Inferring Packet Processing Behavior using Input/Output Monitors

Danai Chasaki, Qiang Wu and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{dchasaki, qwu, wolf}@ecs.umass.edu

ABSTRACT

Programmable packet processors have replaced traditional fixed-function custom logic in the data path of routers. This programmability introduces new vulnerabilities in these systems that can lead to new types of network attacks. We propose a modular packet processor monitoring technique that can help in detecting and avoiding such attacks. Using information about the processing time distribution of individual modules, input/output traffic characteristics can be inferred and abnormal behavior can be detected.

General Terms

Design, performance, security

Categories and Subject Descriptors

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

1. INTRODUCTION

Routers lie at the core of the network. Modern routers need to support many new functions: advanced packet processing, in-network services [4], etc. To achieve this level of flexibility router implementations are based on network processors. In recent years, the advances in the performance of general-purpose multi-core processors has enabled the development of routers with highly parallel, embedded multi-processor systems-on-chip (MPSoCs) as integral components. When the data path of routers was based on application-specific integrated circuits (ASICs), it did not present a potential target for attacks. With the use of programmable components in the data path, this premise has changed. In the data plane of the network, where the actual network traffic is transmitted between end-systems and routers, attackers may aim to eavesdrop on or intercept communications. In our recent work, we have shown that vulnerabilities in the protocol processing functions of routers can be exploited [2].

Defense mechanisms against such attacks have been proposed in our previous work [1]. A limitation of this prior work is the assumption that packet processing is a monolithic processing operation. This assumption, however, does not apply in practice. Due to the complexity of different packet processing tasks implemented on a router, modularity in processing is a necessity. Several approaches to modularity in packet processing have been proposed [3]. Therefore, we need a solution that can support security in such a modular router.

The monitoring approach we have developed previously lends itself well for modular implementations as we show below. Nevertheless, the use of independent monitors for different processing modules poses a new type of security challenge: How can be ensured that the overall operation of a router *across modules* is correct? In particular, an attacker may change the operation of a router such that individual module monitors cannot detect an attack. Therefore, it is necessary a comprehensive defense mechanism that can protect modular router implementations from data path attacks.

2. VERIFICATION IN THE DATA PATH

In this work, we present a modular technique for validation of a router's correct operation by monitoring the "processing" of each module, examining input/output flow characteristics and correlating them with the processing time spent on individual modules.

2.1 Protection Mechanisms

In previous work we have demonstrated the feasibility and effects of such an attack in a real network setting. We implemented it on the Click modular router [3] and on a custom packet processor that has been built in our lab [5]. With just one malicious packet we exploit an integer vulnerability and manage to shut down all packet forwarding on the Click router. On the custom packet processor, which is based on the NetFPGA platform, the attack has a more devastating effect. A single attack packet absorbs the whole system bandwidth, and can propagate the effect to downstream routers as well. Clearly, there is a need to protect modern software-based router designs. In this paper we discuss the limitations of the solution we proposed in [1] and extend it to a more thorough approach for data path protection.

One way to counter this type of attacks is to use processing monitors which track the operations on the network processor, as we proposed in [1]. The monitor can determine if attacks occur because the processor's operations deviate from the operations that are valid – as determined by offline analysis of the processing binary. In that work, we effectively monitor the program execution as a whole.

However, due to vulnerabilities in the data path, we can expect attacks at the protocol level as well. We can have a situation where valid processing instructions are executed on the network processor, but still the overall router behavior is abnormal. For example, in the case of intentionally mislead-

ing routing information advertisement, a large amount of multi-cast packets could be directed to the wrong router interfaces. While observing the system's operation as a whole, we would detect incorrect packet flow behavior even though monitoring the "processing" of the system would determine that the program execution in an instruction-per-instruction basis is correct.

Overall, the processing monitor approach works well, but in terms of packet flow validation it is difficult to say anything detailed about the behavior of a complex piece of code. It can only be done at a level where rules are very lose. This problem is closely related to the lack of modularity in the processing monitor. Therefore, we propose a *modular* approach for verifying data-path processing.

