

A Computation-Communication Sequencing Model for Intrusion Detection Systems

Brian J. d’Auriol

Department of Computer Science
The University of Texas at El Paso
El Paso, TX, USA 79968
dauriol@acm.org

Kishore Surapaneni

Department of Computer Science
The University of Texas at El Paso
El Paso, TX, USA 79968

***Abstract**—A Computation-Communication Sequencing model for network-based computer attacks is proposed. Simplicity of abstraction is achieved by concentrating exclusively on the computation and communication processes involved in an attack. This paper presents preliminary studies resulting from our approach. Applications to port scanning, SYN flood attack and buffer overflow attack are given. The proposed model is best suited as a part of additional system profile models were a realistic IDS based on this approach to be constructed.*

Keywords: Intrusion detection, Attack profiling.

I. INTRODUCTION

Recent cybercrime statistics [1] together with media reports indicate that Internet-based intrusions are prominent. To detect network-based intrusions, we propose to: a) profile normal behaviors of computation and communication processes together with the information flows between them, and b) profile intrusive behaviors of network-based attacks in a network and cluster computing environment. This paper emphasizes the latter in the context of a model aimed at both. The focus of this paper is to present an attack profile model aimed at modeling informational flow between computational and communication processes. The modeling of computation and communication processes for secured communications has appeared in the literature [2].

The profiling of network-based attacks has been used in intrusion detection. User profiling forms the basis for much of the anomaly-based detection methods [3]. In this type of profiling, user behavior preferences and patterns are first determined and then are compared with run-time user behavior. System profiling describes system responses and

constructs a profile of expected system performance. Deviations from expectations flag suspicious activities [3], [4].

This paper is organized as follows. The Computation-Communication Sequencing Model is proposed in Section II together with application examples. Conclusions are given in Section III.

II. COMPUTATION-COMMUNICATION SEQUENCING MODEL

The Computation-Communication Sequencing Model models the computational and communication processes and the informational flow between them. It is based on system profiling. For purposes of this paper, a communication process consists of actions that solely pertain to the point-to-point transmission of information; a computation process consists of actions which are not communications.

Let $G = (V, E)$ where $V = \{CP_i, CM_j\}$ for a computation process CP and a communication process CM , a directed edge $e \in E$ connects either a CP_i to a CM_j or a CM_j to a CP_i . Let P be a property set $P = \{p_k\}$, P^{CP} denotes properties of a computation, similarly, P^{CM} denotes properties of a communication. Hence, $CCS = (G, P)$ denotes the sequencing model. A coarseness measure is defined over a CCS ; a level l , CCS^l defines the degree of abstraction and detail in each CP or CM . The most general description is given by $l = 1$; each CP^1 or CM^1 in CCS^1 may have an associated CCS^2 . In general, a hierarchy of abstraction and detail is established. A particular CCS instance is developed based on an analysis of a particular attack scenario. This general formulation provides

for a common structure in which to represent and manipulate an attack profile. Also, this abstraction provides chronological ordered sets of processes that occur during a network attack.

The following subsections present applications of the Computation-Communication Sequencing model. First, TCP and UDP port scans are modeled. Next, a SYN flood attack is described. Last, a buffer overflow attack is modeled.

A. Port Scan

Port scanning is a mechanism in which to determine the state of the computing environment and potential vulnerabilities of a possible victim's computer.

TCP ports are scanned using a two-phase operation. First, a connection is requested via a TCP SYN message sent to a specified port on a receiving computer. Second, for a successful connection, the responding computer responds with a SYN/ACK signal, otherwise, an RST signal. In the first step, the computation process CP_1 represents a probing process running on the sending computer while CP_2 represents a service associated with a TCP port on the receiving computer. CP_1 initiates the communication process CM_1 that sends and carries a TCP SYN message to CP_2 . CM_1 more exactly refers to the IP process(es) running in the network layer of the various intermediate gateways on the Internet which route the datagrams. The property set includes the contents of the TCP header, e.g. IP addresses and port numbers. Figure 1 illustrates this step. In the second step, if CP_2 is active, then a SYN/ACK message is returned to CP_1 ; however, if not, then a RST message is sent to CP_1 . The receiving computer initiates the communication process CM_2 whose task is to carry the response message. Based upon CM_2 , CP_1 extracts the potential victim's computer vulnerability information. Figure 2 illustrates this step. Hence: CCS_{TCP} consists of $CP_1 \rightarrow CM_1 \rightarrow CP_2 \rightarrow CM_2 \rightarrow CP_1$ with the property sets as shown in the figures.

