Relatório TP3 - Desenvolvimento Seguro de uma Aplicação

Roberto Cachada (a81012) Pedro Gonçalves (a82313)

Braga, julho de 2020

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Engenharia de Segurança
(2º Semestre/2019-2020)
Grupo 7

Conteúdo

1 Introdução

Tendo por base os trabalhos práticos anteriores, onde foram estudadas formas seguras de desenvolvimento de *software* e também ferramentas/testes para avaliar a qualidade do *software* criado, foi pedido ao grupo o desenvolvimento de uma aplicação comando linha que permitisse testar as operações do serviço SCMD (*Signature* CMD), tendo por base os tópicos estudados nos trabalhos anteriores.

Assim sendo, neste relatório será descrito todo esse processo de desenvolvimento seguro da aplicação.

2 Estrutura da Aplicação

Para se conseguir perceber quais os principais riscos de segurança, é necessário em primeiro lugar perceber qual a estrutura da aplicação a criar. Assim sendo, podemos dividir esta aplicação em duas áreas.

A primeira área, consistiria na parte da aplicação que trataria da comunicação com o serviço SCMD, ou seja, seria a área onde estariam implementadas as operações do serviço. Já a segunda área, consistiria na parte da aplicação que trataria da interação com o utilizador, usando neste caso uma interface de comando linha.

No capítulo seguinte, será feita uma análise do risco de segurança da aplicação, e dos mecanismos a implementar de forma a prevenir esses riscos.

3 Análise Risco de Segurança

Tendo por base a estrutura da aplicação é possível perceber, como foi referido no capítulo anterior, que a mesma se divide em duas partes, logo a análise do risco de segurança será feita de forma individual para cada uma dessas, de modo a abranger o máximo da aplicação possível.

Começando por analisar a área responsável pela interação com o utilizador, torna-se fácil perceber que o maior risco existente nesta área consiste na correta validação de input. Assim sendo, e tendo por base o controlo número 5 do *OWASP Top 10 Proactive Controls* [1], é necessário garantir que todos os inputs do programa sejam validados de modo a verificar se possuem o tipo, quantidade e se estão na estrutura pretendida para o mesmo. Caso esta verificação seja corretamente executada, é eliminado o maior risco existente na área responsável pela interação com o utilizador.

Já na área de comunicação com o serviço SCMD, é necessária uma maior cautela, pois existem mais sítios suscetíveis a ataques nesta área. Em primeiro lugar, é necessário criar a estrutura de *payload* seguindo o esquema existente no ficheiro WSDL, de modo a que não sejam permitidos atributos não aceites pelo servidor. Também no âmbito do *payload*, é necessário proceder à validação dos mesmos, de modo a garantir que possuem as características esperadas. Por fim, é também necessário verificar que a *response* recebida segue os parâmetros estabelecidos na documentação do serviço SCMD. Esta última parte necessita de ser executada antes da utilização dos dados recebidos para verificar se não ocorreu algum tipo de *tampering* durante o processo de resposta.

Assim sendo, os casos enumerados nos parágrafos anteriores representam

os maiores riscos à segurança da aplicação, sendo por isso necessário implementar os "contra-feitiços", descritos de forma breve neste capítulo, de modo a eliminar esses mesmos riscos.

4 Ferramentas

Antes de iniciar o processo de desenvolvimento é necessário decidir quais as ferramentas que serão utilizadas para construir a aplicação. Em primeiro lugar, convém referir que a linguagem de programação utilizada para o desenvolvimento da aplicação foi o Golang (Go). A partir daqui, torna-se possível identificar quais as diferentes bibliotecas a utilizar de modo a implementar as funcionalidade pretendidas.

