Unidade Curricular de Segurança de Redes
Mestrado em Segurança Informática
Mestrado em Engenharia de Redes e Sistemas Informáticos

PROOF OF CONCEPT V2V COMUNICATION

Cláudia da Costa Maia
up201905492@up.pt

Edson Aires Júnior Ferreira
up201607407@up.pt

Fabiana Manuela Alves
up201404791@up.pt

***Abstract* — As comunicações *Vehicle-to-Everything* (V2X) foram desenvolvidas com o intuito de reduzir a ocorrência de acidentes e controlar o trânsito. No seguimento do projeto anterior, aprofundamos neste relatório as comunicações *vehicle-to-vehicle*. Assim demonstramos as comunicações entre veículos recorrendo a vários simuladores *e frameworks*.**

Este relatório tem a seguinte estrutura: Contextualização (I); Arquitetura e design da PoC (II); Flood attack(III);Detalhes da implementação (IV); e por fim a avaliação e validação do setup, resultados e conclusão (V).

I. CONTEXTUALIZAÇÃO

As *V2V communications* que vamos demonstrar usam um protocolo wireless, semelhante ao wi-fi, chamado *dedicated short range communications* (DSRC) para enviar mensagens entre veículos. Quando este protocolo é combinado com a tecnologia GPS resulta num sistema de comunicações V2V *low cost* que providencia um conhecimento dos veículos similarmente equipados dentro do alcance das comunicações (mais de 300 metros).

Estas mensagens transmitidas pelos veículos incluem as informações de controlo que são: estado de transmissão, estado do travão, ângulo do volante, histórico do caminho do veículo e previsão do trajeto que vai seguir. Esta tecnologia, alerta os condutores com mensagens visuais, táteis ou auditivas, ou uma combinação das mesmas, habilitando assim o condutor de prevenir colisões.

Tal como em quase todas as tecnologias, existem vários ataques de segurança que podem ocorrer nas comunicações V2V, sendo estas: *flood, modify, read, spoof e replay attacks*. Assim sendo, é necessário garantir a integridade e disponibilidade dos dados, autenticação e ainda garantir que as mensagens são atuais e não repetições de eventos passados.

O ataque que exploramos é o *flood* e consiste na inundação da rede com menagens erróneas. Para isto utilizamos um simulador, em simultâneo com outras *frameworks*, de mensagens entre veículos (OMNeT++, INET, Veins e SUMO).

II. ARQUITECTURA E DESIGN DA POC

Para simular as comunicações V2V utilizamos várias bibliotecas e *frameworks* em simultâneo, sendo estas o OMNeT++, o Veins, o INET e o SUMO.

O OMNeT++ é um simulador de eventos discreto e serve principalmente para construir simuladores de redes, sendo ainda uma biblioteca e *framework*, extensível, modular e baseada em C++.

Veins é uma *framework open source* para correr simulações de redes de veículos, ou seja, para simular as *V2V communications*. Este inclui um conjunto abrangente de modelos para tornar as simulações o mais realista possível, sem comprometer a velocidade. A GUI e IDE do OMNeT++ e do SUMO podem ser usadas facilmente para fazer o setup e correr interactivamente as simulações.

O INET *framework* é uma biblioteca de modelos *open-source* para o ambiente OMNeT++ que providencia protocolos, agentes e outros modelos para os utilizadores poderem trabalhar com as redes de comunicação. É especialmente útil para desenhar e validar novos protocolos, ou explorar novos cenários. Esta *framework* contém modelos para protocolos como: Ethernet, PPP e IEEE 802.11, sendo o IEEE 802.11 um dos principais protocolos descritos no nosso primeiro projeto. Em adição, o INET serve de base a várias simulações como, *vehicular networks, overlay/peer-to-peer networks*, ou LTE.

O SUMO é usado para simular o tráfego rodoviário e inclui a simulação de estradas, peões e transportes públicos.

Tal como podemos verificar na figura 1 a simulação do tráfego rodoviário é feita pelo SUMO, enquanto o controlo da simulação e eventos é feito pelo OMNeT++ e o Veins simula as comunicações V2V.

