Honeypot Application with NS-3 Emulation

Lucas Coelho de Almeida, Andre Araujo de Medeiros, Claudia Jacy Barenco Abbas
lucas.almeida@redes.unb.br,andre.araujo@redes.unb.br,barenco@unb.br
University of Brasilia (UnB)
Faculty of Technology - Darcy Ribeiro Campus - Electrical Engineering Department
Brasilia, Brazil

ABSTRACT

This investigation aims to provide a NS-3 emulation of a honeypot application. The emulated network will be employed to receive attacks, therefore, it will be vulnerable to scans (like NMAP's application), spoofing and DoS (Denial of Service) attacks. Ideally, when this network receives an attack, an emulator will send a message to the network administrator with the attacker's IP address and in a more sophisticated environment, one's firewall will automatically block the attacker's address. This detection would be possible with constant and real time analysis of network parameters such as throughput and packet delay of the emulated network. With this, it would be possible to create emulated honeypot networks that present real challenges to attackers.

CCS CONCEPTS

• **Networks** → Software Emulated Networks.

KEYWORDS

ns-3, emulation, honeypot, security, network attack

ACM Reference Format:
Lucas Coelho de Almeida, Andre Araujo de Medeiros, Claudia Jacy Barenco Abbas. 2019. Honeypot Application with NS-3 Emulation. In *Florence '19: Workshop on ns-3, June 19–20, 2019, Florence, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

NS-3 is a discrete event network simulator for internet systems, focused primarily on research and educational use. This simulator is under the license of free software, therefore, it is publicly available for research, development and personal use. The NS-3 is also designed for integration with real network environments. That means it is possible to emulate protocols and the operational behaviour of the physical and link layers of a standard TCP/IP network. In other words, the application can integrate the virtual world of the NS-3 simulation with the real world. This is only possible because most operational systems already have the ability to implement virtual network devices, which is largely used by virtualization softwares like Oracle's VirtualBox and Vmware's vSphere.

Florence '19, June 19–20, 2019, Florence, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

NS-3 emulation module provides two types of virtual network devices, the first being the "Emu NetDevice", which allows the NS-3 to send data to a "real" network (see figure 1) and the second device is the "Tap NetDevice", which allows an actual host to participate in a simulation as if it were one of the simulated nodes. A simulation can be done in several ways by combining these devices. The "NetDevice" is the interface that defines the API (Application Programming Interface) that the IP and ARP layers need to access to manage an instance of each network device. Specifically, this class encapsulates the correct format of MAC addresses used by a device (see reference 4).

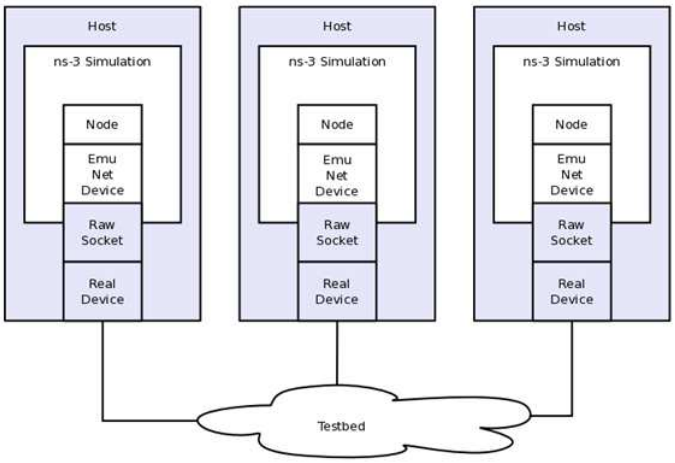

Figure 1: Example of an emulation with NS-3

The NS-3 network emulation module provides a controllable environment and allows the evaluation of behaviours of adapter cards and real protocols. In emulation, real hosts running real-world applications and protocols are able to interact through an environment that simulates specific network conditions. This way, NS-3 can be utilized as a tool for analyzing network attacks of the DoS (Denial of Service) type or other scans that are part of the preliminary phases of an attack like NMAP. A network attack can be defined as any method, process, or environment used to attempt to compromise network security. Users who perform network attacks are commonly called network intruders, hackers, or crackers. These attacks are done to steal data, steal software, use data for financial gain or spying, modify stored data, make a host unavailable, among other types of violations. Specifically DDoS (Distributed Denial of Service) attack is an attempt to make a system's resources unavailable to its users (other programs can also be users of resources). Web servers are the typical targets of this attack,

