

Information System Security Operation

Guaranteed Internet Stack Utilization

Guaranteeing Network Accessibility During Attack

Overview

The goal of Guaranteed Internet Stack Utilization (GINSU) is to guarantee network accessibility by an end-host, even in the event of an attempted denial of service or resource exhaustion attack. To provide this guarantee of accessibility, GINSU shifts the paradigm from complying with network requests to protecting the application and network from each other. GINSU augments an existing operating system's network stack with fine-grained resource monitoring and controls. GINSU also includes tools and guidelines for administration and deployment.

Solution

To provide a guarantee of accessibility, GINSU involves a paradigmatic shift in end-host behavior. Formerly, the network stack acted in eager complaisance with the requests of the application program and the network; applications and network traffic were assumed to be safe and well behaved and the stack's responsibility was to merely "be liberal in receiving and conservative in sending". With the advent of Distributed Denial of Service (DDOS) attacks, Code Red type worms, and other malicious network agents, it is no longer the case that both the network and the application are well behaved. Indeed, the Code Red outbreak shows that a compromised application and the passive network can both be seen as malicious agents.

GINSU shifts the paradigm of trusting the network and the application, to one of protecting the application and the network from each other. To this end we have re-architected an existing operating system's network stack based on a stack-slice paradigm, with flow-based, fine-

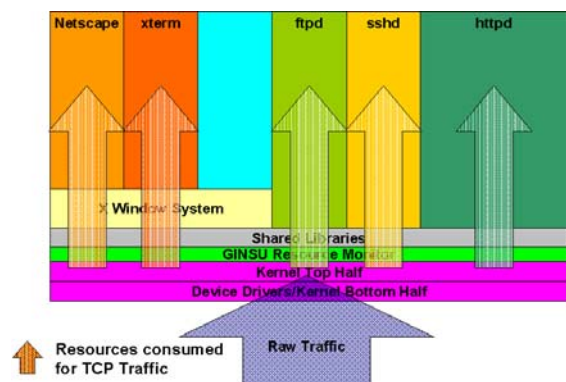
grained resource monitoring and management features. We have incorporated advanced traffic categorization mechanisms early into the processing stream. We provide tools and guidelines for administration and deployment of this system to defeat contemporary attacks, both by protecting the application from hostile network activity and by protecting the network from compromised applications.

Approach

GINSU is implemented as a set of Linux kernel modules and administration tools. The McAfee® WebShield E250 is our demonstration platform, although our code is portable to any recent Linux distribution. GINSU provides an interface for differentiating traffic sources, sinks, producers, and consumers using a number of different identification mechanisms. The data structure is characterized as the slice (hence the Ginsu Knife™ allusion). All traffic entering and leaving a host is assigned to a slice based on a slice specifier. These include IP Address (with subnet masks), TCP or UDP port, process name, and user ID.

Slices are the targets of resource limits and reservations. Resources include bandwidth (in bits per second increments), connections (in both rate per second and aggregate), CPU utilization (fractions of allotted time consumed).

GINSU Slices Partition Resources to



This work sponsored by DARPA through Air Force Research Laboratory, Contract Number F33615-01-C-1973, with McAfee Research, which is now the Security Research Division of SPARTA.

Slices exist in a hierarchical structure, in which, for example, a process which has children can be charged for the aggregation of resources used by the process group. The slice/resource collection is a complete “DAG” (Directed Acyclic Graph) capable of representing extremely complex relationships between users, connections, and networks.

GINSU allocates network traffic queues on a per-slice basis, and performs early demultiplexing of traffic (prior to TCP/IP stack processing) in order to (a) properly “charge” the intended receiving slice for the resources consumed by processing the traffic, and (b) discard traffic received in excess of resource limits as early as possible.

GINSU leverages the concept of “Lazy Receiver Processing” (LRP) developed by Peter Druschel at Rice University. LRP enables us to expend a minimum amount of packet processing under the “System” task, and properly charge all kernel resources (CPU time, memory, “objects”) consumed by the kernel on behalf of receiving processes.

Finally, GINSU integrates into the Linux “Traffic Control” and IPTABLES management interfaces. Administrators who are already familiar with the tools used for firewall and bandwidth management on a Linux platform will have no difficulty managing the GINSU resource reservations and slice specifications.

Demonstration

Our demonstration illustrates a typical application of a WebShield-like device as a boundary security gateway with a network-accessible management interface. We show how such a device can be used to protect a subnet by inspecting web and messaging traffic, but must often be “over engineered” (provisioned far in excess of typical capacity) to guarantee service levels. For instance, the demonstration shows that a gateway device can not easily ensure service levels to satisfy both client throughput and management interaction while under heavy loads.

The benefit of the GINSU processing, as demonstrated, reveals that for a given level of hostile (or unwarranted) traffic, the same hardware can appear both more responsive and more tolerant of load spikes. Additionally we demonstrate how, without the GINSU slice isolation features, hostile traffic can adversely affect traffic through the gateway and interfere with a protected client’s use of the network. GINSU, through its use of “Lazy Receiver Processing” and “Per-Slice Queues” is able to efficiently and effectively shed excess traffic based on administrator-applied limits and reservations, before that traffic is able to consume sparse resources on the gateway device.

Demo Topology