Tendo por base novamente o documento da OWASP [1], este refere que devem ser utilizadas bibliotecas e *frameworks* que tratem da implementação das funcionalidades pretendidas ao invés da criação por parte do utilizador das suas próprias bibliotecas. Tendo isso em conta, o grupo tentou escolher bibliotecas mantidas pelos desenvolvedores da linguagem, ou quando tal não fosse possível, escolheu bibliotecas *open-source*. Assim sendo, serão de seguida listadas as bibliotecas escolhidas pelo grupo para auxiliar na implementação do projeto:

- github.com/spacemonkeygo/openssl - *Wrapper* de OpenSSL para Go;
- github.com/akamensky/argparse - Biblioteca inspirada na biblioteca *argparse* da linguagem Python;
- bytes - Biblioteca que implementa funções para manipulação de *slices* de bytes;
- crypto - Biblioteca que possui as mais comuns constantes criptográficas;
- encoding/base64 - Biblioteca que possui as mais comuns formas de *encoding*;
- net/http - Biblioteca que permite interagir com os servidores;
- strings - Biblioteca que implementa funções para manipulação de UTF-8 encoded *strings*.

Após selecionadas as bibliotecas a utilizar, o grupo passou à fase da implementação da aplicação.

5 Implementação, Testes e Dificuldades

Esta etapa foi a que criou maiores dificuldades ao grupo. Em primeiro lugar, o facto de nenhum elemento do grupo possuir qualquer tipo de experiência com a linguagem a utilizar juntando ao facto de ambos os elementos se encontrarem sobrecarregados com projetos de outras cadeiras, levou a que não fosse possível implementar todas as funcionalidades da aplicação. Outra situação que causou dificuldades foi o facto de nenhum dos elementos do grupo possuir chave móvel digital, o que impossibilitou a realização de testes ao código desenvolvido.

Falando da implementação em si, o primeiro passo foi a criação das funções que permitissem executar o protocolo SOAP. Devido ao facto de o grupo não ter encontrado nenhuma biblioteca que tratasse da execução do protocolo de *request* de forma automática, tal como acontece na aplicação fornecida pelo professor, foi necessário fazer tudo de forma manual, o que tornou o processo mais trabalhoso e mais sujeito a erros. Em primeiro lugar foi preciso perceber qual a estrutura do *payload* a enviar e da resposta do servidor. Para tal, foi utilizada um extensão do Chrome, Wizdler, que nos permitia dado um ficheiro WSDL descobrir qual as ações e *endpoints* do mesmo e também do método HTTP suportado pela operação de SOAP. A partir dessa informação, simplesmente foi criada a estrutura do *payload*, tendo cuidado para validar os dados fornecidos ao mesmo, e efetuado um pedido html para o servidor dando o *payload* como input. De notar que é efetuado controlo de erros, para "apanhar"possíveis erros nas respostas aos *requests*.

Depois de implementadas as funções que tratavam dos diferentes pedidos SOAP, o passo seguinte teria sido a criação de um ficheiro que permitisse o teste das diferentes funções. No entanto devido à falta de tempo tal não foi possível.

Por fim, o grupo pensou em avaliar o *Fan-Out* da aplicação criada, para tentar perceber qual a sua dependência de módulos externos e internos. Infelizmente, devido à ferramenta usada para fazer este tipo de avaliação ser paga, *TICS framework*, não foi possível executar essa avaliação, mas teria sido interessante para compreender a interdependência entre os diferentes componentes.

6 Conclusão

Com a elaboração deste projeto, mesmo considerando que o resultado final não foi o que seria esperado pelo grupo, considerámos que ficámos com uma melhor ideia dos vários passos necessários para o correto desenvolvimento seguro de uma aplicação, o que juntando aos conhecimentos obtidos nos projetos anteriores forma uma boa bagagem para o nosso futuro profissional.

Em relação às funcionalidades não implementadas, mesmo com a opção de entrega até ao dia 13/7, seria impossível ao grupo implementar as mesmas, pois os elementos do grupo encontram-se completamente sobrecarregados com trabalhos para outras cadeiras.

Em tom de resumo, mesmo considerando que o resultado final foi abaixo do razoável, foi impossível ao grupo fazer melhor devido à situação do semestre em que ambos os elementos do grupo se encontram.

Referências

[1] Anton, K; Manico, J; Bird, J.: OWASP Top 10 Proactive Controls V3 (2018)

[2] Miranda, J.: Material de apoio à disciplina (2020)