

BBN REPORT-8385

SPIE Memory Requirements Reduction

Contract No. N66001-00-8038

December 31, 2003

Prepared for:

NSA R23, Network Traceback & Attack Attribution
National Security Agency
9800 Savage Road Suite 6534
Fort Meade, MD 20755

Prepared by:

BBN Technologies
10 Moulton St.
Cambridge, MA 02138

SPIE Memory Requirements Reduction

Charles Lynn, Walter Milliken, and W. Timothy Strayer

BBN Technologies
10 Moulton Street, Cambridge, MA 02138

{clynn, walter, strayer}@bbn.com

December 31, 2003

1 Overview

The Source Path Isolation Engine (SPIE) [1] is an architecture used to trace a single packet back to its point of entry into an internetwork that has been instrumented with the requisite technology. The technology can be deployed either within the internetwork's routers or as a "tap box" associated with the routers. The functionality results in a graph through the instrumented internetwork. This graph does not exclude any nodes through which the given packet might have passed (no false negatives), and a configurable parameter that controls the inclusion of nodes through which the packet might have passed but did not (false positives).

Given that the traceback happens after the fact, memory is required to save information for later use. The amount of memory is determined both by the acceptable level of false positive nodes (routers) in the resulting graph and by the sum of the time required to isolate the attack packet, pass it to the traceback system for processing, and for the processing to be completed. The parameters used in the papers cited above use about 0.5% of a router's bandwidth times the length of time required to complete traceback processing. The amount of memory required becomes an issue for high speed routers (e.g., that have many OC-192 links).

This paper examines tradeoffs that may reduce the amount of memory needed in an enabled internetwork. In particular, we elaborate on the following strategies:

- Secondary storage
- Network as storage
- Digest table Compression
- Digest table paging frequency
- History aging or decaying
- Attack equivalence classes
- Interface identification
- Parameter tuning

Note that the following sections use the terminology of an "attack packet" and a "victim," even though the packet of interest may not be associated with an actual attack.

2 Use of Secondary Storage

The most obvious solution to reducing the memory requirements is to increase memory to the point where that is no longer an issue. The amount of trace history memory required by SPIE is a linear function of the interval during which a traceback can be performed. Given that some of the limitations on the amount of memory that a given router or tap box can contain are due to the required power and heat dissipation, one can reduce the amount of the limiting type(s) of memory by moving the information to secondary storage devices, such as a disk. The idea of shunting the history tables to secondary media for long-term storage has been presented in the SPIE literature, but not as a means of keeping current digest tables.

Disks with capacities well above 100 GBytes are commodity items, so capacity is not really an issue. Their use, however, would require an architecture that allowed the disk data transfer rate to exceed that of the rate of digest table generation, such a RAID design. Note that the secondary storage might be external to the router or tap box, or could be located remotely if sufficient network bandwidth is available. This design provides virtually unlimited storage.

3 Use the Network

Another place to find additional memory is actually within the network itself. If the network has excess bandwidth, or spare fibers or wavelengths that are used infrequently, like for SONET protection, then one could send the (possibly compressed) digest tables to a collection point (the Collection and Reduction agents (SCARs) in the SPIE architecture). The bandwidth of the links used must exceed the rate of digest table creation. In this scenario, the longer the network path, the more bits there will be “in flight” to the collection point. Having the data bounce back and forth would also increase the amount of data in flight. While this technique may not by itself store much data in the network, it can move the location of the memory from the routers or tap boxes to the collection points, where it may be easier to provide the required power and heat dissipation capacity.

4 Lossless Compression

History memory requirements may be reduced by compressing the digest tables. Many compression techniques exist, but for our purposes, only lossless compression should be considered. The problem with compression techniques, especially run-length encoding, is that they do not work well on randomly distributed data. In the case of SPIE, the amount of compression possible might be limited by the use of universal hash functions specifically designed to spread the set bits in a digest table uniformly throughout the table.

However, the SPIE design calls for tables to be paged with a frequency based on a worst case filling of about 12.5%. This means that the the less filled a table, the longer the expected intervals of zero-bits. Long runs of zeros is good for run-length encoding algorithms. The length of the run of set bits—that is, ones—is usually fairly small since the set bits are uniformly spread throughout the table, The runs of zero bits would grow in inverse proportion to the fullness of the table.

