Anticipatory Distributed Packet Filter Configuration for Carrier-grade IP-Networks

Birger Toedtmann, Erwin P. Rathgeb

Computer Networking Technology Group
Institute of Experimental Mathematics
Duisburg-Essen University, Germany
{btoedtmann|erwin.rathgeb}@iem.uni-due.de

Abstract. Packet filters have traditionally been used to shield IP networks from known attack flows, ususally within firewall systems connecting trusted and non-trusted network segments. As IP networks grow and tend to connect to more and more neighbor networks with unknown trust status, carrier-grade operators in particular are beginning to experience raising costs due to increasingly complex filter configurations that have to be applied to their networks, in order to maintain a desired security level. In this paper, we present a discussion on the general properties of distributed packet filter configurations and an algorithm for a simplified compilation of anticipatory static packet filter configurations in heterogenous IP networks.

Keywords: Network Security, IP Spoofing, Packet Filters, Critical Infrastructure Protection

1 Introduction

Over the past years, operators of private and public IP networks have seen an increased amount of security related incidents, ranging from the rare targeted break-in attempt to the more frequent worm and virus spread. One method to protect against these threats is to set up and maintain special traffic-examination and -blocking functions at the edges of the network. The more sophisticated class of systems providing such functions are commonly called 'firewalls', which are often not only capable of simple packet-by-packet filtering but can also handle the inspection of the content of an entire connection.

The major benefit in deploying firewalls is an organizational one: maintain one system that keeps out unwanted traffic (and the malicious content it would import otherwise) instead of individually securing hundreds or even thousands of end-systems inside the network. However, this is only reasonable in an economic sense if the border between trusted parts of the network and non-trusted parts is known and if the number of links to from one part to the other is comparatively small. Large carrier-grade IP network operators in particular are confronted with the problem that they have many interconnection points to other networks and

This work was partially funded by the Bundesministerium für Bildung und Forschung of the Federal Republic of Germany under contract 01AK045. The authors alone are responsible for the content of the paper.

must also support a very high traffic throughput at these points. This makes setting up and maintaining firewall systems at interconnection points a prohibitively costly task. Furthermore, borders are not as static any more as in the past, because when network operators grow and merge, the borders of their networks move. Nevertheless, IP carriers have an increased demand for filter functions especially to shield internal management communication driving their networks from being disrupted by denial-of-service (DoS) attacks [01,02]. To meet this demand without having to deploy a set of expensive firewalls, operators usually fall back on the capabilities of commercial off-the-shelf routing and switching platforms to filter packets. This is often done in a very simple way by configuring filter rules on interfaces line by line within the routers or switches command line interface. A drawback of this method is that it is difficult to automate, especially in heterogenous, multi-vendor environments where filter configurations often have to be adapted to meet the routing platforms specific configuration syntax: as packet filter configuration has never been standardized in IETF management working groups, many operators still maintain packet filter rule sets semi-automatically or even manually.

As a consequence, the need for a flexible mechanism that computes effective filter points (nodes and interfaces) and provides syntactically correct filter statements for the platforms within these network is growing. Our contribution in this paper is an investigation of a new method that automatically finds efficient filter placements for large, carrier-grade, IP networks with heterogenous components. We reconcile the filter-based protection against potential attack flows with anticipated network behaviour upon failure states, where independent routing plans provide resilience. Our method allows for arbitrary threat and use scenarios for a given network and incorporates the diverse, varying filtering capabilites of the nodes inside the network as well as the syntax needed to configure filtering behavior in nodes.

1.1 Related Work

In recent years, there has been a fair amount of research on packet filter configuration issues and firewall technology, however, these approaches most commonly focus either on platform/technology specific problems (developments from firewall vendors) or investigate issues that arise after filter rule sets have been applied. In particular, policy management has been researched, e.g. conflicts that may result from distributed rule sets and how to resolve them [08,05]. Although the distribution of packet filters in networks has been suggested earlier [06,09], it was, however, without incorporating the topologic effects that we investigate and describe in this paper. Automatic packet filter compilation for firewall systems has been researched [06,07], but also without considering topologic effects.

