CMD-SOAP - Teste das operações do serviço SCMD

Henrique José Carvalho Faria nº82200

Departamento de Informática, Universidade do Minho

Resumo Neste trabalho foi-nos pedido que emulássemos o programa *CMD-SOAP* desenvolvido pelo professor em *Perl*. Adicionalmente este programa deve aplicar medidas de segurança que previnam diversos ataques que podem ser realizados por idividuos mal intencionados. Assim, começamos por emular o programa, seguindo-se um estudo das vulnerabilidades mais prováveis de ocorrerem em programas bem como das vulnerabilidades mais conhecidas de programas *Perl*. Assim este relatório começa por descrever o processo de emulação da aplicação e posteriormente realiza uma sumula das vulnerabilidades possíveis e de que forma foram tratadas.

Palavras-chave: Perl · Buffer-Overflow · String Vulnerabilitys · Integer Vulnarabilitys · Input Validation · white List

1 Criação da aplicação CMD-SOAP em Perl

Para este trabalho seguiu-se a estrutura do programa original separando o corpo da aplicação, que ficou no ficheiro test_cmd_wsdl.pl, das operações sobre ficheiros, chave móvel e ligação ao servidor, que ficaram no modulo cmd_soap_msg.pm, e da variável APPLICATION_ID que ficou no modulo cmd_config.pm. Adicionalmente foi criado um módulo para tratamento de segurança da aplicação chamado *verifiers.pm*.

Para o corpo do programa o Perl começa por importar a variável *$APPLICATION_ID* do modulo *cmd_config* através da subrotina *get_appid()*. Caso esta variável não esteja definida o programa termina.
Em seguida é verificado se o programa foi invocado com argumentos, caso contrário o programa também termina. Caso tenha sido invocado com argumentos é realizado um parser dos dados recorrendo ao módulo *Getopt::Long* que faz uso de flags para identificar inequivocamente cada variável recebida como parâmetro. Em seguida estes argumentos são colecionados num array cujas variáveis definidas serão verificadas fazendo uso das subrotinas pertencentes ao módulo *verifiers.pm*.
Caso o input passe nas verificações de segurança, o pedido do cliente é passado ao módulo *cmd_soap_msg.pm* e dependendo do tipo de operação pedida pelo utilizador é chamada a respetiva subrotina. Caso o utilizador pretenda testar todo o programa pode usar a operação *test* que corre todas as subrotinas por ordem de forma a realizar a assinatura com a chave móvel digital sobre o ficheiro fornecido.
Convêm referir que, ao contrário do que foi realizado no ficheiro original *test_cmd_wsdl.py* as verificações de segurança referentes ao código das mensagens recebidas do servidor foram realizados no módulo *cmd_soap_msg.pm*.

1.1 Módulos

Para instalar os módulos necessários pode-se utilizar uma ferra menta chamada *cpanm*. Pode-se descarregar esta ferramenta para linux com o comando de terminal *sudo apt install cpanminus*.
Após instalar a ferramenta deve-se garantir que se tem os seguintes módulos instalados[1]:

1. XML::Compile::WSDL11
2. XML::Compile::SOAP11
3. XML::Compile::Transport::SOAPHTTP
4. Encode
5. Bit::Vector

[1] Nota: Para descarregar os módulos use no terminal o comando cpanm install 'nome do módulo'

6. Digest::SHA
7. HTTP::Request
8. HTTP::Parser
9. MIME::Base64
10. File::Basename
11. File::Slurp
12. Try::Tiny
13. Getopt::Long
14. Switch
15. File::Basename
16. Crypt::OpenSSL::X509
17. Crypt::PK::RSA
18. Crypt::Misc
19. Carp::Assert

2 Vulnerabilidades Gerais

No desenvolvimento de um software devemos sempre garantir que não divulgamos informação sobre como a nossa aplicação está construida. Durante o deenvolvimento do código deparamo-nos com dois possíveis problemas a tratar.

