FIREDRL:

Firewall Integrity Review
Exploit Description Report Language

COMS 4115
Project White Paper

Marvin J. Rich
e-mail: mjrich@us.ibm.com

Introduction

The internet is a ubiquitous entity that touches all aspects of modern society. The services provided by the underlying protocols have enhanced business and personal communication to a worldwide scope. As the internet has integrated into the normal processes conducted by society, security has emerged as a necessary component for providing services that can be accessed worldwide. For example, a Local Area Network (LAN) configuration representing a business typically has data online that must be protected from unauthorized access, as well as data that must be accessible by everyone, in order to conduct a competitive business. These opposing requirements must co-exist on a common set of applications and protocols, which makes security of the resulting LAN configuration a complex undertaking for network administrators.

Firewalls are a first line of defense for LAN security. They are not the only component of network security, but an important one nonetheless. A firewall imposes and enforces rules on the type of traffic and services allowed to flow in or out of a network. Typically a firewall will be implemented at a router, which is an entrance/exit point for a LAN. In order to implement the various levels of security and services for a typical LAN, several routers are usually present, which partition the total LAN into segments with varying degrees of access. Firewalls will generally be implemented on each of these routers to enforce a degree of access rights, but may be implemented on host machines also. For large LANs, it is difficult to determine the security imposed by various levels of firewalls, such that the intended services controlled by the firwalls are indeed provided at each host on the LAN.

The Firewall Integrity Review Exploit Description Report Language, FIREDRL (pronounced “fire drill”), is a rapid LAN configuration and exploitation test language. This language is specific to the domain of firewalls. It allows a network administrator, to design the firewall aspects of a network by defining the network, testing fundamental exploits, and reporting the affects of the attack on hosts of the network. This process can be repeated until the administrator is satisfied with the operation of the firewalls in the LAN. The FIREDRL language provides the following capabilities:

- Easy LAN Prototyping
- Easy Firewall rule specification
- Rapid Firewall Exploit Testing
- Flexible Exploit Result Reporting

Overview

For explaining the capabilities of the FIREDRL language, a typical LAN configuration is depicted in Figure 1. The LAN contains 3 segments labeled X,Y and Z respectively. LAN segment X has two routers (R1,R2), and 3 hosts (A,B,C). LAN Segment Y has two routers (R2,R3) and also 3 hosts (D,E,F). LAN segment Z has 1 router (R3) and 2 hosts (G,H). Each LAN segment typically serves some functional requirement. For this illustration, the segments denote different security policies as to what can be accessed. The firewalls are implemented on all the routers for this illustration (though a firewall can also be implemented on a host machine).