which seek to make hosted pages and services unavailable. DDoS attacks overwhelm these web servers, making them unavailable. This type of attack may force a system to reboot, consume all network resources or occupy the communication channel between users and the server. Thus, by using NS-3 that emulates data exchange with the real world, it would be possible to identify a DDoS attack looking at only the throughput and packet delay of the network. Also, the emulated network could be used as a decoy to confuse and/or misguide a possible malicious attacker. In addition, considering a complex scenario, the emulated network could have lots of internal TCP/UDP connections and subnets and every part of the simulation would form a maze that would make it tremendously difficult for an attacker to find valuable information.

2 EXPERIMENTAL ENVIRONMENT

The NS-3 module used for this work was the "TapBridge" feature which allowed an integration between a real-world host and the simulation. This feature has this name because it connects the inputs and outputs of a NS-3 network device to the inputs and outputs of a Linux network device (i.e., through software, a network device is created and it acts like one of the physical world but is able to transmit data to a virtualized world. That is exactly what Linux's well-known "TUN / TAP devices" preform. It should be noted that the behaviour of transmitted data from one side to the other is similar to the task of a bridge device. There are three basic operating modes of this module available to users which are: "ConfigureLocal", "UseLocal", and "UseBridge". In "ConfigureLocal" mode, the "TapBridge" is responsible for creating and configuring "TAP" devices. In "UseBridge" or "UseLocal" modes, the user provides a setting and the "TapBridge" module adapts the emulation to this setting. These latter two cases, however, are not yet fully documented and their implementations are limited. They require adapter configurations that do not reflect the normal operation of today's network interfaces. Knowing this, the concept to be presented in this work was based on the "ConfigureLocal" mode, which is the best tackle to the objectives of this analysis.

3 DISCUSSION OF RESULTS

The initial goal of the experiment was to create an application that would continually emulate a network in NS-3 and that would constitute an easy target for attackers. When any attempt of attack/intrusion to this emulated network is made, the simulation should trigger external scripts and warn network administrators (users or other software) of the attempt. However, it has been found that NS-3-supported emulation is still very limited due to NS-3's own features of processing and responding to real applications. With the simulator, it is possible to imitate various behaviours but NS-3 still does not have complete support to include external programs in projects (such as an HTTP server that actually receives requests and responds with HTML code). However, since it is possible to embed simple applications into NS-3's simulations, the requirements for this investigation of an attack on an emulated honeypot network, were completely fulfilled with the built-in NS-3 modules.

Without the development of a true application or external program to be embedded in the NS-3 emulation, it would not be possible to reach the original goal (a program that detects and warn of real time invasions). Previous studies on the subject provide another view of the question, perhaps even more interesting than the previous one: NS-3 is probably the safest way to "trick" potential attackers. Emulated networks can contain extremely complex topologies and with knowledge of programs that exploit lower layer information (such as PING, ARP, NMAP, and TraceRoute) as well as knowledge of routes and routing protocols, an attacker could be made to spend hours investigating a network without even suspecting that it is an emulation. In addition, the most curious fact is that, because NS-3 is designed specifically for network performance analysis, and does not contain traces of actual applications responding accordingly to requests, an attacker could never make a successful attack on a node of the emulated network. After all, there are no services to be knocked over, only the features that the NS-3 is using for the target socket of the attack at that moment. What is even more curious is that the main tool of the attacker will not be able to detect an emulation. This explain why it is possible to "ping" these nodes, scan them and find routes among other things but it is not possible to interact at the application level. Therefore, it can be concluded that NS-3 emulation feature is possibly one of the best forms of creating complex honeypot applications and not only run tests on it, but use it to actually prevent attacks in the real world.