UDP port scanning works by sending a packet and looking for an ICMP (Internet Control Message Protocol) 'port unreachable' response. If no response is received after several attempts, the port is assumed to be open. As before, a UDP port scan is also divided into two steps. In the first

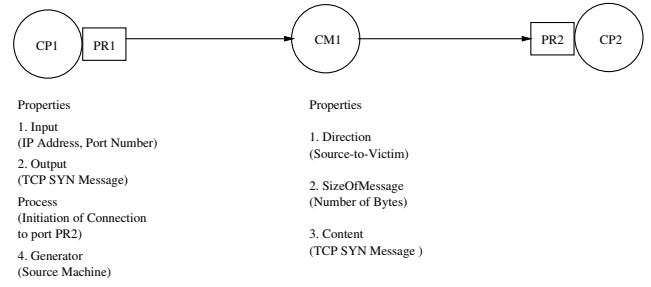


Fig. 1. TCP Probe Scan Step 1

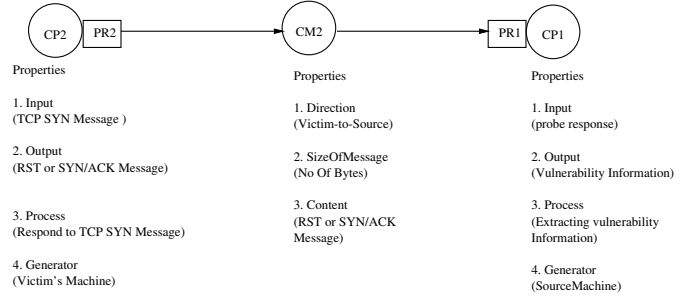


Fig. 2. TCP Probe Scan Step 2

step, CP_1 is a probing process running in the application layer of source machine. CP_1 sends UDP datagrams with 0 bytes of data to a computation process CP_2 which is running in the application layer of receiver computer. CP_2 refers to the service associated with the receiving port. UDP datagrams from CP_1 are submitted to the communication process CM_1 for physical transmission from CP_1 to CP_2 . If CP_2 is active, no response is sent back to CP_1 , otherwise, an ICMP 'port unreachable' error message is carried by the communication process CM_2 . CM_2 includes the protocol flag field, set to 1, to indicate ICMP. Hence: CCS_{UDP} consists of $CP_1 \rightarrow CM_1 \rightarrow CP_2 \rightarrow CM_2 \rightarrow CP_1$.

It is no surprise that CCS_{TCP} and CCS_{UDP} have isomorphic graphs and corresponding property sets due to the very similar nature, purpose and behavior of these events.

B. SYN Flood Attack

The establishment of a TCP connection between two communicating remote processes involves a three-way handshake protocol. A half-open connection exists when the first two steps of the handshake are completed, but the third step is not. Each half-open connection requires memory resources. Once a designated limit of such memory resources is

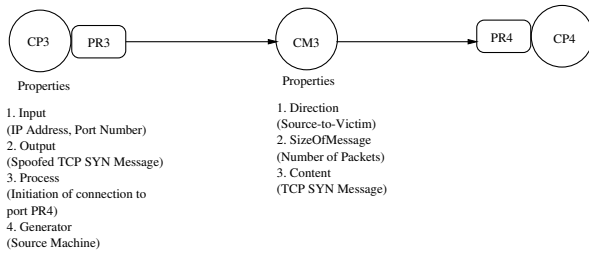


Fig. 3. TCP SYN Flood Step 1

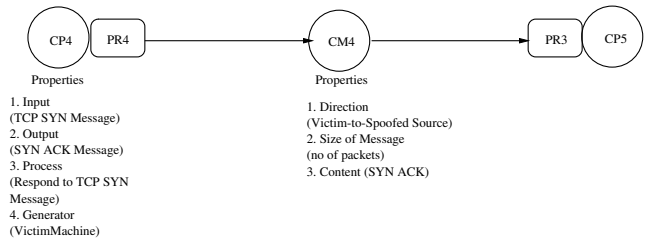


Fig. 4. TCP SYN Flood Step 2

reached, no new connection attempts are accepted. And legitimate connection requests are denied, leading to a Denial of Service.

