Firewall Anomaly Detection With A Model Checker for Visibility Logic

Bassam Khorchani and Sylvain Hallé
Université du Québec à Chicoutimi, Canada
Email: bassam.khorchani@uqac.ca, shalle@acm.org

Roger Villemaire
Université du Québec à Montréal, Canada
Email: villemaire.roger@uqam.ca

***Abstract*—An anomaly in a firewall is a relationship between two of its rules that may hint at a possible misconfiguration of its filter. One notable limitation of existing solutions for firewall analysis is that they provide algorithms tailored for the verification of specific anomalies. We introduce a modal logic, called *Visibility Logic* (VL), which can be used to express arbitrary patterns between rules inside a firewall. A model checker allows one to verify any formula expressed in visibility logic, of which traditional anomalies are merely particular instances, with running times of under one second for 1,500 rules.**

Index Terms—**firewall; rules; visibility logic; model checking**

I. Firewall Rule Analysis

In order to secure a corporate network or some subnetwork within it, network traffic is usually filtered according to such criteria as origin, destination, protocol and service. Typically, dedicated routers, called *firewalls*, are equipped with *filters* specifying which packets should be forwarded and which should be discarded.

A *filter* is a sequence of *rules* that are tried in order, up to the first matching one. A rule consists of a *condition*, which is a region of the packet's space (usually consisting of source/destination IP addresses, protocol, source/destination ports), and of a *decision* (usually accept/deny). Each packet hence goes through the rules in sequence up to the first matching condition, whose decision determines whether the packet is forwarded or discarded. In our setting a filter is simply formalized by its sequence of conditions, which are axis-parallel packet space regions and a unique propositional variable a representing the *accept* decision.

Figure 1A schematic filterfigure.1 illustrates a schematic filter, where we consider only source and destination IP addresses represented in dotted decimal (four dot-separated numbers in the range $0-255$). To simplify presentation, we use a $*$ to represent the complete range $0-255$. In this figure the first rule expresses the fact that packets with any source IP address and destination in the range 132.208.100.$*$ are accepted. The second rule ensures that remaining packets with source address in the range 130.$*$.$*$.$*$ and destination in the range 132.208.$*$.$*$ will be denied. Note that the last rule ensures that all packets that match no previous rule will be denied.

Rule	Source IP addr.	Dest. IP addr.	Decision
1	*.*.*.*	132.208.100.*	accept
2	130.*.*.*	132.208.*.*	deny
3	130.*.*.*	132.208.100.*	accept
4	*.*.*.*	*.*.*.*	deny

Figure 1. A schematic filter

Configuring a filter is a well-known error-prone task. Network management researchers have therefore introduced filter properties, called *anomalies*, which either reveal or hint to a possible misconfiguration.

A. Anomalies

Network management researchers have introduced filter properties, called *anomalies*, which either reveal or hint to a possible misconfiguration. In particular, Al-Shaer's work [1], [2] considered the following cases, involving a pair of rules.

1) Simple Shadowing: A rule r_1 is *simply shadowed* if there is a rule r_2, preceding r_1 in the filter, and such that all packets satisfying r_1's condition already satisfy r_2's. In such a case r_1 applies to no traffic and is therefore either misplaced or unneeded. For instance rule 3 is simply shadowed by rule 1 in the filter of Figure 1A schematic filterfigure.1.

2) Correlation: Correlation happens when a later rule matches some packet already matched by r while having a different decision. In this case the filter is not necessarily misconfigured, but it could be useful to inform the network engineer that the second rule's decision will not apply to all packets satisfying its condition. We can formalize this property with the following formula. For instance correlation happens for rules 1 and 2 in the filter of Figure 1A schematic filterfigure.1.

3) Generalization: Generalization happens when a later rule matches more than r, but has a different decision. While generalization has legitimate uses, such as rejecting packets from some host and then accepting traffic from all remaining machines on its subnet, it could be useful here also to inform the network engineer that the rule will not apply to all packets satisfying its condition. For instance rule 4 generalizes rule 1.

4) Simple Redundancy: Finally, a rule is simply redundant if it is simply shadowed by a rule with the same decision. In this case, the rule can be removed without changing the packets that are accepted. This is the case for rule 3 which is simply shadowed by rule 1 in the filter of Figure 1A schematic filterfigure.1.

B. Beyond Simple Anomalies

These "classical" anomalies have been reported in various formulations in previous works. However, most of these anomalies only *hint* at misconfigurations, on the grounds that the administrator's intent is ambiguous. Yet, we have seen, for example, that the presence of correlated rules can be legitimate, such as when a rule is better expressed as exception intervals overriding a subsequent, more general rule. In many cases, anomalies have as much to do with policy than with correctness.

