"Roto-Rooting" your Router:

Solution against New Potential DoS Attacks on Modern Routers

Danai Chasaki
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{dchasaki}@ecs.umass.edu

ABSTRACT

Our work presents the first practical example of an entirely new class of network attacks – attacks that target the network infrastructure. Modern routers use general purpose programmable processors, and the software used for packet processing on these systems is potentially vulnerable to remote exploits. We describe a specific attack that can launch a devastating denial-of-service attack by sending just a single packet. We also show that there are effective defense techniques, based on processor monitoring, that can help in detecting and avoiding such attacks.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking—*Routers*; C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Design, Performance, Security

1. INTRODUCTION

Modern routers are becoming vulnerable to new attacks because of the technology shift towards programmable infrastructure [3]. Most service provider core routers contain programmable multi-core architectures, e.g. Cisco Carrier Routing Systems (CRS-1, CRS-3) and Juniper's T series. These routers are equipped with functionality beyond simple packet forwarding, ranging from load balancing to application security, access control, e.g. F5's BIG-IP products, where these services are available on a single device. While a lot of attention has been given to end system security, very little work has addressed security concerns in the network infrastructure itself. In our work, we consider the data plane of the network where modern routers are now exposed to vulnerabilities like any other software-based system.

We have shown that a single cleverly crafted malicious packet can change the protocol that runs on the processor of the router, and launches a denial-of-service attack [2]. All bandwidth of the outgoing link on the router is absorbed, and the attack propagates to all vulnerable downstream routers. An overview of such an attack example is illustrated in Figure 1. If we think of the impact of "clogging" a series of core routers with "attack" packets causing denial of service, we realize the importance of protecting them. One could think of tackling the problem using a virus scanner or an intrusion prevention mechanism. However, the multi-processor systems-on-chip that lie inside modern routers have relatively small amount of available resources. They cannot support an operating system and cannot afford the computationally heavy security solutions that are traditionally used to protect end systems. More importantly, any software based solution is not fast enough for our problem. A single packet's processing cycle lasts for only a few microseconds, which means that if the attack is launched by only one packet, its effects will be visible instantaneously. The whole system will "clog" before the intrusion detection scheme is able to detect the attack.

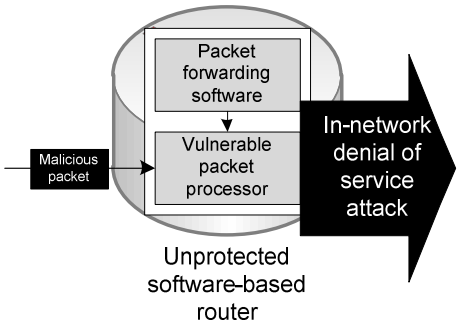

Figure 1: Example of in-network attack.

Our router "roto-rooting" technique is a hardware based solution adapted from embedded systems security literature. It monitors the router's processor in real time, and once deviation from normal behavior is detected, the processor is reset and the router is brought back to correct operation. This happens within 6 instruction cycles (∼100 ns), which is a quick enough response to detect and recover from the DoS attack. Our monitoring system is a light-weight method that does not slow down the system and consumes a small amount of resources.

2. ATTACK DEMONSTRATION

Routers perform a variety of protocol processing operations (IP forwarding, intrusion detection, tunneling etc.). While we don't have access to confidential network development source code, we can safely assume that some protocol operations will require adding a header to a packet. Thus, the kind of attack we describe can be realized in any network protocol that contains hidden vulnerabilities.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.
ACM 978-1-4503-0797-0/11/08.

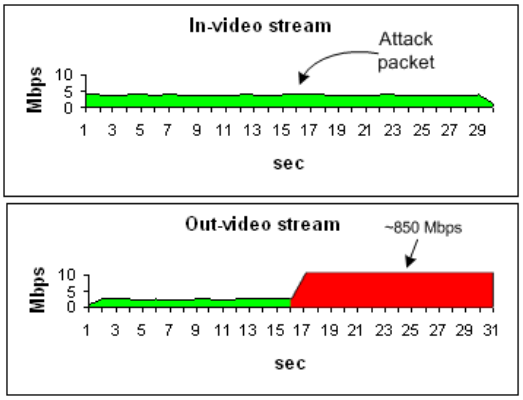

Figure 2: Traffic Rates at Input Port and Output Port of Vulnerable Router. Benign video traffic is shown in green, attack traffic is shown in red.

