas *databases* de vulnerabilidades é natural que algumas vulnerabilidades em certos produtos nem sejam detetadas (pelos desenvolvedores desse produto).

Com isto, se a cada *TPC* fosse associado a vulnerabilidade, numa base de dados conjuntos, ficávamos mais próximos da solução do problema atendendo a que a informação seria mais facilmente consultada.

Conclusão

Este trabalho foi interessante para explorar o tema de desenvolvimento seguro de software ou, pelo menos, uma pequena parte deste tema bastante abrangente.

Através deste projeto foi possível aprender e interiorizar quais são os riscos e consequências provinientes do uso de *third party components*. Por outro lado, também foi possível ficar a saber mais sobre quais as boas práticas (práticas seguras) que se devem seguir, assim como, ficar com uma ideia do que irá ser o futuro neste campo, se tudo correr como o planeado.

Em suma, este projeto foi um meio bastante interessante de expor uma situação ao alunos e o grupo sente que foi uma maneira muito eficaz para adquirir conhecimento.