It would hence be desirable to relax some of the previous anomalies, so as not to declare an error in some specific corner cases. More generally, one may be interested in finding, for various reasons, arbitrary patterns of rules inside a firewall, and act upon them. In the following, we describe two such possible patterns.

1) Second-Degree Generalization: A first case deals precisely with the exception vs. general rule that triggers a false-positive anomaly. A firewall administrator may decide to accept "first-degree" generalization —that is, the case where an interval of addresses is given a decision that stands in exception to a general rule— and decide to ignore any warnings given by the detection of this type of anomaly.

Yet, for the sake of readability, the administrator may wish to detect and act upon any "second-degree" generalization. In such a case, a first rule gives a decision for some interval I, a later rule overrides that decision for an interval $I' \supseteq I$, and a third rule again reverses that decision for yet a wider interval $I'' \supseteq I'$.

2) Overlapping Chains: A second case is to monitor sequences of overlapping rules with alternating decisions. Such a set of three or more rules forms an overlapping "chain"; one may argue that, again, while two overlapping rules may have legitimate uses, an overlapping chain seriously decreases the legibility and any such chain be detected.

II. Visibility Logic

Since anomalies are special instances of patterns, it is natural to develop a pattern language in which they can be expressed, general enough for the cases discussed in the previous section. We present in this section our formalization of Visibility Logic, an extension of previous work on the formalization of spatial sequences [3]. We start by defining a set of binary relations suitable for distinguishing visibility-related concepts such as occlusion, obstruction and covering. We then use these binary relations to define a modal logic, and show how this language is suitable to express the anomalies described above.

A. Visibility Relations

Since we are considering visibility along a viewpoint, our basic structures are finite sequences of spatial regions, which we will simply name *spatial sequences*. In the case of firewalls, spatial regions are simply the packet space's regions induced by the restriction on port, source and destination IP address ranges.

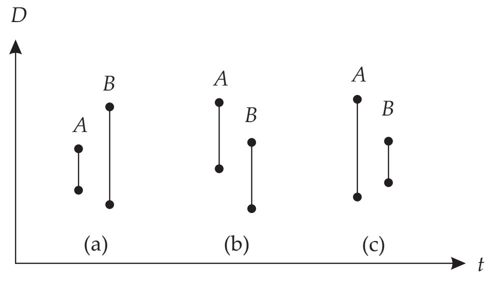

Figure 2. Occludes, obstructs and covers, with foreground on the left.

1) $\mathscr{R}, r \models P$, for some propositional variable P, if $P \in \mathscr{P}(r)$,
2) the usual definitions for the Boolean connectives $\neg$, $\wedge$, $\vee$,
3) $\mathscr{R}, r \models \bigcirc_R \varphi$, if there is a r' such that $R(r,r')$ and $\mathscr{R}, r' \models \varphi$, for r' the first such region (in sequence order),
4) $\mathscr{R}, r \models \Diamond_R \varphi$, if $\mathscr{R}, r' \models \varphi$, for some region r' satisfying $R(r,r')$,
5) $\mathscr{R}, r \models \Box_R \varphi$, if $\mathscr{R}, r' \models \varphi$, for all regions r' satisfying $R(r,r')$,

Table I
THE DEFINITION OF VISIBILITY LOGIC OPERATORS.

One can view each firewall rule as being "stacked" beneath the previous one, with an observer standing on top of the stack and looking downward. The first rule of the filter (rule #1) can hence be seen as lying at the extreme foreground of the observer's viewpoint. A packet trickles down the stack until it hits one of the rules' spatial region. If we consider two regions in isolation, the major aspects are the visibility of the background region and to what extent the foreground region obstructs this background region. If A is in the foreground of B, we can distinguish the following three conditions.

- *A occludes B*, when $A \subseteq B$ (Figure 2Occludes, obstructs and covers, with foreground on the leftfigure.2 a),
- *A obstructs B*, when $A \cap B \neq \emptyset$ (Figure 2Occludes, obstructs and covers, with foreground on the leftfigure.2 b),
- *A covers B*, when $A \supseteq B$ (Figure 2Occludes, obstructs and covers, with foreground on the leftfigure.2 c).

Let us denote these relations by $R_{\subseteq}(A,B)$, $R_{\cap}(A,B)$ and $R_{\supseteq}(A,B)$ respectively. Note that these relations are not mere set-theoretical notions, since the A parameter must be in the foreground of B.