4 CONCLUSION

It was possible to build a real and useful application using just the built-in features of NS-3 and with more effort it could be possible to develop and implement lots of security features for real networks. The main purpose of this investigation was to show that the NS-3 tool is suitable not only for analysis tasks but for building and providing better tools for real world network administration. In the appendix, the reader can visualize how useful the NS-3 emulation module is when one tries to perform some of the most common operations of network attacks and scans.

5 REFERENCES

(1) "NS3" - Barenco Abbas, Claudia"
(2) "Tap NetDevice" - https://www.nsnam.org/docs/models/html/tap.html
(3) "Emu NetDevice" - https://www.nsnam.org/docs/release/3.11/models/html/emu.html
(4) "Emulation Overview" - https://www.nsnam.org/docs/release/3.11/models/html/emulation-overview.html
(5) https://www.nsnam.org/docs/release/3.9/doxygen/group___tap_bridge_model.html
(6) https://www.nsnam.org/doxygen/tap-wifi-dumbbell_8cc.html
(7) https://www.nsnam.org/doxygen/tap-wifi-dumbbell_8cc_source.html
(8) https://www.nsnam.org/doxygen/tap-csma-virtual-machine_8cc_source.html
(9) https://www.nsnam.org/doxygen/tap-bridge-helper_8cc_source.html
(10) https://www.nsnam.org/wiki/HOWTO_make_ns-3_interact_with_the_real_world

APPENDIX - EXPERIMENTAL PROCEDURES

A simple network with 3 routers was designed where the first one communicates through a WiFi network with the second one and the second is connected through a point to point connection with the third. The emulated host is connected directly to the first router which uses the TapBridge module, connecting the real computer host to the simulation. This router can communicate with the other nodes of the simulation through the NS-3 internal network. The third router is connected to a LAN Ethernet network, where users can communicate with each other and also with the emulated host (see figure 2).

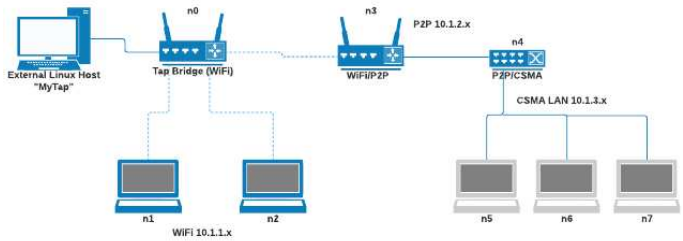

Figure 2: Topology used to emulate communication between a Linux host and NS-3 simulated network (inspired in reference 7 of the main document).

The following images show results obtained when testing a DoS attack with a Python script to the emulated network described earlier and the test of other commonly used attack tools such as NMAP, Traceroute and Ping.

Figure 3 shows the NS-3 compiling and running the emulation script. The host used was a virtual machine which had Linux Ubuntu distribution.

```
root@latitude-VirtualBox:/opt/ns-allinone-3.27/ns-3.27# ./waf --run
trabalho-adr-lucas-andre
Waf: Entering directory `/opt/ns-allinone-3.27/ns-3.27/build'
[ 955/2702] Compiling scratch/trabalho-adr-lucas-andre.cc
[2670/2702] Linking build/scratch/trabalho-adr-lucas-andre
Waf: Leaving directory `/opt/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (20.604s)
Versão do TCP utilizada : ns3::TcpSocketFactory
```

Figure 3: Emulation script running, after being compiled.

Figure 4 shows the return of the command "ifconfig" which lists the network devices installed in the host. Note that there is a device called "thetap".

Figure 5 shows the first most likely action an attacker would take after gaining access to the Linux host: run the "arp -a" command to discover the other nodes it can access.