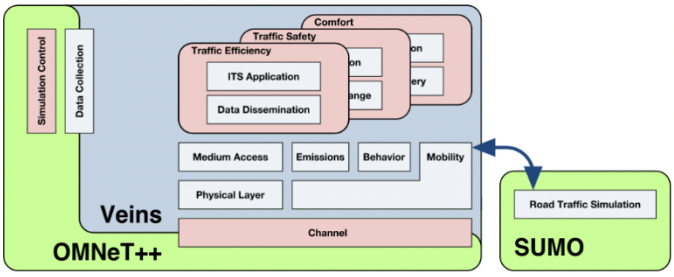

Figura 1. Simuladores Veins, OMNeT++ e SUMO

III. FLOOD ATTACK

Os *Flood Attacks* também conhecidos por *Denial of Service (DoS) attacks* consistem no envio de volume de tráfego altamente elevado para um determinado sistema até ao ponto de o mesmo não conseguir examinar e permitir qualquer tipo de tráfego na rede. Uma forma bastante comum deste tipo de ataque é *o SYN flood attack* que faz alvo a qualquer tipo de sistema conectado à internet e que providencie serviços do *Transmition Control Protocol (TCP)*. Este tipo de ataque procura consumir a conexão das tabelas de estado presentes em diversas componentes de infraestrutura, tal como balanceadores de carregamento, *firewalls, Intrusion Prevetion Systems (IPS*) e a servidores de aplicação em si. Este tipo de ataque é capaz de desabilitar até dispositivos com capacidade de manter milhões de conexões em simultâneo.

O *SYN flood attack* ocorre quando a camada de TCP se encontra saturada, impossibilitando a conclusão do TCP *three-way handshake* entre cliente e servidor em todas as suas portas.

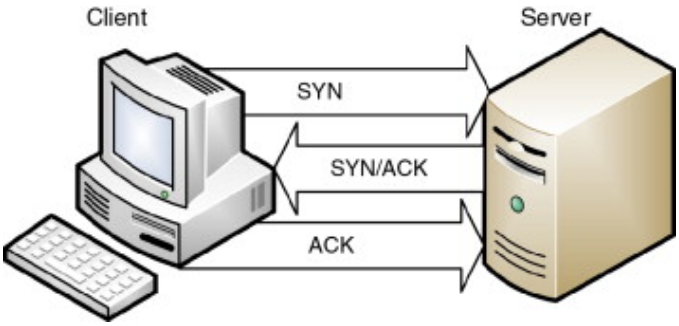

Figura 2. Three-way-handshake

Todo o tipo de conexão do protocolo TCP requere o *three-way handshake*, que consiste num conjunto de mensagens trocadas ente cliente e servidor:

- O *three-way handshake* é iniciado quando o cliente envia uma mensagem SYN ao servidor;
- O servidor recebe a mensagem e responde com outra mensagem SYN/ACK;
- Por último, o cliente confirma que a conexão com uma mensagem ACK final;

O propósito desta troca de mensagens encontra-se na validação da autenticidade de cada conexão para que se possa estabelecer uma chave de encriptação que permitirá assegurar as comunicações subsequentes.
Portanto um *TCP SYN flood attack* ocorre quando o atacante satura o sistema com pedidos SYN com o objetivo de sobrecarregar o seu alvo e impossibilitá-lo de responder a novos pedidos de conexão reais. Este tipo de ataques é por vezes denominado por *half-open* por permitirem conexões inseguras que podem resultar no colapso total do servidor.

Ao considerarmos o impacto e as consequências que um ataque destes pode ter num servidor web, torna-se bastante percetível o nível de dano que o mesmo pode realizar numa rede de comunicação V2V/V2X. Acidentes de viação, colapso do funcionamento de veículos ou mesmo situações de atropelamento são alguns exemplos de situações potencialmente perigosas que podem resultar em perda de propriedade ou até mesmo perda de vida humana.

IV. DETALHES DA IMPLEMENTAÇÃO

Optamos por fazer uso do *Veins Simulator* como principal ferramenta de observação, análise e simulação de um ambiente veicular. Este simulador permite-nos gerir tráfego, acidentes de viação e também trocas/disseminação de mensagens numa dada rede veicular. *Vehicles in network simulation* (VEINS) teve na sua constituição especial atenção na comunicação entre veículos. A principal limitação deste simulador encontra-se na impossibilidade de utilização de certas funcionalidades para propósitos experimentais. A sua implementação não pode ser descrita como *vehicle-to-everything (V2X).*

A. Simulação utilizando subprojecto Veins_Inet

O Veins fornece um conjunto abrangente de modelos específicos de *Inter-Vehicle comunications (IVC)* que servem como estrutura modular para simular aplicações. Cada modelo está contido no OMNET++ podendo ser instanciado numa simulação em execução para dessa forma fornecer a funcionalidade necessária.