Let CP_3 be the attack process which releases a spoofed TCP SYN packet directed towards process CP_4 running on the victim's computer. The packet is conveyed via CM_3 . At the victim's computer, the socket, Internet protocol control block (Inpcb), and the TCP control block structure (tcpcb) for this TCP connection attempt are allocated. Allocations remain in effect until cleared, for example, by timed-out mechanisms or by completion. Since the purpose is to consume resources, multiple instances occur. This step is illustrated in Figure 3.

The process CP_4 after receiving a TCP SYN packet, acknowledges the connection request with a SYN ACK packet sent via CM_4 to the spoofed source addresses given in the TCP SYN packet, and sets a timer which is typically 75 seconds. The process CP_5 executes on the third-party computer corresponding to the spoofed address. CP_5 returns an ACK packet via CM_5 to the CP_4 process thus completing the connection request. However, in an attack instance, this is unlikely. Hence, PCP_5 includes an existence probability. If CP_5 does not exist, thereby, CM_5 also not existing, a time out occurs which results in CP_4 dropping the connection and clearing the memory resources. One instance of TCP SYN Flood Step 2 takes place for each corresponding instance of TCP SYN Flood Step 1.

C. Buffer Overflow Attack

Buffer overflow is a common exploited vulnerability. There are several kinds of buffer overflow attacks. Here, a client-server approach in which the client represents the source and server represents the victim's machine is considered.

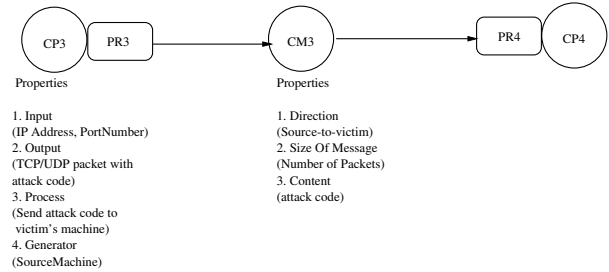


Fig. 5. Buffer Overflow Attack Step 1

Figure 5 represents the first step of the buffer overflow attack. Here, CP_3 is a buffer overflow attack process running on the client/source machine and CP_4 is a server process running on the server that provides for the buffer overflow vulnerability. CM_3 is the communication process, which carries message packet consisting of the attack code to be executed on the victim's machine from client/source machine to victim's/server machine. This attack code is stored by CP_4 in a buffer on the victim's machine for later execution.

In the second step shown in Figure 6, CP_3 sends another message to CP_4 through CM_4 . CP_4 stores this message in some unbounded stack buffer which is smaller than the message size and which is adjacent to program code pointer of CP_4 . As a result, the buffer overflows and the adjacent code

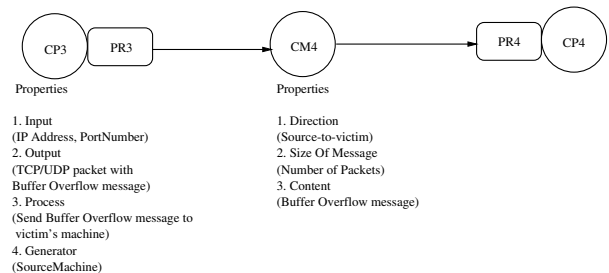


Fig. 6. Buffer Overflow Attack Step 2

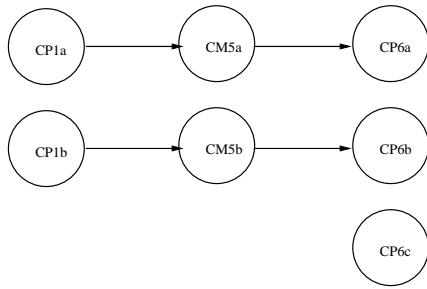


Fig. 7. A second level model for buffer overflow attack

pointer value is overwritten, in this case, with the address of the starting statement of the attack code stored on victim's machine from the previous step. As a result, the next instruction to be executed by CP_4 is the first instruction of the attack code. The execution of this attack code results in a security violation on the victim's machine, this occurs in CP_5 , not shown in the figure.