Current digest tables must be quickly accessible during a traceback query. It may be cumbersome to be decompressing and recompressing the tables as the queries come in, but this is largely a function of the frequency of the queries. With run-length encoding, however, it should be relatively easy to check if the run of bit locations is in a run of zeros or a run of ones, so perhaps decompression is not necessarily required.

5 Usage Based Digest Table Paging

The frequency of digest table paging is based on the worst case of the router’s operating capacity. The paging frequency then ensures that the Bloom filters will maintain at least a target probability of a false positive, where the probability of a false positive is a function of the number of hash functions, the number of packets whose hashes are recorded in the digest table, and the size of the digest table.

Routers rarely run at full capacity; networks are generally designed to be over engineered.

The hardware that performs the hashing, records the hashes in the digest table, and pages the active digest table to history memory, and the size of the history memory need to be designed to handle the maximum packet rate. The nominal result is 0.5% of the aggregate router link bandwidth times the length of time allowed for an attack packet to be isolated and the traceback queries to be answered. But routers rarely run at full capacity; networks are generally designed to be over engineered. If the links are not operating at the maximum rate, then the digest tables will contain fewer than the number of packets allowed before the desired false positive rate would be exceeded. The lower the link utilization, the lower the probability of a false positive given a constant digest table paging rate. Consequently, excessive memory usage is traded to ensure that the worst case router operation will meet a false positive target probability.

One could instead perform paging based on the number of packets received rather than on a fixed time interval (derived from the maximum packet rate). In this case, the probability of a false positive would remain constant but the amount of history memory required in the system would be subject to the current router load, increasing as the router is more heavily used, and decreasing during light loads.

Memory is more effectively used, but there is one major design tradeoff to consider. If the number of pages held in memory is constant, the length of the history (that is, how far back into the past one can query about a given packet) will vary based on the router's load.

6 History Aging

The generic router-based SPIE implementation specifies a time interval after the attack packet has traversed the enabled internetwork during which a traceback is possible. The amount of memory required to perform a traceback is a multiple of the sums of the time required to identify the attack packet and the time to perform the traceback processing. The level of confidence in the resulting attack graph is independent of the time required to identify the attack packet and initiate the traceback (up to the point where some of the digest table required to answer the query has been overwritten with more recent data). If one is willing to accept a lower level of confidence for attack packets that take more time to isolate than for attack packets that are quickly isolated, then the amount of memory can be reduced (or, alternatively, the window of traceability can be extended for the same amount of memory).

One way to implement aging is to remove a configurable set of bits, say every k^{th} bit, from a digest table. When all the bits removed from a given digest are unset bits, then there is no loss of information. However, when one or more of the removed bits is a set-bit, there is information loss (this is essentially a form of lossy compression). Whether or not the removal of bits from a given digest is lossy can be determined as the bits are being removed, and the answer can be recorded using one additional bit per digest table. Note that removal of set-bits from the digest table means that false negatives are now possible for those queries whose hash outputs equal one of the removed bits. Again, the possibility of whether or not a given query may result in a false negative response can be decided since the location of the bits removed from the digest table are known—if a query has the possibility of a false negative, the traceback algorithm could just assume that the packet was seen, and continue the query at the router's neighbors. This effectively turns the possible false negative into a possible false positive, which is something that the SPIE traceback algorithm can deal with.

One could also OR together multiple digest tables. The probability of a false positive (or false negative if bits have been removed) will be higher using the merged table, but the longer length of history could be more significant. The question of how many bits should be removed to obtain the best results can be analyzed.

7 Equivalent Attack Packets

The longer that it takes for an attack packet to be isolated and presented to the SPIE traceback manager (STM), the longer the digest tables must be retained. If the time for understanding the attack and isolating a representative

packet can be reduced, then the amount of packet history can also be reduced, saving memory.

If an attack is not limited to exactly one attack packet, but instead consists of many attack packets of similar type, then the analysis required to isolate an early attack packet can be used to more quickly isolate any recurrence of subsequent packets of the same type. This is to say that the IDS, once it understands the type of attack, can be trained to look for all attack packets in that attack equivalence class.

Instead of tracing on the early attack packet, which, by the time the attack is recognized, could be quite old, the idea is to select another newer attack packet from the attack equivalence class and trace on that one. By relaxing the requirement that hard to isolate single attack packets have to be traceable (worst case scenario), the length of history that the SCARs need to retain, and thus their memory, can be reduced.