O Veins_INET é um subprojecto (incluído no Veins) e foi o exemplo escolhido para podermos demonstrar no POC, permite usar Veins como um modelo de Inet. Neste caso iremos exemplificar com INET4, dessa forma poderemos usar o conjunto completo de recursos de Inet *framework* numa simulação Veins unindo o melhor dos dois mundos, usar o Veins como uma base sólida para uma junção bidirecional de uma rede e uma simulação de tráfego rodoviário, bem como configurar, executar, avaliar e gerir essas configurações tal como agregar o conjunto de recursos fora da comunicação V2X puramente baseada em WLAN oferecidas pelo INET *framework*.

B. Como funciona?

De forma a utilizar o simulador resolvemos instalar máquina virtual com sistema operativo UBUNTU. A versão instalada do OMNET++ foi a mais recente disponível, neste caso a versão 6.0.1.

Para colocar o OMNET a correr temos inicialmente de entrar no diretório onde o mesmo se encontra instalado e enviar o comando:

$ source setenv
$ omnetpp

```
fabiana@Ubuntu:~/Downloads/omnetpp-6.0.1$ source setenv
Environment for 'omnetpp-6.0.1' in directory '/home/fabiana/Downloads/omnetpp-6.
0.1' is ready.
fabiana@Ubuntu:~/Downloads/omnetpp-6.0.1$ omnetpp
Starting the OMNeT++ IDE...
fabiana@Ubuntu:~/Downloads/omnetpp-6.0.1$
```

Figura 4. Comandos que tem de ser enviados para iniciar a simulação

O IDE é aberto para podermos utilizar a nossa simulação.

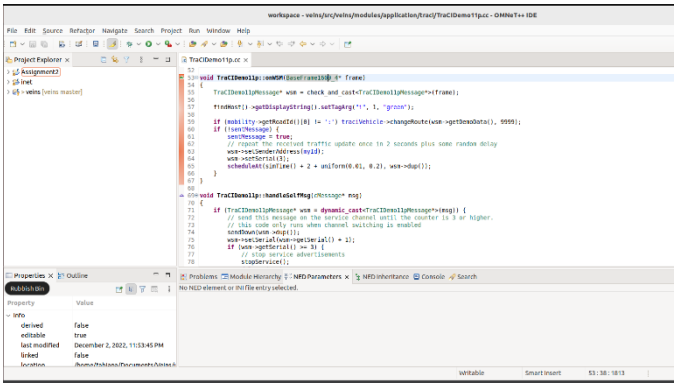

Figura 5.OMNET++ IDE

De forma a utilizar as duas *frameworks* INET e VEINS o projeto faz referência aos mesmos, assim quando se faz build podemos usufruir de ambas as ferramentas pois ambas estão instanciadas.

Para a simulação funcionar e de modo a podermos incluir o SUMO na mesma temos de colocar o servidor do Veins a correr.

```
fabiana@Ubuntu:~/Documents/Veins/veins/bin$ ./veins_launchd -vv -c /home/fabiana
/Documents/sumo/bin/sumo-gui
Logging to /tmp/sumo-launchd.log
Listening on port 9999
```

Figura 6: Comando para correr o servidor VEINS juntamente com o SUMO

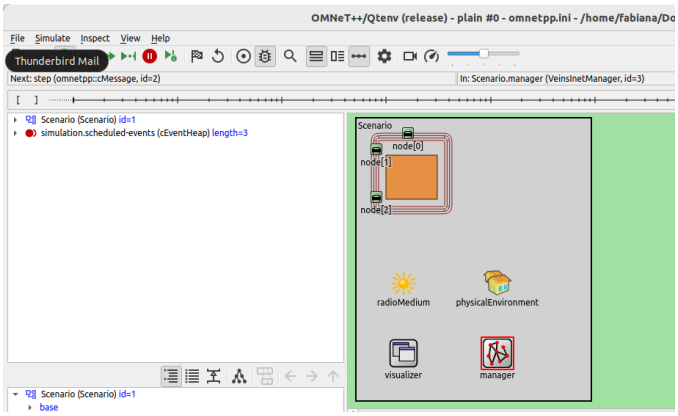

Figura 7: 2 OMNET++ com a simulação a correr

Os nós/carros herdam do INET um *WIRELESS-Host* padrão utilizam o protocolo de rede IEEE802.11 juntamente com uma pilha TCP/IP completa. Cada carro/nó usa um módulo de mobilidade do tipo ***VeinsInetMobility*** para rastrear a sua posição. Para configurar a pilha de rede dos carros que entram na configuração necessitamos de um módulo ***HostAutoConfigurator*** que deve ser incluído em cada carro. O ***VeinsInetManager*** cria e gere os carros na simulação.