The current state in the area of automated packet filter configuration in multi-vendor environments is that there exists no Management Information Base (MIB) that allows setting filter rules via the Simple Network Management Protocol (SNMP).[1] The Common Open Policy Service (COPS), which has been

[1] The RMON MIB does provide the filter group, however the only possible action specified for RMON is capturing the packets that match a filter pattern.

developed for policy-based networking and supports the configuration of classification statements, lacks a method for defining security related actions that are not IPsec-specific [10]. Middlebox Communications (MIDCOM), Simple Middlebox Configuration (SIMCO) and NAT/Firewall NSIS Signaling Layer Protocol (NSLP) are newer standardization efforts that aim for automatic configuration of firewall functions in so-called middleboxes (usually application layer gateways) but are quite heavyweight when it comes to implementation and scaling issues [11,12,13,14]. It is thus still the best way to use the routing or switching platforms command line interface (CLI) when configuring packet filter setups.

2 Model Assumptions and Definitions

The development of packet filters has since the beginning seen many differing approaches and naming conventions. Common ground can be found on the general notion that packet filters are a combination of two functions: a *classifier* and an *action* associated with it. The classifier tries to match a packet to a predefined pattern. Usually only the packet header is inspected for this operation to ensure timely decision making. Classsifiers that additionally check a backlog or history table of connections and packets seen at a previous time are called *stateful*. If the packet header matches the specified pattern, the action assigned to it is invoked upon the packet. In our approach, only blocking actions are used, these are specified with *allow*, *permit* or *accept*, and *disallow*, *deny* or *drop*.

A classifier/action pair is usually denoted as a *filter rule*, whereas a list of such rules is known to be a *rule set*. Some vendors also refer to those rule sets as *access control lists*.[2] When a packet does not match any of the individual rules in such an (ordered) list, a default rule, also known as *filter policy* applies. A policy that implicitly drops all non-matching traffic is called *whitelisting,* whereas a policy that accepts all non-matching traffic is known as *blacklisting*. Within the remainder of this paper, we will use simple (non-stateful) disallowing filter rules and blacklisting as we only state explicit prohibits on packets that match a specific pattern. In the following, we investigate rule sets that are distributed over a subset of nodes comprising an IP network, thereby assembling a *distributed packet filter configuration* that enforces a specific global filter policy with local packet filters without requiring a deployment of singular firewall systems.

2.1 Direction-based Filtering

Adversaries have long since adapted to the existence of packet filter systems and thus have developed their own set of techniques to circumvent them as far as possible. One method is to let the injected packets just look like legitimate packets – this is possible because IP networks allow every user to craft the packets they are going to send into the network themselves. This technique has long been known as "spoofing" [15] and is still quite popular despite increased deployment of anti-spoofing mechanisms in modern access networks; mostly because these types of networks still account only for a small fraction of all vulnerable hosts

[2] E.g. Cisco, Juniper Networks.

within the Internet [03,04]. As a consequence, when activating allowing and disallowing filters on an interface of a network node, the operator faces a trade-off concerning legitimate and malicious traffic that traverses this interface: if accepting filters are active, malicios data packets crafted by the adversary to match the configured pattern within the classifier will be falsely allowed into the network. We call this a *false negative* filter decision. On the other hand, prohibitive filters that have been placed on a path where legitimate packets travel will discard them, usually terminating a favored connection. This case we call a *false positive* filter decision. An operator therefore must anticipate the directions where the legitmate and malicious packet flows will most likely come from, to minimize the costs incurred by either false positives or false negatives. This is illustrated by Fig. 1 where the alternative filter placements influence future potential damages for the operator.

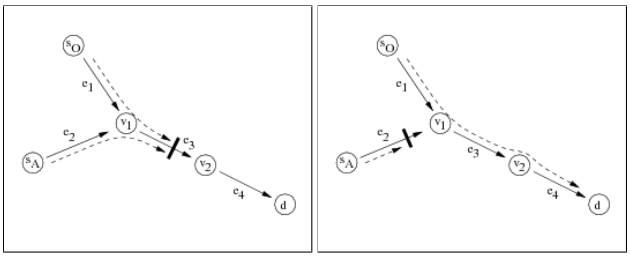


Fig. 1: Direction based filter placement. In (a), packets sent from operator node s_O and attacking node s_A that match the prohibitive filter on edge e_3 will both be discarded, resulting in a lost connection from s_O to d, but also preventing a succesful attack. In (b), the filter has been moved onto edge e_2 where it will only discard attack packets, and not legitimate user packets.