O primeiro problema enuncia-se em seguida, "Como encerrar a aplicação com uma exceção passando uma mensagem de erro ao utilizador sem lhe revelar informação sobre o código da aplicação?". Para resolver este problema usamos a função *die*, o problema é que esta para além da mensagem fornece informação sobre a linha onde ocorreu a exceção. Felizmente caso se adicione
n ao final da mensagem de erro emitida pelo die este omite a informação referente á linha.

O segundo problema encontrado é referente á função *open*, até ao ano 2000, a função open usava 2 parâmetros, um para a variável para a qual se lê e uma para o ficheiro a ler. O problema acontece caso o utilizador use um ficheiro cujo nome comece, por exemplo, com o sinal *>*, isto levará a que por exemplo caso seja dado como input o ficheiro *>/etc/passwd* nós acabamos de apagar o ficheiro de passwords do Linux. Para resolver este problema usamos a versão do open com 3 variáveis, uma para guardar a informação a ler do ficheiro, uma para o tipo de leitura a realizar no ficheiro e uma para o nome do ficheiro. Acresce a este problema o facto de que caso o open use um pipe em vez de um ficheiro, ao falhar este devolve o pid do subprocesso na mensagem de erro, como queremos evitar divulgar qualquer informação sobre a aplicação usamos então a função *die* para emitir o erro sem comprometer a nossa implementação tomando o código a seguinte forma: *open(variável para leitura, modo de leitura, ficheiro a ler) or die ...* .

Nota: As restantes questões de segurança foram abordadas e tratadas num modulo perl a parte chamado ***verifiers.pm*** *criado para separar de forma legivel as subrotinas usadas para segurança das subrotinas do programa principal.*

Existem enúmeras vulnerabilidades a tratar para além das duas supramencionadas, nomeadamente:

1. Restrições sobre a memória
2. Neutralização do input durante a geração da página web
3. Improper Input Validation
4. Information Exposure
5. Out-of-Bounds Read
6. Neutralização de elementos especiais para comandos SQL (SQL Injection)
7. use after free
8. Integer Overflow or Wrapparound
9. XML Injection
10. OS Commands Injection
11. SQL Injection

Destas vulnerabilidades a 1ª 4ª, 5ª, 6ª, 7ª e 8ª podem ser ignoradas visto que o *Perl* trata da alocar as variáveis na memória libertando o programador da manutenção da mesma, para além. Assim vamos debruçar-nos sobre a vulnerabilidade 3, 9,10 e 11.

2.1 Improper Input Validation

Nesta secção falaremos um pouco dos inputs e da validação realizada sobre os mesmos. A validação de inputs de uma aplicação é fulcral para o bom funcionamento da mesma, nunca devemos acreditar que o utilizador usará a aplicação da melhor forma ou para fins nefastos.
Assim os inputs a verificar são: o nome do ficheiro, o número de telefone, o pin, o OTP e o process Id.

- Nome do ficheiro
  Para realizar a verificação do nome do ficheiro foram criadas 2 listas, uma white list com os caracters aceitaveis para constituirem o nome de um ficheiro (As White Lists são especialmente proveitosas visto que é mais facil indicar o que é aceitavel do que o que não é, em contrapartida limitamos um pouco os nomes possíveis para os ficheiros fornecidos) e uma black list onde removemos algumas hipóteses aceitáveis na white list mas que não podem ser dados como input do nome do ficheiro que são as flags usadas nos inputs do programa.
  Convem notar que tentativas de inserção de vários comandos através da adição de *;* ou de pipes com o caracter / não funcionam pois não pertencem á lista de caracters permitidos pela white list.