Figure 6 shows the return of the command "ping 10.1.1.2", which attempts to reach the node with this IP address. This node is the second computer connected to the TapBridge router in figure 2. This would be the second most likely behaviour after the use of "ARP".

```
          Interrupt:16 Base address:0xd240

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:80 errors:0 dropped:0 overruns:0 frame:0
          TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6951 (6.9 KB)  TX bytes:6951 (6.9 KB)

thetap    Link encap:Ethernet  HWaddr 00:00:00:00:00:01
          inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::200:ff:fe00:1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:516 (516.0 B)

latitude@latitude-VirtualBox:~$
```

Figure 4: TapDevice created by the program, after being compiled.

```
root@latitude-VirtualBox:/home/latitude# arp -a
? (10.1.1.2) at 00:00:00:00:00:02 [ether] on thetap
? (10.61.22.1) at 00:1c:7f:62:b2:b5 [ether] on enp0s3
? (10.1.1.4) at 00:00:00:00:00:04 [ether] on thetap
root@latitude-VirtualBox:/home/latitude#
```

Figure 5: The ARP program can detect the presence of other nodes in the network '10.1.1.0'.

```
root@latitude-VirtualBox:/opt/ns-allinone-3.27/ns-3.27/scratch# ping
 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=11.9 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=2.73 ms
64 bytes from 10.1.1.2: icmp_seq=3 ttl=64 time=2.53 ms
64 bytes from 10.1.1.2: icmp_seq=4 ttl=64 time=1.57 ms
64 bytes from 10.1.1.2: icmp_seq=5 ttl=64 time=3.79 ms
64 bytes from 10.1.1.2: icmp_seq=6 ttl=64 time=2.52 ms
```

Figure 6: The actual host is already able to use the program "Ping" with nodes of network '10.1.1.0'.

Figure 7 shows the most likely behaviour an attacker would take after using a sniffer application or realizing that there is a conversation between a node in the "10.1.3.0" network and the other node in "10.1.1.0" network. By running the "traceroute" command it is possible to discover the routes needed to be added to the routing table.

```
root@latitude-VirtualBox:/home/latitude# traceroute 10.1.3.1
traceroute to 10.1.3.1 (10.1.3.1), 30 hops max, 60 byte packets
 1  10.1.1.2 (10.1.1.2)  15.373 ms  15.788 ms  16.743 ms
 2  10.1.1.4 (10.1.1.4)  17.404 ms  17.669 ms  17.934 ms
 3  * * *
 4  * * *
 5  * * *
 6  * 10.1.2.2 (10.1.2.2)  23.496 ms  27.269 ms
root@latitude-VirtualBox:/home/latitude#
```

Figure 7: The TraceRoute program can trace the route to reach the network '10.1.3.0'.

Figure 8 shows in red the addition of two new routes to the routing table of the Linux real host. The other two underlined commands show that, after the successful addition of the new routes, the networks "10.1.2.0" and "10.1.3.0" of the figure 2 are accessible using "ping" commands. In this case, the developer had to put a TCP client in one of the hidden nodes in "10.1.3.0" and a TCP server in the "10.1.1.0" network. Therefore, in a real life scenario, an attacker using sniffer applications (such as Wireshark) would see the conversation between the nodes and try to add routes to reach the other network. The conversation, then, would be a "trap" to gain attacker's attention. It is worth to note that every aspect of the design of the simulation can be used to make a honeypot emulated network.

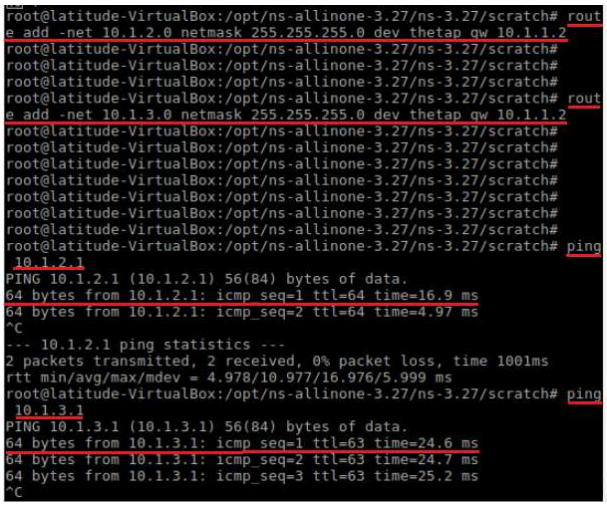

Figure 8: After setting up two new routes, the actual host is also able to use the "Ping" program to network nodes '10.1.2.0' and '10.1.3.0'.