The underlying problem is thus to find the minimum costs associated with each packet filter configuration in terms of this trade-off. Formally speaking, we have

- source nodes s_O(operator) and s_A(adversary) and destination node d
- paths $p_O \in P_O$, which is the path set for all paths from s_O to d
- paths $p_A \in P_A$, which is the path set for all paths from s_A to d
- probability ω_{p_o} of a false positive case that a filter wrongly terminates a connection, this is a compound of the initial probability that this connection itself is up and that it is filtered somewhere on the path p_O
- damage D_O that is incurred if this connection to a service, e.g. SSH from management system to managed control node, is lost due to a misplaced filter
- operational risk $R_O = \omega_{p_o} D_O$
- probability φ_{p_A} of a false negative case where an attacker succeeds in sending packets to the destination node, this is a compound of the initial probability that the attacker sends packets and that he is not filtered anywhere on the path p_O
- damage D_A that is incurred by disruptions caused by an adversary on needed services, e.g. an overloaded SSH port within a control node due to a missing filter
- attack risk $R_A = \omega_{p_A} D_A$

As the costs in terms of the above risks are disjoint for all possible filter configurations, because either a prohibitive filter has been placed there (in this case φ will always be 0) or an allowance filter has been placed (here, ω will always be 0), total costs are additive over the set of valid paths from s_O to d and s_A to d:

$$R_{total} = \sum \omega_{p_o} D_O + \omega_{p_A} D_A, \forall p \in P_O \cap P_A$$

The challenge now is to minimize the total risk for the operator by choosing an efficient distributed packet filter configuration.

2.2 Routing Interference

Network operators are usually more concerned with availability issues than with security issues; however, when it comes to distributed packet filter configurations, both requirements overlap significantly. As we have established in the preceeding section, the major task when trying to minimize false negatives and false positives is to reliably determine sources of legitimate and malicious traffic flows. Unfortunately, in carrier-grade networks these sources change quite frequently as network components fail and resilience mechanisms set up alternative paths. As a consequence, the probability of a specific traffic flow to appear at a specific network nodes interface also depends upon the failure probabilities of the network components and the characteristics of the resilience mechanisms in place. In IP networks, the most important resilience mechanism is its destination-based, hop-by-hop routing mechanism. It determines, based on a routing plan, within all forwarding nodes the best next hop for known destination networks. When a

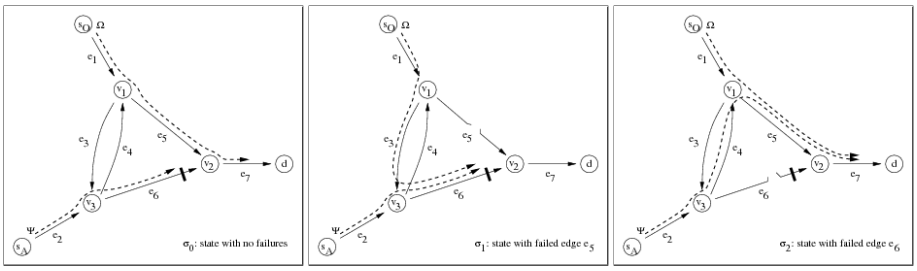


Fig. 2: Filter placement and routing interference. In this small example scenario, packets a routed in an destination-based, hop-by-hop fashion. The filter placement on edge e_6 prohibits attack packets from reaching destination node d in the failure-free situation (a), but this changes significantly when edges e_5 or e_6 fail. In (b), legitimate traffic is shifted by the routing plan onto a new path that runs over the filtering edge e_6, resulting in a lost management connection. In (c), attack traffic is wrongy detoured along a non-filtered path, allowing attack packets to reach d.

network component – a node or a link – fails, a routing algorithm adjusts to the new network state and disseminates the information about new best next hops, which are then stored in the forwarding table. The difficulty with this resilience mechanism and static packet filter configurations is illustrated in Fig. 2: when a failure occurs, packet flows may be directed over alternative paths, which may result in the wrong flows being dropped (giving a false positive) or accepted (giving a false negative). When trying to integrate a distributed filter configuration with a resilience method based on routing, one must take these trade-offs into account. The major problem is thus to incorporate the routing plan for as many network states as possible to get the corresponding path sets.