B. Syntax and Semantics

In order to formalize spatial sequence properties, we introduce in this section a modal logic, which we call *Visibility Logic* (VL). From a logical point a view, a *spatial sequence* $\mathscr{R}$ on a *space* $\mathscr{S}$ is a sequence of subsets of $\mathscr{S}$, that we will simply call *regions*, where every $r \in \mathscr{R}$ is labelled by a set of propositional variables $\mathscr{P}(r)$. In the present case, we need only one propositional variable, a, which is true exactly when the rule's decision is "accept".

Visibility Logic is the multi-modal logic on the binary relations $R_{\subseteq}$, $R_{\cap}$, $R_{\supseteq}$ and $R_{\parallel}$. We introduce for each of these relations R, modal connectives $\bigcirc_R$, $\Diamond_R$ and $\Box_R$. This logic is interpreted on spatial sequences as defined in Table IThe definition of Visibility Logic operatorstable.1.

Note that according to the definitions of our relations, if $R(r,r')$ holds then r appears before r' in the spatial sequence. We therefore have that the semantics of $\bigcirc_R$, $\Diamond_R$ and $\Box_R$ refer to regions appearing later in the spatial sequence. We furthermore also consider converse connectors $\bigcirc_R^{-1}$, $\Diamond_R^{-1}$ and $\Box_R^{-1}$, in order to speak of previous regions.

To simplify notation, we will write $\bigcirc_\subseteq$, $\Diamond_\subseteq$ and $\Box_\subseteq$ instead of $\bigcirc_{R_\subseteq}$, $\Diamond_{R_\subseteq}$ and $\Box_{R_\subseteq}$. Similarly we will use $\bigcirc_\supseteq$, $\Diamond_\supseteq$, $\Box_\supseteq$ and $\bigcirc_\cap$, $\Diamond_\cap$, $\Box_\cap$.

C. Revisiting Firewall Anomalies

Equipped with this language, we can now revisit the anomalies described earlier and show how they are properties definable in Visibility Logic. This, in turn, entail that checking for an anomaly on a given rule-based filter reduces to a process called *model-checking* for the special logic we defined. Visibility logic hence offers a uniform approach to anomaly verification, instead of relying on algorithms tailored to specific anomalies.

1) Simple Shadowing: If we let $\mathscr{R} = r_1, r_2, \dots r_n$ be the set of rules in a given firewall, then r_i is simply shadowed if and only if the following expression holds:

$$\mathscr{R}, r_i \models \Diamond_\supseteq^{-1} \top$$

The $\Diamond_\supseteq^{-1}$ operator applies to all rules r preceding r_i, such that $r \supseteq r_i$. The symbol $\top$ denotes the constant "true"; therefore, the expression will return true whenever there exists a rule preceding r_i that covers it. This is indeed the definition of simple shadowing.

2) Correlation: Using the same notation, r_i is correlated if the filter satisfies the following expression:

$$\mathscr{R}, r_i \models (a \wedge \Diamond_\cap \neg a) \vee (\neg a \wedge \Diamond_\cap a)$$

This expression can be broken down into two alternative cases. The left-hand side of the $\vee$ connective states that the decision of r_i is "accept", and that there exists a rule r following r_i, such that r_i overlaps r and r's decision is "deny". This occurs when r is correlated with r_i. The right-hand side of the $\vee$ connective covers the case where r and r_i's decisions are reversed.

3) Generalization and Simple Redundancy: Generalization and simple redundancy are formalized in almost the same way as correlation, the difference lying in the visibility relationship used to express the rule ($\subseteq$ and $\supseteq$ instead of $\cap$), and the direction in which to look (backwards in the case of redundancy). This yields the following two expressions, which we will not describe in detail.

$$\mathscr{R}, r \models (a \wedge \Diamond_\subseteq \neg a) \vee (\neg a \wedge \Diamond_\subseteq a)$$

$$\mathscr{R}, r_1 \models (a \wedge \Diamond_\supseteq^{-1} a) \vee (\neg a \wedge \Diamond_\supseteq^{-1} \neg a)$$

4) Second-Degree Generalization: The interesting point of using Visibility Logic for expressing constraints on spatial sequences is that one is not restricted to the classical anomalies already studied in earlier works. Hence the second-degree exception anomaly we described earlier can also be written as a VL formula. Formally, a rule r is a second-degree exception if and only if the following holds:

$$\mathscr{R}, r \models a \wedge \Diamond_\supseteq^{-1} \left(\neg a \wedge \Diamond_\supseteq^{-1} a\right)$$

Formally, the expression says that r's decision is "accept", and that there exists a previous rule r_i such that $r \supseteq r_i$ whose decision is "deny", which itself has a previous rule r_j such that $r_i \supseteq r_j$ and whose decision is "accept". In such a case, r is indeed a second-degree exception. We omit the mirror case where all decisions are reversed.