Figure 7 shows a second level CCS model for this attack. This attack has a characteristic of a single direction communication flow.

D. Integrating Multiple Sequence Models

Taken in isolation, each of the above Computation-Communication Models state rather obvious results. However, a combination of models provides for extrapolation of previous and expected attack occurrences. In and by itself, a single suspicious event detected at any of the communication processes may not indicate the full-scope of the attack. The combined models provide a mechanism to locate previous computation and communication processes in an attack stream, and, provides for heightened awareness of potential future attack detection instances. In Figure 8, network sensors labeled S1 are likely to detect a SYN Flood. Hence, at that point, auditing of CM_1 and CP_2 processes could establish the existence of a preliminary port scan. Also, increased monitoring of CM_5 and CP_6 could help track or prevent subsequent attacks.

III. CONCLUSIONS

A simple Computation-Communication Sequencing model for network-based computer attacks is proposed in this paper. Simplicity of abstraction is achieved by concentrating exclusively on

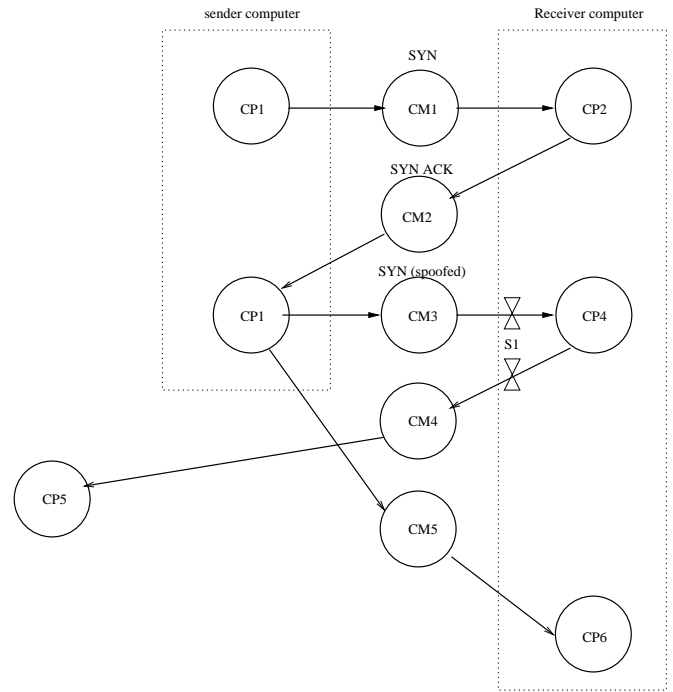


Fig. 8. Integrated port scan, SYN flood and buffer sequencing models

the computation and communication processes involved in an attack. This paper presents preliminary studies resulting from this approach. The model inherently describes information flow, but is not a strong enough model in which to represent non communication-based interactions between processes. This motivates the incorporation of the model with a state-based model [5] to enable more powerful systems profiling.

REFERENCES

- [1] "Csi/fbi computer crime and security survey," 2003, <http://www.gocsi.com/>.
- [2] I. Foster, N. T. Karonis, C. Kesselman, G. Koenig, and S. Tuecke, "A secure communications infrastructure for high-performance distributed computing," in *Proc. of the 6th IEEE Symp. on High-Performance Distributed Computing*, 1997, pp. 125–136. [Online]. Available: <http://www.globus.org/documentation/incoming/secure.pdf>
- [3] A. K. Jones and R. S. Sielken, "Computer system intrusion detection: A survey," Feb. 9, 2000. [Online]. Available: <http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-%survey-v11.pdf>
- [4] R. Balupari, B. Tjaden, S. Ostermann, M. Bykova, and A. Mitchell, "Real-time network-based anomaly intrusion detection," pp. 1–19, 2003.
- [5] B. J. d'Auriol, "Network vertical intrusion model (NetVIM)," in *Proc. of The 2005 International Conference on Security and Management (SAM'05)*, Monte Carlo Resort, Las Vegas, Nevada, USA, June 2005, in press.