- Número de telefone
  No caso do número de telefone desenvolvemos 2 regex sendo que um funciona para números internacionais e nacionais e um que funciona apenas para números nacionais, os respetivos regex apresentam-se em seguida:

  - */^\+[0-9]{1,3} [0-9]{4,14}$/*
  - */^\+351 [0-9]{9}$/*

  Como a chave móvel digital para a qual a aplicação se destina normalmente está associada a números de telemovel portugueses mantivemos o segundo regex embora tenhamos deixado em comentário o segundo regex caso pretendamos estender a aplicação a números estrangeiros. A razão de escolhermos apenas números nacionais prende-se com a esolha de implementar uma segurança com granularidade mais fina visto que o número de dígitos de telemovel varia de país para país e o indicativo também.
  Nota: Convem notar que, nas expressões regex, são usados por vezes 2 simbolos, o ^ no inicio do regex e o $ no fim. Estes simbolos indicam ao perl

que o regex tem de corresponder ao inicio do input e ao final deste respetivamente, isto é, caso ambos os simbolos sejam usados o perl entende que o input a testar tem de ser totalmente formado pelo regex e, caso não seja, falha a verificação.

- PIN

  O PIN é um conjunto de 4 digitos, assim, para o testar bastou um regex simples que garantisse isso: */^[0-9]{4}$/*.

- OTP

  O OTP é verificado como endo um conjunto de 6 digitos. Mais uma vez o regex usado é bastante simples: */^[0-9]{6}$/*.

O principal foco da segurança na nossa aplicação foi aplicado ás strings. Os critérios usados na *white List* e na *Black List* não são muito restritivos, principalmente porque as strings são maleáveis e o utilizador pode dar o nome que quiser ao documento que pretende utilizar com a aplicação. Desta forma é necessário ter atenção a utilizadores mal intencionados que pretendam usar os critérios laços de filtragem de input para fins diferentes daquele para o qual a aplicação foi feita.

2.2 OS Injection

O Perl é relativamente suscetivel a injeção de comandos do sistema operativo, isto pois permite utilizar pipelines com o caracter / como input ou comandos seguidos separados pelo caracter *;*. Para ambos os casos existe uma solução que apesar de restringir a liberdade do cliente de nomear os seus ficheiros recorrendo aos caracters / e / garante que estes ataques não ocorrem, o que do ponto de vista de uma maior qualidade na segurança da aplicação é o ideal.

2.3 XML Injection

Nesta subsecção vamos tratar de qualquer tentativa de injeção de código *XML* na nossa aplicação. Para lidar com tentativas de injeção de código *XML* através de um input foi criada uma subrotina chamada xmlInjection no modulo *verifiers.pm*.
Nesta subrotina para evitar eventuais tentativas de injeção de código *XML* foi realizado um regex com a forma: */<[a-zA-Z]*(>[^ (<\/)]*</[a-zA-Z]*|\/)?>/* . Este regex permite realizar match entre elementos desta linguagem através de sintaxe conhecida detetando padrões como *<qualquer coisa> ... </qualquer coisa>* ou *<qualquer coisa/>*.

2.4 SQL Injection

Para tratar eventuais tentativas de injeção de código *SQL* através das variáveis, foi definida uma subrotina chamada *sqlInjection* que possui um array de palavras chave usadas na syntax *SQL* que são comparadas através de um regex com os argumentos. Caso seja detetado num argumento uma palavra pertencente á syntax *SQL* o programa emite uma mensagem de erro a avisar que detetou uma tentativa de SQL injection.

3 Certificados Falsos

Um problema quando se lida com certificados prende-se com a validação dos mesmos. De forma a contornar este problema recorreu-se ao módulo *LWP::UserAgent*, este fornece uma opção por defeito de verificação automática do servidor e da sua legitimidade chamada *verify_hostname*. Assim são escolhidos protocolos seguros e é assegurado que nos ligamos a um servidor que possui um certificado válido.