3 Distributed Packet Filter Computation

Generally speaking, we are in search for a packet filter configuration that protects our network from malicious packet flows coming from a specific attack

source. Usually, this attack source is somewhere *outside* our network, whereas the valid packet source (e.g. a management station) is somewhere *inside*. The task is therefore to find a border between outside and inside, and to find one that is *short*, to avoid placing too many filter rules and to keep the number of nodes that enforce the filters small. We are furthermore interested in a flexible mechanism that can cope with shifting security requirements such as changed damage factors or threat levels, i.e. the inital attack probability. Operators in the past simply put filters on their border routers, which is easy (there is no need to specify attack sources, probabilities and damage factors) but in many cases not efficient. The mechanism we describe here is therefore designed to compute a corresponding virtual border by minimizing the total risk as described in section 2.1 and simultaneously keeping the number of filter configurations to deploy as small as possible.

3.1 A Flexible Packet Filter Placement Algorithm

Any approach providing a way to compute direction-based packet filter configurations must incorporate a legitimate, desired usage scenario and a malicious, non-desired threat scenario in order to find a suitable border and place permits and prohibits accordingly. Each of those scenarios will be composed of a flow description and a topologic source specification which indicates from which direction a specific flow is expected to come. In our approach, both use and threat scenario correspond to the same flow description f but provide separate topologic flow source descriptions s_O and s_A. This means that the flow specifcation itself will provide the necessary information for the filter classfiers (IP source and destination addresses, transport protocol, source and destination ports) but it acknowledges that adversaries may craft attack packets that will look exactly like legitimate user packets. In constrast to this, the network specification within

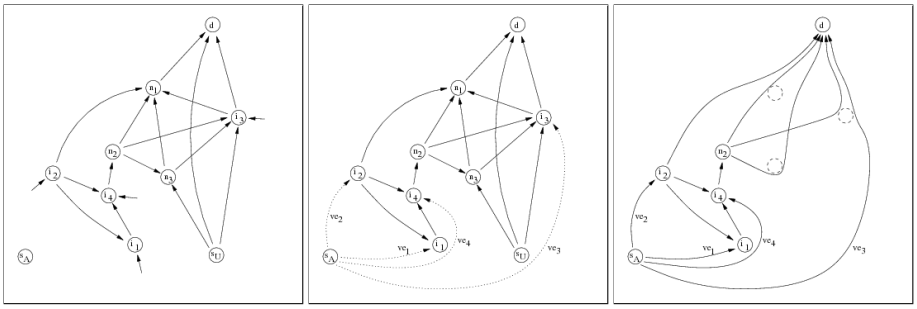
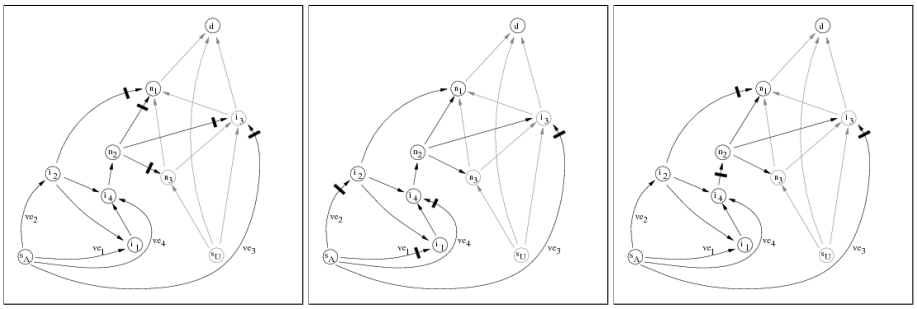


Fig. 3: In (a), we reduce an exemplary network topology to a directed graph with destination node d as the sink and edge routers - the interconnection points i_1- i_4 - together with outside attacking node s_A. In (b), the attacking node is connected to the graph at the interconnection points via virtual edges ve_1- ve_4. When the subgraph containing the dominant operational risk edges has been removed, the residual filter candidate graph emerges as shown in (c).

the use and threat scenarios will give us the needed differentiation between attack packet streams and user packet streams. This is illustrated by Fig. 3 where the valid source node is known and the operator additionally gives entry points

for potential attackers to the network – usually these coincide with the interconnection nodes towards neighbor networks.