Figure 9 shows the scan of node "10.1.1.3" using "NMAP" to look for open TCP ports. This is the node which has the TCP Server and the application finds the open port "8080".

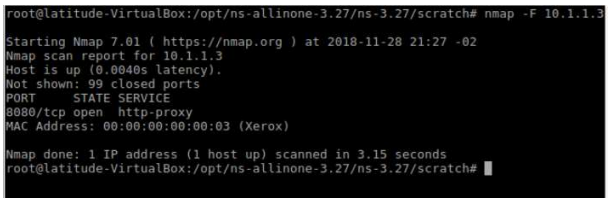

Figure 9: The NMAP program realizes that there is a TCP application on node 10.1.1.3 using port 8080.

Figure 10 shows the triggering of a Python script which performs a DoS attack.

Figure 11 shows the result of the DoS attack from the attacker's perspective. It is a success and there are no hints that it was just an emulation. For each connection it prints a text and this particular script needed 30.000 connections to consume the simulation

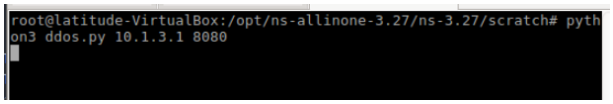

Figure 10: A DDoS attack script written in Python is triggered by targeting the IP address 10.1.1.3, which has port 8080 vulnerable.

resources. It is worth to note that, as NS-3 is developed in C++ language and doesn't have the overheads of the application layer, it is far more difficult to attack it successfully: it is really fast compared to the other implementations of servers. As an addition, after the end of this experiment, the attack was directed to a Python web server. It was around ten times more difficult to stop the simulation presented with a DoS resource consuming attack rather than with a Python real web-server.

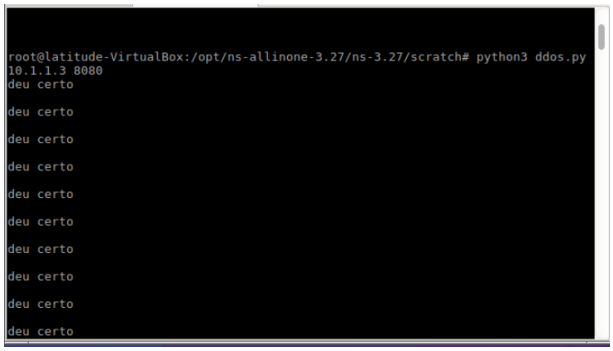

Figure 11: The attack is a success, and there is no evidence of a false application.

Figure 12 shows the result of the DoS attack from the simulator's perspective (the simulation lasted 10 seconds). The simulation stopped after the huge number of 30.000 connections were left open. The IP address of the attacker is shown as "10.1.1.1", the same IP address of the first router in figure 2 which shows the topology used in this experiment. It proves that the Linux host and the simulated router where seen as the same node inside the emulation.

As FlowMonitor tool can't show results in real time (it only delivers results after the completion of the simulation) it is important to build an external program that detect attacks of an emulation, collects the attackers real IP address and advises other programs or administrators. In our experiments, simulations never stopped until they reached 10 seconds (the time programmed inside script), but attacks consumed all resources of networking process inside the Operational System and the communications and simply stopped working before the configured end time. After this, it was necessary to restart the virtual machine. But it should be noted that the development of an external program working with NS-3 could solve almost every one of these problems and automatically reinitialize the emulation after these attacks.

Figure 12: The NS-3's FlowMonitor tracks all connections and data streams, and after the end of simulation, it list all the requests resulted by the Python DoS attack script.