4 Como correr o programa

Para correr o programa desenvolvido temos várias opções. Para saber qual a sintaxe pela qual o programa se rege deve-se chamar o programa com um argumento ou sem argumentos escrevendo no terminal uma das seguintes opções:

- perl test_cmd_wsdl.pl
- perl test_cmd_wsdl.pl -h

Caso se pretenda saber quais as opções de input disponíveis para cada operação devemos inserir uma das seguintes opções:

- perl test_cmd_wsdl.pl -o test -h
- perl test_cmd_wsdl.pl -o gc -h
- perl test_cmd_wsdl.pl -o ms -h
- perl test_cmd_wsdl.pl -o mms -h
- perl test_cmd_wsdl.pl -o otp -h

Para testar as várias operações basta inserir um dos seguintes comandos no terminal, adicionalmente pode-se escolher se se usa a opção -prod ou não com cada comando:

- perl test_cmd_wsdl.pl -o test -f ../LICENCE -u "user phone number' -p 'your pin'

- perl test_cmd_wsdl.pl -o gc -otp 'OTP received in your device' -procId 'ProcessID received in the answer of the CCMovelSign/CCMovelMultipleSign command'

- perl test_cmd_wsdl.pl -o ms -u 'user phone number' -p 'your pin'

- perl test_cmd_wsdl.pl -o mms -u 'user phone number' -p 'your pin'

- perl test_cmd_wsdl.pl -o otp -otp 'OTP received in your device' -procId 'ProcessID received in the answer of the CCMovelSign/CCMovelMultipleSign command'

Caso corra a aplicação com o comando *test* o resultado esperado deve ser o seguinte:

```
test Command Line Program (for Preprod/Prod Signature CMD (SOAP) version 1.6 technical specification)
  version: 1.0

+++ Test All inicializado +++
 0% ... Leitura de argumentos da linha de comando - file: .file user: .user pin: .pin
10% ... A contactar servidor SOAP CMD para operação GetCertificate
20% ... Certificado emitido para Henrique José Carvalho Faria pela Entidade de Certificação EC de Chave Móvel Digital de Assinatur
a Digital Qualificada do Cartão de Cidadão 00003 na hierarquia do  Cartão de Cidadão 006
30% ... Leitura do ficheiro ../LICENSE
40% ... Geração de hash do ficheiro ../LICENSE
50% ... Hash gerada (em base64): OXLcl0T2SZ8Pmy2/dmlvKuetivmyPd5m1q+Gyd+zaYY=

60% ... A contactar servidor SOAP CMD para operação CCMovelSign
70% ... ProcessID devolvido pela operação CCMovelSign: cea92d12-fa93-4224-b6a7-0e211239d36a
80% ... A iniciar operação ValidateOtp
Introduza o OTP recebido no seu dispositivo: 809788
90% ... A contactar servidor SOAP CMD para operação ValidateOtp
100% ... Assinatura (em base 64) devolvida pela operação ValidateOtp: YNXrGahGlev45vPP0Sa05Dany07mhXojwZRmEo2X2b1EDc08Vbz1p8xnowd/
S6+LABl+zTHfhHEyXqF/j6y4+G14eCB1NHOav1IGdEs61mels66K8BMxuN61vmbwhwlcFA4KBBRn864nLkv3u+Gz36Kuq/ncOTW5EsY4/PDGOnN7Zvh5Si8b/zgCv5Hw4G
fRbv5oxgi96spvVEBQYumpV8EquFJdU7rwdGb8OoDBbMlJw7BiYJmFiD1h7ekVR1KjlJnxclFAyWXy1U2leqFj8gp9La1+3oA5wnKLF/h+U4HgSmJGalH6xaL67KGnNm5R
Tlfd2K9mJmSEehTrqarWgwNoivxRpTgGTiHkMIg9zxkzYvAUXGFm+JBT5c10h6hMuHw4fLt6GdeXPZOkL+Lis0vybjELzNcIWOQLr5KdwH95awgPaEF05mlTmtMHDoAw7g
oacq3j0xsnylKZkDqPI1LIMhWMAw0bwsJoHSOnTGw5iGHWJZ3uNclI01JtB7+n
110% ... A validar assinatura ...
Assinatura verificada com sucesso, baseada na assinatura recebida, na hash gerada e na chave pública do certificado de Henrique Jo
sé Carvalho Faria

+++ Test All finalizado +++
```

Figura 1: Resultado de correr o comando perl test_cmd_wsdl.pl -o test -f 'file name' -u 'user phone number' -p 'your pin' com ou sem a opção prod