8 Interface Identification

Construction of the traceback graph is based on identification of nodes through which a given attack packet might have passed. Starting with the node through which the attack packet reached the victim, all neighboring nodes are queried to determine whether they can be pruned from the graph. At each node, the query result is either “was not seen here” or “might have been seen here.” It might be possible to use information about the specific interface through which each packet passed to reduce the rate of false positives, which in turn may reduce the amount of required memory, either at the node itself, or at neighboring nodes.

Typically, each packet traversing a node is hashed multiple times using a different universal hash function, and the resulting hashes are used as a Bloom filter. For no additional cost in memory, but increased processing cycles, one (or more) of the hashes could be applied to the concatenation of the packet and an identifier of the interface through which the packet passed. This could be either the receiving interface, or the exit interface. The query made by the traceback processor would be more complex in that the attack packet would have to be hashed multiple times, once for each possible interface, and the resulting sets of hashes would each need to be looked up in the digest table. Note that a negative response from any of the digest tables that do not include an interface identifier results in a certain determination that the packet did not pass through the node, otherwise any negative result for one of the per-interface hashes removes that interface from the attack graph. Additional analysis specific to the situation is required to evaluate whether use of per interface information can reduce the total system memory required for a given level of confidence in the resulting graph.

9 Tuning to a Specific Internetwork

The SPIE project set out to prove that it is possible within reasonable memory requirements to accurately trace packets at every hop in the network. The analysis to prove this assertion considered worst case router capacities and made some simplifying assumptions about the number of neighbors connected to the router. In fact, the mathematical analysis in [1] argues that the router’s degree (the number of neighbors) can be ignored and the memory requirements are still reasonable. This is an analytical tradeoff of memory for mathematical simplicity; being more precise about the router’s degree will make the memory requirements more precise.

Tuning a deployment to a specific internetwork can reduce the amount of memory required, either in sum over the internetwork, or within an individual system. The memory usage estimates are for the worst case. With the same simplifying assumptions, being more specific about the network’s topology can save memory.

9.1 Partial Deployment

Many networks are engineered with the highest speed router as a core that interconnect lower speed routers at the edges, which interface to the customers—a “cloud and spoke” design. Often, the paths through the high-speed routers are traffic engineered multi-protocol label switched (MPLS) paths. It may be adequate, if all of the “spokes” are SPIE capable, not to require that the high-speed routers in the cloud be SPIE capable. (This assumes that the

routers in the cloud could have not been compromised and are not themselves producing attack packets, and that transforms are unlikely at the core routers.)

Is the path through the cloud really important? Not enabling the routers in the cloud means that the advantages of a full graph are lost. Specifically, in the case of a full graph, any false positives at node that are more than one link away from the graph can not be used to eliminate the false positive as an inconsistency. These cases can no longer be treated as inconsistencies.

Thus, in general, if a partial deployment scheme is to be used, the false positive rate at the nodes that are SPIE capable must be reduced to a level lower than was acceptable when all nodes are SPIE capable. Memory is essentially moved from the high-speed core routers to the lower speed edge routers, to achieve the same level of confidence. Since the edge routers typically have lower speed links, the total amount of memory might be reduced.

This technique can also be selectively utilized at individual nodes in a given internetwork topology consisting of multiple cloudlets and other nodes. The advantages of this scheme are tightly coupled to the topology of the enabled internetwork. Common or specific candidate topologies can be analyzed to assess the benefits of selective node-by-node deployment.

Coupled with the above is the issue of how the traceback information is to be used. For example, a complete graph may not be required if the information will (only) be used to “plug the hole.” On the other hand, if the intent is to present evidence to a court, then a “gap” in the graph may be less conclusive or convincing to a jury.

9.2 Use of Layer 2 Switching

A benefit from the use of MPLS paths through routers is that the packets are being switched at layer 2 and not routed at layer 3. Consequently, there is no need for any memory to store packet transformation information for the switched traffic. Note that it is difficult to collect packet transformation information in tap boxes, so that this way to reduce memory is only effective when the functionality is implemented within a router.

10 Conclusions

Several techniques to reduce the memory requirements of a SPIE capable network have been presented. Some are generic and others depend on the specific topology of the network. The amount of memory reduction that can be achieved using each technique, and the associated costs, are topics for further analysis. Additional memory savings are possible by designing new networks, or redesigning existing networks, to meet the requirements of the topology specific techniques.

References

- [1] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., AND STRAYER, W. T. Single-packet IP traceback. *ACM/IEEE Trans. on Networking* (Dec. 2002).