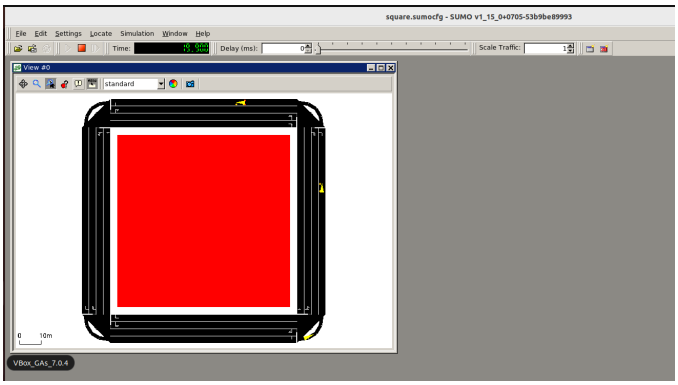

Figura 8: 3Sumo com a simulação a correr

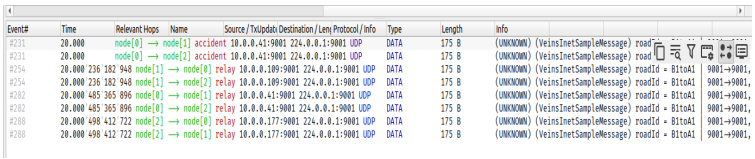

Figura 9: Formato das Mensagens durante a simulação

Durante a simulação o que podemos observar é que assim que o primeiro nó encontra um acidente, esse mesmo evento é partilhado com os restantes nós, onde por consequência graças a essa mesma informação alteram o seu trajeto de modo a não serem incomodados por esse mesmo evento e não haver congestionamento de trânsito.

V. AVALIAÇÃO E VALIDAÇÃO DO SETUP, RESULTADOS E CONCLUSÃO

Com a simulação conseguimos implementar as comunicações entre veículos e um cenário onde um carro tem um acidente, comunica aos outros carros e estes mudam a sua trajetória de modo a evitar o acidente.

No entanto, não conseguimos implementar e concretizar o ataque, pois como foi necessário utilizar várias *frameworks* para as diferentes partes da simulação, não conseguimos perceber que ficheiros alterar e como criar um ataque *flood*, já que o projeto tem muitas bibliotecas, pastas e dependências no código de outros ficheiros. Para além disto, encontramos pouco apoio na internet para este tipo de simulação, já que é um tema recente e o que encontramos nunca explicava realmente como perceber o código com que estávamos a trabalhar.

Embora não tenhamos conseguido demonstrar o ataque, conseguimos prever pela simulação que se houvesse um ataque de inundação da rede com muitas mensagens maliciosas, causando um *denial of service (DoS),* e os carros não recebessem a comunicação de que houvera o acidente, estes seguiriam a trajetória inicial não evitando assim o acidente. Isto aconteceria também num ataque *man-in-the-middle* que impedisse as mensagens de chegar aos carros recetores.

Assim, concluímos que as comunicações V2V/V2X serão responsáveis para um grande progresso e inovação no mundo veicular melhorando a qualidade de vida do ser humano através da prevenção de acidentes, aumento na eficiência da gestão do tráfego e na condução autónoma. Infelizmente com todos os benefícios que esta nova tecnologia nos fornece estão associados também novos riscos que podem pôr em causa a integridade do veículo e dos seus ocupantes. *Flood attacks, man-in-the-middle attacks e replay attacks* são algumas das ameaças que esta nova tecnologia terá que enfrentar e superar de modo a que todos possamos usufruir de maneira segura de todos os pontos positivos que estas comunicações nos oferecem.

REFERENCES

[1] SUMO OMNeT++. (2020, April 10). Omnet Simulator. https://omnetsimulator.com/sumo-omnet/

[2] OMNeT++ Discrete Event Simulator. (n.d.). https://omnetpp.org/

[3] INET Framework - What Is INET Framework? (n.d.). https://inet.omnetpp.org/Introduction.html

[4] Veins. (n.d.). https://veins.car2x.org/