Here, we demonstrate that vulnerabilities in software-based routers are not only hypothetical, but can occur in common protocol processing code. For the discussion of our attack, we picked the congestion management (CM) protocol, which uses a custom protocol header that is inserted between the IP header and the UDP header. A security aware programmer would perform a check on the packet's total size, before shifting the UDP datagram and inserting the new header into the original packet. Since the total length field of the UDP header is a 16-bit field, the programmer could choose to assign the packet length to a 'unsigned short' integer type, so that the embedded processor's limited resources are not wasted. That is the part of the protocol that contains an integer overflow vulnerability.

The vulnerability does not exhibit problematic behavior for most "normal" packets that are short enough to accommodate the new CM header within the maximum IP packet length. However, if an attacker sends a packet with malformed UDP length field (e.g. 65532), the size check passes even though it should not. The, a large number of bytes is copied into the new packet buffer, which is designed to hold data up to the maximum datagram size. This last step has resulted in the notorious buffer overflow attack, which will overwrite the processor's stack, and finally overwrite the return address of the program. Thereby, the attacker can make the program jump to malicious code that is carried inside the packet payload. In our attack, we insert a few instructions of assembly code into the payload, which repeatedly broadcast the same attack packet in an infinite loop.

We have implemented our attack on the Click modular router and on a custom packet processor [1] based on the NetFPGA platform. We send video traffic into a router that implements the CM header insertion routine, which exhibits the integer overflow vulnerability. Figure 2 shows our results. Benign traffic is sent and the router forwards it as expected until a single attack packet is injected into the incoming traffic. Since the attack packet triggers an infinite loop of retransmitting itself, all output traffic consists of attack traffic. Only one packet of this type has caused a denial-of-service attack that jams the router's outgoing link at full data rate.

3. DEFENSE MECHANISM

To defend against this type of attack on the packet processing systems of routers, we proposed a secure packet processor design [1]. We briefly describe the operation here and demonstrate that it can defend against the attack we describe.

It uses a fine-grained hardware monitor to track the instruction-level operations of the packet processor. These operations are compared to a reference model of operation that has been obtained through offline analysis of the processor's binary file. Under normal conditions, the operations reported by the processor match the offline model. If an attack occurs, the behavior of the processor changes in an unexpected way (e.g., executes malicious code instead of executing the functions that it was programmed for) and the executed operations no longer match the reference model. The secure packet processor can detect this condition, drop the offending packet, and initiate a recovery process that resets the processor core and allows the normal operation to resume.

We have implemented this type of monitor on the same custom processor used for the experiments shown in Figure 2. The security monitor runs in parallel to the packet processor, and is designed to use four pipeline stages. Thanks to the tight hardware architecture, the prototype successfully detects the example attack, halts the processor, drops the packet, and restores the system within 6 instruction cycles. This small recovery time allows our router to operate at full data rate even when under attack. The overhead for adding a monitoring system to the packet processor is very small (0.8% increase on slice LUTs and 5.6% on memory elements).

Since the security monitoring performance results on a single core processor are encouraging, we are planning to extend this work to highly parallel routing systems. The ultimate goal is to have a 16-core version of the secure packet processor running on a development board, and evaluate its throughput performance and scalability. The challenges of a design where security monitors run in parallel to each core are a) how to share programs between the cores, and b) how to share the offline models of the programs between the monitors.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CCF-0952524.

4. REFERENCES

[1] Chasaki, D., and Wolf, T. Design of a secure packet processor. In *Proc. of ACM/IEEE Symposium on Architectures for Networking and Communication Systems (ANCS)* (La Jolla, CA, Oct. 2010).

[2] Chasaki, Danai, W. Q., and Wolf, T. Attacks on network infrastructure. In *Proc. of IEEE International Conference on Computer Communications and Networks (ICCCN)* (Maui, HI, Aug. 2011).

[3] Cui, A., Song, Y., Prabhu, P. V., and Stolfo, S. J. Brave new world: Pervasive insecurity of embedded network devices. In *Proc. of 12th International Symposium on Recent Advances in Intrusion Detection (RAID)* (Saint-Malo, France, Sept. 2009).