MC2: Peer to Peer Based Network Masquerading for

Mission Critical Clouds

Francois D’Ugard, fduga002@fiu.edu, Florida International University

Ming Zhao, ming@cis.fiu.edu, Florida International University

ABSTRACT:

Virtualization is an increasingly popular approach to manage rising information technology costs and complexity in every sector of the economy. Cloud computing allows organizations of any size to provision infrastructure resources as needed and flexibly scale technology resources to meet changing demands. Cloud providers pool hardware resources and allocate them based on the requests of their users. In order to efficiently allocate these resources providers must aggregate users of different requirements and workloads onto the same physical infrastructure. However, this approach increases the likelihood that a malicious user can collocate a VM alongside a target VM in order to extract information or disrupt its functioning in some way.

This project will deliver mission assurance to mission-critical applications in cloud computing systems. Our approach relies on developing a complete network graph on a virtual private network of peer to peer connections. With the purpose of masquerading the messages created by co-operative virtual machines in a typical cloud computing system. Our network graph consists of a peer to peer overlay network that interconnect OpenStack virtual machines and is based on the IP-over-P2P (IPOP) framework. The project will focus on developing an extension to IPOP that will allow for the communications among the VMs to be routed by an overlay network in an OpenStack-based cloud system.

Our proposed solution will deliver mission assurance to mission-critical applications in cloud computing systems. We will do so by leveraging the unique capabilities of virtualization technology and develop a dynamic and distributed approach to route messages among co-operative virtual machines in typical cloud computing systems. We achieve this by using two distinct approaches: 1.) a Tor-like design to route packets through random trusted siblings to obscure the traffic generated by the virtual machine network. 2.) Secondly applying a message forwarding technique to reduce traffic differences and require attackers to change their attack vector from a targeted attack to a random attack as discussed in [1]. This technique increases the availability of important components in the system.

IMPLEMENTATION:

We rely on two distinct approaches to achieve our project goals. The first is a packet encapsulation technique. The second is a network flooding technique that reduces traffic differences among virtual machines. We use these techiniques in tandem to achieve network homogeneity.

In the packet encapsulation technique each outgoing packet is encapsulated with the forwarding address of the next component in the network path. The second technique, described in [1] “Improving complex distributed software system availability through information hiding”, floods the network with forwarded packets to reduce traffic differences. Each controller would forward a packet from 1 to *f* times. The controller chooses a random destination node every time it forwards a packet. The experimentation described in [1] indicates that the optimal number of forwards for a ten component system is between ~4.5 and 5 forwards, with decreasing returns beyond that range.

Packets that originate at the machine where the controller resides (i.e. the local peer) are never inside the scope of the local controller. Instead these packets are handled within the ipop tap device. IPOP TAP is the tap packet handler for ipop-tincan. It performs packet-level operations, translating packets, and the setup, reading and writing to and from the TAP device.

The root of the problem seems to reside in how the controller initializes sockets. The controller relies on AF\_INET or AF\_INET6 as the packet family specified in Python’s socket class. These packet families only view packets received by the local machine. If however instead of specifying AF\_INET/6 we specify the more general AF\_PACKET packet family, we should be able to view every packet that passes through our machine including packets that originate locally. Additionally we can bind a socket to a particular network device using the bind() function. With it we can bind the controller’s sockets to the IPOP device created by ipop\_tincan. I am currently testing this implementation. If this works the following solution may not be required.

We can achieve a similar result by modifying the IPOP TAP component. The function that reads packet data from the tap device that was locally written, and sends it off through a socket to the relevant peer(s) is ipop\_send\_thread() and can be found in the packetio.c file in the ipop\_tap source code. Modifying this source code should allow us to manipulate outgoing packets by leveraging the techniques described above.

EXAMPLE:

Given a network M of N = 3 components i.e. M = {A, B, C} and Algorithm 1.

Algorithm 1 Message Forwarding(N, f, dest, msg)

1: for i ← 1 to f do

2: randomly generate a number j within [1, N] except the sender ID

3: if component j = dest then

4: dispatch msg to destination component dest

5: return

6: else

7: forward msg to component j

8: end if

9: end for

10: dispatch msg to destination component dest

11: return

Each component applies Algorithm 1 to every incoming and outgoing packet. When A executes Algorithm 1 it forwards *f* many packets, flooding the network. The destination of each *fi* packet is randomly chosen. If during any one of the *f* iterations component B is chosen it also applies Algorithm 1 to every incoming or outgoing message, the same is true for component C. With some probability it is guaranteed that no packet will travel indefinitely throughout the network.

REFERENCES

[1] Wang, L., Leiferman, Y., Ren, S., Kwiat, K., & Li, X. (2010, March). Improving complex distributed software system availability through information hiding. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 452-456). ACM.