As has been outlined in section 2.2, anticipating all paths P_O, P_A that the valid traffic and the attack traffic may take through the network is a requirement for computing where filter placements would be reasonable to reconcile resilience requirements (routing) with security requirements (filtering). Unfortunately, this generally requires a complete network state enumeration for any combination of failed elements, which is of P#-complexity. However, state space reduction is possible if the number of components per path is not too small and the availability of the components is comparatively high [16], which is a very typical characterstic of carrier-grade IP networks. Thus, in our algorithm, we first determine the number of concurrently failed elements we need to inspect, in order to reach a significantly high share of the state space. We then enumerate over all the remaining states and invoke the routing algorithm used for the network for each state, in conjunction with the legitimate use endpoints (s_O, d) and the attack endpoints (s_A, d). We thus yield all two sets of most probable paths for both sources, together with the probability by which they will be effective – this is done by combining the initial flow probability and the availability data for all components respectively. In the next step, we iterate over all paths and over all edges within the paths and add the specific probability of the selected path to the individual edge. As a result, we get a set of edges that additionally contain their legitimate use and attack probability values. Now, we are able to compute the set of *dominant operational risk edges* by comparing the individual operating risk and the attack risk of each edge – assuming that a path will always be filtered on one edge only. This edge set contains all edges where filters should never be active because the risk of wrongly terminating an important management connection is just too high, compared to the acompanying attack risk. All remaining edges of the network comprise the *residual filter candidate graph*: at any edge within this graph, a filter may be placed to prevent adversaries from injecting malicious packets, comprising a virtual border. This is illustrated by Fig. 3 (c), where all dominant operational risk edges have been removed.



(a) **(b)** **(c)**

Fig. 4: Filter placement strategies. Backward placement in (a) yields 5 filters near the dominant operational risk set of edges, while the traditional, forward placement (b) needs 4 filters at the networks borders. (c) yields the minimum number of 3 filters by exploiting a focal point inside the network for a more efficient filter placement.

Until here, we have reasoned how to assess where it is advisable to place filters and where the costs in terms of operational risk prohibits the placement of

filters. In the last step, the actual filter placements are computed. Two ways to find filter placements on the remaining graph are quite obvious: starting from the destination node, going backward over the edges for each path, placing filters as near to the part of the network over which legitime, non-filtered paths run. This approach, illustrated in Fig. 4 (a), reduces the availability of the operators connection to a minimum and provides no direct benefit. The opposite way is to place the filters as near to the attacker as possible, which is the traditional way, moving filter sets to the border of a network as depicted in 4 (b).Inspecting Fig. 4 (c), we can see that it is possible to prevent adversaries from injecting packets into our network by placing fewer filters than the traditional border-placement strategy would suggest. Thus one optimization is to compute a *minimal cutting path edge set*. Another variant of this strategy is not to minimize the total number of filters to be set up but to minimize the number of nodes where filters must be configured – which is the result of a *minimal cutting path node set*. It is easy to see in the example network that a minimal cutting path node set is $\{i_2,i_4,i_3\}$ or $\{n_2,n_1,i_3\}$. This indicates that we usually will get more alternatives here, raising the opportunity to optimize based on filter costs that can be set by the operator. If operators need to upgrade their routing platforms in order to deploy extensive packet filter setups, they may prefer to keep the number of upgrades small and they may prefer to choose the least costly upgrades: if upgrades for the nodes $n_1$and n_2 are cheaper than for the interconnection points, they will prefer the latter variant. Algorithm 1 represents a method to compute an efficient virtual border for filter placements based on the minimal filter number variant.

```
(Step 0: Extract state space Θ)
(Step 1: Extract path sets)
for all sources s_O ∈ S_O, s_A ∈ S_A do
  for all states σ ∈ Θ do
    P_O ← R_σ(s_O, d)
    P_A ← R_σ(s_A, d)
(Step 2: Compute edge utilization)
for all paths p ∈ P_O ∩ P_A do
  for all edges ϵ ∈ p do
    if p ∈ P_O then
      add availability(p) to ω_ϵ
    if ϵ ∈ P_A then
      add availability(p) to φ_ϵ
(Step 3: Compute risk distance)
for all paths p ∈ P_A do
  for all edges ϵ ∈ p do
    if ω_ϵ D_O < φ_ϵ D_A then
      filter candidate set F_c ← (ϵ, c_ϵ = 0)
      for all paths p ∈ P_A do
        if ϵ ∈ p then
          add 1 to edge candidate count c_ϵ
        choose ϵ from F_c with highest c_ϵ (it is cutting the most paths)
        for all paths p ∈ P_A do
          if ϵ ∈ p then
            P_A = P_A − p
if P_A ≠ ∅ then
  goto Step 3
```

