



Active Countermeasures Research

Virus throttle

Most recent computer worms have had one thing in common. The disruption and headaches have come not just from the worms themselves, but also from the amount of network traffic they generate as they try to spread.

Electrical appliances have a fuse or circuit breaker that lets them draw enough current to operate normally, but cuts the power when the current becomes dangerous. What if there were something similar for network traffic, something that lets systems behave normally, but stops them from "going wild"?

There are many varieties of computer worm. Some try to damage the host systems; others take control of it for later use. Some even try to repair or reverse the damage caused by other worms. Regardless of purpose, they all have one thing in common: they try to propagate themselves as fast and as widely as they can over the network. The blizzard of network traffic that results is usually the first sign of an infection, and often is its most damaging effect, often hampering efforts to deal with the infection.

This frantic activity contrasts sharply with the normal behaviour of most systems. Consider what typical users of office computers do. They work on a few files held on a network server. From time to time they check their email, look at a couple of web sites, or exchange instant messages with friends. In any period of a few minutes, a normal system contacts a handful of systems, while a worm-infected system tries to contact thousands.

React to unwanted behaviour Our "throttle" can detect and react to this change from normal to infected behaviour. Located in a system's low-level networking code, it keeps track of a small number of recently used network addresses; when the system tries to contact an address not in the list, the module delays it for a second. If the number of connections on hold becomes too large, the throttle "trips" like a circuit breaker, blocks further traffic from reaching the network, and notifies the user.

Fast and effective We have tested the throttle in simulations and with a test network of several dozen systems. In normal situations, the module is not detectable among the usual delays seen on a network. When a worm such as Code Red or Nimda strikes, however, the throttle acts quickly enough to prevent malignant packets from reaching the network. The worm's attempt to spread is blocked, and the network remains free of spurious traffic.

A new line of defence The throttle complements typical anti-virus and personal firewall systems. Anti-virus systems act rather like customs and security officials at airports, looking for and arresting known suspects or contraband. Personal firewalls act like guards on doors, deciding whether to let particular programs or users pass through. The throttle, on the other hand, reacts solely to undesirable behaviour, regardless of who or what is causing it. It adds a significant new capability to a system's defences.

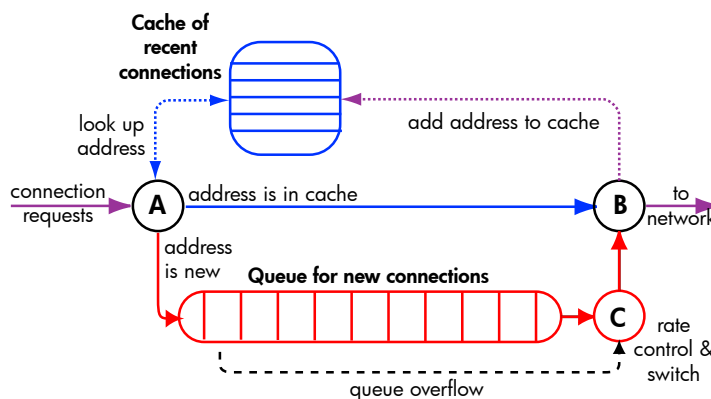


Trusted Systems Laboratory – HP Labs, Bristol UK

A Closer Look

The connection behaviour of normal systems shows a characteristic “locality”: during any short interval, the set of addresses in use is relatively small and slow-changing. Locality is a common property in computer systems, widely exploited in both hardware and software design. The most common use is in caches, which boost performance by ensuring that frequently-used items are the quickest to access.

The throttle's design exploits locality in a different sense. Rather than speeding up access to known objects, it tries to slow down access to unknown ones.



In the diagram, the flow of network connection requests is from left to right. As each connection request arrives at A, the throttle checks whether the address of the target system is one recently used by checking a cache. In a typical corporate environment, the cache contains addresses of file, email, and web proxy servers. Requests for connections to recently used addresses pass straight through the throttle without further delay.

The throttle treats connection requests for other addresses differently. It holds the request and puts the destination address at the end of a queue. Normally, destination addresses in the queue are released (at C) at the rate of one per second, together with the corresponding requests. As the request passes B, the throttle adds the new address to the cache, possibly replacing an older one. In real terms, this might happen when the user loads a web page directly from a local server, rather than through the proxy.

When a system becomes infected with a worm, it typically tries to connect to a wide range of addresses. Few of these will be in the cache, so the throttle queues them. Whereas in normal use, the queue might contain one or two requests, now the queue fills almost immediately. The throttle takes this as an indication of unwanted behaviour, and switches off the flow of connection requests at either B or C. Often, the queue fills before the first packet has been released from it. The spread of the worm and the consequent storm on the network stops here.

The choice of whether to stop all connection requests (at B) or just ones to new addresses (at C) depends on a number of factors. The important thing is to balance the desirability of keeping contact with systems in the cache, possibly allowing users to save work and contact support staff, against the risk of the worm being able to target that small set of addresses.

Contact

Richard Brown
HP Laboratories
Bristol, UK
BS34 8QZ

richard.brown@hp.com

+44 117 312 8778



i n v e n t