2.2 Modular Verification

We develop a scalable solution which uses both a processing monitor and a packet flow monitor per module. We refer to Click as our example router architecture, because the system is already divided in individual packet processing modules (elements). Different elements implement simple router functions, which can be combined to one graph and build a complete and extendable router configuration.

We are able to verify each module separately and thus the system operation as a whole, by setting up and using a processing monitor and an input/output monitor (packet flow monitor) for every element. The power of this approach lies in the users' ability to combine modules into arbitrary graphs where packets proceed along one path through the graph. First, we can verify the execution path of every element by using a processing monitor [1], that tracks all the individual instructions executed on the processor while the particular element is active. Secondly, we can profile the processing time and delay that each packet encounters in every element. These characteristics depend on the functionality of each element and can help us determine if the system encounters any unusual delays while processing the packets. The way the modules are composed allows aggregation of packet flow specification and thus verification of router operation from port to port.

2.3 Port-to-Port Packet Flow

Assuming that we have n processing elements $m_1 \dots m_n$, the overall functionality of the router is represented by a graph connecting these elements. Graph $G = (M, E)$ consists of set of all elements, M, and set of directed edges, E, indicating that transition of traffic between elements is allowed. For simplicity of our discussion, we assume that all packets enter processing at node m_1 and leave processing at node m_n. Let $\alpha(m_i)$ denote the arrival rate of traffic a module m_i. The departure rate at module m_i is denoted by $\delta(m_i)$.

Problem statement: determine what the appropriate $\delta(m_n)$ of the router is. It should not be too low, which would indicate too many drops; and not too high, which would indicate denial of service attack.

Approach: Starting with $\alpha(m_1)$ and based on the processing time distribution of each and every node in the graph, we compute the range of values of $\delta(m_n)$.

To determine the characteristics of traffic at a node m_i, we determine all paths that can lead from the input to that node. Let $p(m_{i_1}, m_{i_2}, \dots, m_{i_m})$ denote a path from m_{i_1} to m_{i_m}. Figure 1 shows an example network configuration and

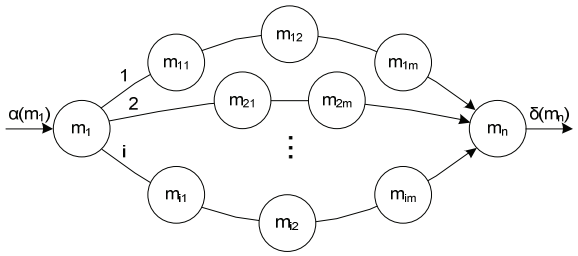

Figure 1: Validating packet flow on node m.

represents the introduced notation more clearly.

Let P_{im} be the processing time spent on node m_{im}. The total processing time on every path to node m_i can be computed by convoluting the processing time on all individual nodes in that particular path: $P_i = P_{i1} * P_{i2} * \dots * P_{im}$. If path i receives an f_i percentage of the total traffic rate then the total processing time on that path is $\sum_{i=1}^{m} f_i \cdot P_i$.

Overall, we can validate port-to-port packet flow by checking if the following equality holds:

$$\alpha(m_1) * (\sum_{i=1}^{m} f_i \cdot P_i) = \delta(m_n) \quad (1)$$

Thus, the effect of the processing delay incurred on the input traffic distribution by all intermediate modules is directly reflected on the output traffic distribution. By monitoring this distribution, deviations from normal processing behavior can be detected.

3. SUMMARY

In this work, we present a modular technique for validation of a router's correct operation based on processor monitoring. We track the processing time distribution of each module and examine input/output traffic flow characteristics. By correlating the input traffic distribution with the processing time spent on individual modules, we can verify if the output traffic distribution is the expected one.

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)* (San Diego, CA, Oct. 2010).

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

[3] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. The Click modular router. *ACM Transactions on Computer Systems 18*, 3 (Aug. 2000), 263–297.

[4] Wolf, T. In-network services for customization in next-generation networks. *IEEE Network 24*, 4 (July 2010), 6–12.

[5] Wu, Q., Chasaki, D., and Wolf, T. Implementation of a simplified network processor. In *Proc. of IEEE International Conference on High Performance Switching and Routing (HPSR)* (Richardson, TX, June 2010).