Algorithm 1: Filter placement by creating a short virtual border

The complexity of this algorithm can be influenced quite heavily by the operator. Shortest path computations exhibit $O(m \times log(n))$ each for adverse topologies [19] and have to run over the selected state space which can amount to $O(2^n)$ if fully explored. The computations of the risk distances are of linear complexity

and a formally correct implemented minimal cut on the residual path set will run $O(\sqrt{n} \times m)$ [20], but both have to be invoked only once per f and are thus neglibible. If the state space is confined to a sensible proportion with respect to the discriminating risk function, the overall computational time for each f considered usually is within an acceptable timeframe – for the example depicted by Fig. 6, computations were well below 10 seconds for 5% of the state space. However, path exploration on the state space remains a critical issue for the method used here.[3]

3.2 Incorporating Platform Capabilities

Until now we have investigated how a distributed packet filter configuration that locally enforces a global filter policy can be compiled in an efficient way. However, operators often face the problem that platforms do not share the same capabilities, which are subject to the installed software release or acquired feature set. As a consequence, networks may include nodes that are not capable of filtering the considered packet flow f, for one of the following reasons:

- the node is technically not able to classify for f
- the node is technically not able to execute a drop or accept action on packets that match f
- the node is principally capable of filtering packets that match f, however, crucial information needed by the classifier is not available (e.g. IP addresses describing f cannot be obtained)
- the operator does not want the node to filter packets matching f, because of performance considerations

---

(Step 3: Compute risk distance)
...
if $\omega_\epsilon D_O < \varphi_\epsilon D_A$ **and** $\epsilon \in C$ **(the set of filter capable edges) then**
  $\Gamma_A = \Gamma_A - \epsilon$
  ...
if $P_A \neq \emptyset$ **and** $\Gamma_A \neq \emptyset$ **then**
  goto step 3
if $P_A \neq \emptyset$ **then**
  Issue notice: "remaining attack paths P_A **not filtered!"**

Algorithm 2: Filter placement with capability integration

We can easily incorporate this case into our existing framework by first checking all edges in the network for their respective filter capabilities, excluding those edges that do not have their source in a node that can put outgoing filters on their respective interface and the destination in node that cannot put ingoing filters on the respective interface. The resulting filter capable edge set C is used within Algorithm 2 to avoid edges where filters cannot be set up. If the attack path set cannot be emptied after all candidate edges have been investigated, the user will be noticed that an open attack path still exists.

[3] A more exhaustive discussion on algorithmic complexity and running times for test topologies is beyond the scope and limited space of this paper.

3.3 Prototype Implementation

The algorithm described above was implemented as a component of a larger management process that takes global access policies for packet flows and converts them to local, platform specific filter rule sets for a given network. We developed a Java application called Access Policy Configuration Point (APCP), that reads a network specification including all nodes within the network and their connections to each other as well as platform type and operating system version – a network discovery process providing this specification for a live network was also a part of the application. The user has to provide policy strings, enhanced by damage factors:
"protect <application protocol handle> from <source node/group handle> to <destination node/group handle> fp_damage <factor> fn_damage <factor>".