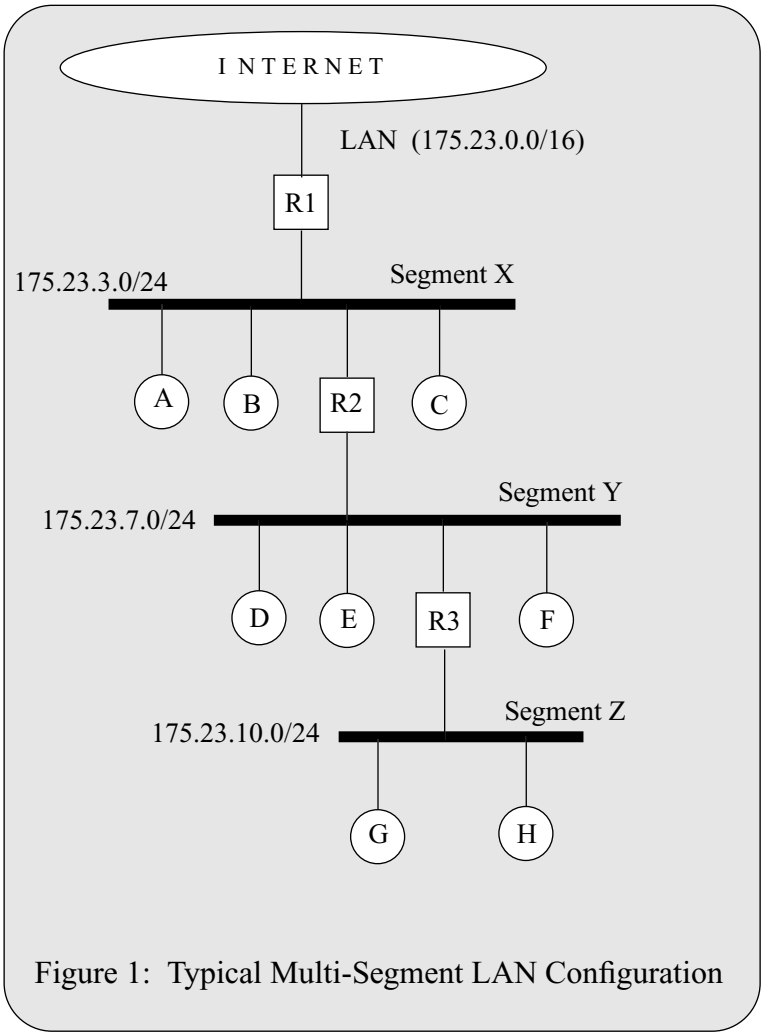

Figure 1: Typical Multi-Segment LAN Configuration

Rapid LAN and Firewall Configuration

The FIREDRL language can easily lay out the topology of a LAN, which is the environment that an exploit attack is analyzed against. Using the example in figure 1, the FIREDRL statements required to describe the LAN are as follows:

```
1   //Define total lan segment topology
2   MyLan       = Segments{Segment_X,Segment_Y,Segment_Z};
3   Segment_X = hostname{A,B,R1,R2,C};
4   Segment_Y = hostname{D,E,R2,R3,F};
5   Segment_Z = hostname{G,R3,H};
6
7   //Define host_ids in each segment
8   Segment_X = hostid{10,20,30,40,50};
9   Segment_Y = hostid{2,3,41,25,61,};
10  Segment_Z = hostid{34,35};
11
12  //Define IP Addresses and Subnets of LAN
13  MyLan       = ip(175.23.0.0/16);
14  Segment_X = ip(175.23.3.0/24);
15  Segment_Y = ip(175.23.7.0/24);
16  Segment_Z = ip(175.23.10.0/24);

17  //Filter rules
18  Segment_X.R1 = filter(IN,PORT=80, YES);
19  Segment_Y.R2 = filter(IN,PORT=25,NO);
```

Lines (2-5) establish the overall network topology by assigning names to key network hosts and routers a top-down fashion. FIREDRL can deduce the firewall hosts by nature that the host name will exist in multiple network segments. Lines (7-10) assigns unique (semantic check) host IDs to the established hosts. Finally lines 13-16 assign IP addresses to the topology. Subnet mask information is appended to the IP address. For example a "/24" string appended to an IP address indicates that the first 24 bits are the network address, which implies a 255.255.255.0 subnet mask. Lines 18-19 configure firewall rules on routers R1 in Segment_X and R2 in Segment_Y respectively. One can filter incoming or outgoing traffic based on the rules. Of course the language is free format with line comments allowed (lines 1,7,12 and 17).

Rapid Exploit Testing and Reporting

An exploit attack is a specific method used to attempt to penetrate the LAN from the internet. FIREDRL provides control instructions which allow logical decisions that can be made to guide an attack sequence. Common exploits are packaged with the support such that only control decisions need be mapped out in the test. An example in FIREDRL is as follows:

```
1   //Issue ICMP TTL Probe Attack
2   success = 0;
3   for (TTL = 2 to 20 by 1) {
4      ICMP(echo,TTL);
5      if  (resp.success)
6         print("Successful Probe for TTL=", TTL);
7   }
```

This attack uses ICMP echo requests, with increasing TTL value, to discover the routing depth in the LAN. This information provides topology clues to an attacker. Familiar programming control constructs such as "for" and "if" statements in lines 3 & 5 allow decisions to be made while progressing through an attack. The control flow constructs are wrapped around an attack directive (line 4) to realize a wide ranging probe of the network. Print statements, as depicted in line 6 provide a method for displaying the results of the attack on the network. The user is free to print the status of interest, based on the placement of print statements.

Implementation Model

The underlying execution language is Java. The FIRDRL language is compiled into Java executables that, given a network topology and an exploit definition, generate a report on how the exploit was handled by the firewalls depicted in the network. The actual internet exchanges are abstracted to object message/response transactions between the attacker and the network. The attacker serves as the client, probing messages to gain knowledge about the network. The network description serves as the server, which provides responses based on the filtering rules specified in its firewall setup rules (or lack thereof).

Availability

The Beta version 1.0 is scheduled for release in December of 2005. A language manual will preceed the delivery in December, which will describe the version 1.0 capabilities. The targeted capabilities for the version 1.0 delivery are as follows:

- Full LAN specification support
- Firewall Filter Rules Support
- Exploit control language support
- Three exploits (Ping Discovery, ICMP TTL Probe, and Port Scan)
- Exploit Results Reporting

Conclusion: The FIREDRL Advantage

The FIREDRL language allows for rapid LAN prototyping and firewall testing. Rapid LAN prototyping is realized by abstracting the LAN down to its key hardware and software components, as viewed from a firewall. The key hardware components are hosts and routers. The key software components are firewall rule sets, IP addresses, and TCP ports. Rapid firewall testing involves a simple control language wrapped around a set of exploitation attack abstractions. This allows easy testing of an exploit as an attacker would invoke it, by using the control constructs to easily direct an exploit to a range of targets.

The FIREDRL advantage is that the abstract prototyping constructs incorporated in the language affords the user a mechanism to gain in depth knowledge of the firewall capabilities of his/her network without a large expenditure in actual network design and testcase effort. The abstraction level also provides a steep performance advantage. The firewall capabilities are summarized for the entire LAN, such that knowledge of a LAN can be gained in a shorter period of time. This ultimately results in a faster implementation of a total LAN security solution through rapid stabilization of the firewall security component.