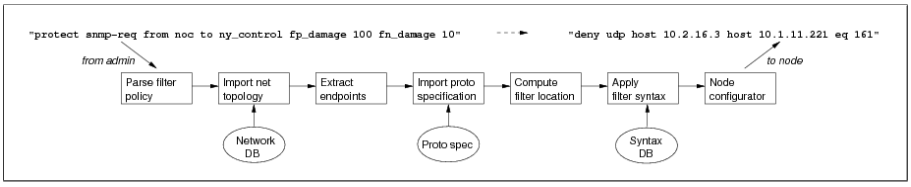


Fig. 5: APCP build process for distributed packet filters. After reading the filter policy statements and the network description as well as the threat specification, the APCP expands all endpoints and the application protocol to build the flow specification and compute the best virtual border. At the end, a syntax database is consulted to yield applicable filter statements for all platforms where filters will be configured.

When these policy specifications have been entered and the network description file, which also included a threat specification (subnet specifications for external, untrusted networks plus attack probability) has been read, the APCP expands the endpoints for the calculation of the virtual border where filter placement would be most effective. An example is illustrated in Fig. 5 where the strings "noc" and "ny_control" have been expanded to 10.2.16.3 and 10.1.11.221. In the next step the protocol handle is expanded to yield a complete flow specification f (in our example udp:10.2.16.3:*:10.1.11.221:161). Afterwards, the algorithm described in section 3 is invoked to compute an efficient filter placement for the network.

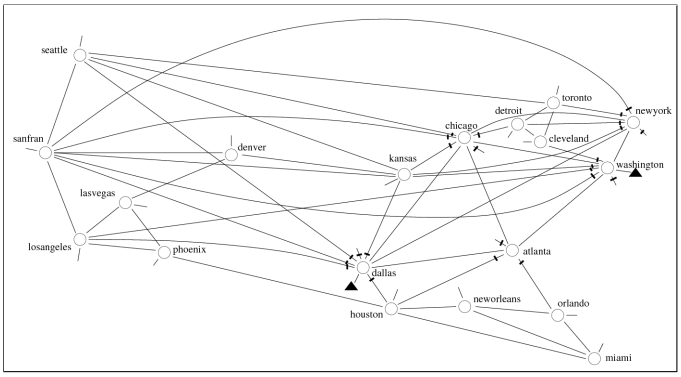


Fig. 6: Distributed filter configuration example. This case relates to a simplified topology from UUNET (1997). In order to protect a connection between a management station at Dallas and a control node at Washington, the APCP set up a new virtual border in the east, reducing the number of filter nodes significantly. The software also warned when focal routers such as Chicago were marked as non-filter capable.

The finishing steps are then to convert the information needed for the classfier into the syntactically correct format for each of the target platforms and to export the configuration statements into the nodes themselves. Currently we are able to provide conversion rules for Cisco IOS, Juniper ERX and Linux platforms. As a case study, we took a topology known from UUNET and applied a use and a threat scenario where an internal management connection has to be protected against attack sources placed at the borders, as illustrated by Fig. 6.

3.4 Discussion

Our mechanism allows a flexible computation of efficient packet filter placements along a virtual border within the network, with respect to given usage and threat scenarios and a weighting function (by damage factor). However, one might argue against *anticipatory, static* packet filter configurations we have presented here, favoring *adaptive, dynamic* packet filter mechanisms, because the best method against routing interference is, of course, to incorporate the network state and the accompanying routing decisions into the filter placement decision just-in-time. To give an example following the case depicted in Fig. 2: a drop filter should only be in placed and active in node v_2, if edge e_5 is operational. Particularly when using link-state routing algorithms, which require every node to contain a full view of the complete network state, it is possible to create ad-hoc filtering decisions on the routing information. This has also been suggested in a slightly different manner in [17]. However, several reasons can be put forward against such a mechanism:

1. In contrast to routing, operators usually do not want an automated, self-adapting filtering mechanism, simply because of the disruptive effects of drop filters.
2. When a change in the topology of the network occurs, routing algorithms always have a convergence period. Within this period, dynamically placed packet filters may wrongly discard packets (similar to the micro-loop effect seen with OSPF [18]; when using protocols such as BGP, the convergence period is even within minutes).
3. Currently, there exists no single routing or switching platform that implements dynamic, network state dependent packet filters.

Thus our approach is more suitable for the problems carrier-grade operators face currently, even with the input overhead needed in contrast to the traditional, border placement of filters. However, in terms of computational complexity, the runtime behaviour for Step 1 – the extraction of all possible path sets – still needs improvements. When not using an incremental routing algorithm [19], a full convergence for each failure condition set by Step 0 is needed, which is for networks with hundreds of nodes in the range of tens of seconds each. When this is multiplied by the number of states needed for a significant share of the complete state space, computation time reaches tens of minutes, in adverse circumstances even more than an hour.

4 Conclusions

In this paper we presented a new, flexible mechanism for computing distributed packet filter configurations for large, heterogenous IP networks. Instead of placing filters at outer borders only, our algorithm usually finds a more efficient virtual border, reducing the number of filters needed. We integrated a filter capability detection method in order to maintain a tight filter setup despite nodes not being available for filter configuration. We implemented the mechanism and enhanced it with a syntax conversion to meet platform-specific configuration demands and were thus able to demonstrate its usefulness for carrier-grade, multi-vendor environments.

For future work, we plan to evaluate the suitability of the mechanism for different topology sets and varying usage and threat scenarios. Furthermore, we will expand the set of filter actions, which are not restricted to drop and accept filters only, but can, depending on the purpose of the individual filter setup, include rate limiting, normalizing and cryptographic processing as well.

References

[01] Ferguson, P., Senie, D., "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", IETF Request for Comments (RFC) 2827, 2000
[02] Rescorla, E., Handley, M., "Internet Denial of Service Considerations", Internet Draft, IETF Spetember 2005.
[03] Pang, R., Yegneswaran, V., Barford, P., Paxson, V., Peterson, L. "Characteristics of Internet Background Radiation", Proceedings of the ACM Internet Measurement Conference, October, 2004.
[04] Dietrich, D. "Bogons and Bogon Filtering", Presentation at NANOG33, February 2005, http://www.nanog.org/mtg-0501/pdf/deitrich.pdf.
[05] Al-Shaer, E.S., Hamed, H.H., "Discovery of Policy Anomalies in Distributed Firewalls, INFOCOM 2004.
[06] Guttman, J., "Filtering Posture: Local Enforcement of Global Policies", Proceedings of the 1997 IEEE Symposium on Security and Privacy, May 1997.
[07] Bartal, Y., Mayer, A., Nissim, K., Wool, A., "Firmato: A Novel Firewall Management Toolkit", Proceedings of the 1999 IEEE Symposium on Security and Privacy.
[08] Hinrichs, S., "Policy-Based Management: Bridging the Gap", Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC '99), December 1999.
[09] Ioannidis, S., Keromytis, A., Bellovin, S., Smith, J., "Implementing a Distributed Firewall", Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS '00), November 2000.
[10] Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., Sastry, A.: "The COPS (Common Open Policy Service) Protocol", RFC 2748, IETF January 2000
[11] P. Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., Rayan, A., "Middlebox communication architecture and framework", RFC 3303, IETF August 2002.
[12] Stiemerling, M., Quittek, J., Taylor, T., "Middlebox Communications (MIDCOM) Protocol Semantics", RFC 3989, IETF February 2005.
[13] Stiemerling, M., Quittek, J., "Simple Middlebox Configuration (SIMCO) Protocol Version 3.0", Internet Draft, IETF September 2004.
[14] Stiemerling, M., Tschofenig, H., Aoun, C., "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", Internet Draft, IETF July 2005.
[15] Heberlein, L.T., "Attack Class: Address Spoofing", Proceedings of the Nineteenth National Information Systems Security Conference pp. 371-377, October 1996.
[16] Li, V.O.K., Silvester, J.A., "Performance Analysis of Networks with Unreliable Components", IEEE Transactions on Communications, Vol. 32, No 10, October 1984, pp1105-1110
[17] Park, K., Lee, H., "A Proactive Approach to Distributed DoS Attack Prevention using Route-Based Packet Filtering", Tech. Rep. CSD-00-017, Department of Computer Sciences, Purdue University, December 2000.
[18] Francois, P., Bonaventure, O. "Avoiding transient loops during IGP convergence in IP networks", IEEE INFOCOM 2005.
[19] Narváez, P. , Siu, K.Y., Tzeng, H.Y., "New dynamic algorithms for shortest path tree computation", IEEE/ACM Transactions on Networking (TON), Vol. 8, p.734-746, December 2000.
[20] Even, S., Tarjan, E.R., "Network flow and testing graph connectivity", Siam Journal on Computing, Vol. 4, p507-518, 1975.