5) Overlapping Chains: The second relaxed anomaly we introduced, overlapping chains, can also be expressed in Visibility Logic as follows:

$$\mathscr{R}, r \models a \wedge \Diamond_\cap (\neg a \wedge \Diamond_\cap a)$$

This formula is true whenever r's decision is accept, there exists a subsequent rule r_i overlapping r with an opposite direction, which in turn has a subsequent rule r_j overlapping with again an opposite direction. The end result is a chain of three rules that overlap each other with alternating decisions.

III. A Model Checker for Visibility Logic

To assess the feasibility of the approach, we implemented a model checker for Visibility Logic and performed tests on sample rule bases. The definitions in Table IThe definition of Visibility Logic operatorstable.1 provide a direct, recursive algorithm to evaluate any VL formula. The truth value of the top-level connective of a formula is computed recursively by combining the truth value of its child operators.

For Boolean connectives the process is straightforward; the case of $\mathscr{R}, r \models \bigstar_R \varphi$, where $\bigstar \in \{\Diamond, \bigcirc, \Box\}$ and $R \in \{\cap, \subseteq, \supseteq\}$, is handled in the following way. One first computes, given the current rule r, the list of all rules $\{r_{j,1}, r_{j,2}, \dots\}$ such that $R(r,r')$ holds. If $\bigstar$ is the operator $\bigcirc$, the expression is true exactly when $\mathscr{R}, r_{j,1} \models \varphi$. If $\bigstar$ is the operator $\Diamond$, the expression is true exactly when $\mathscr{R}, r_{j,k} \models \varphi$ for some $r_{j,k}$. Finally, if $\bigstar$ is the operator $\Diamond$, the expression is true exactly when $\mathscr{R}, r_{j,k} \models \varphi$ for all rules $r_{j,k}$.

The VL model checker is a direct implementation of this algorithm in Java. It takes as input a VL formula φ, a rule base $\mathscr{R}$ and a starting rule r, and returns as output "true" or "false" depending on whether $\mathscr{R}, r \models \varphi$. The model checker was tested using four different rule sets, produced in the same way as described in [1]. We measured the time required to evaluate all the formulæ described in the paper, for rule sets ranging from 100 to 4,000 rules. The results are plotted in Figure 3Model checking time according to rule base sizefigure.3. It is straightforward to determine that the complexity of the aforementioned algorithm is $O(|\mathscr{R}|^k)$, where $|\mathscr{R}|$ is the size of the rule base and k is the number of nested operators in

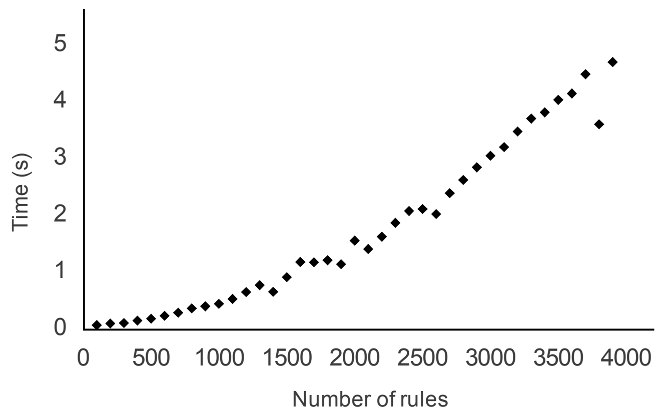

Figure 3. Model checking time according to rule base size.

the formula to verify. Since the formulæ in the paper have at most two nested operators, we observe that the approach scales quadratically to large firewall rule bases, and yields validation times under one second for as many as 1,500 rules.

IV. Related Work

The detection of anomalies in firewall rules has been the subject of a number of related works in the past. The Firewall Policy Advisor [4] is one of the earliest tools for firewall analysis. It uses Binary Decision Diagrams (BDDs) to represent rules. BDDs have also been considered in FIREMAN to represent regions and test for the shadowing, generalization, correlation and redundancy anomalies [5]. This approach has been evaluated experimentally, detecting anomalies in a 800-rule filter in less than 3 seconds.

Alternatively, a tree structure that represents a spatial decomposition of regions into non-overlapping axis-parallel regions is used in [6] in a prototype tool. Special decision tree data structures have also been introduced in [7], [8] to process sequences of regions, but with neither theoretical nor experimental evaluation.

While these tools are efficient at detecting a predefined set of firewall anomalies, their algorithms are hard-coded and built-in. To detect different patterns in rule bases, such as the second-degree exceptions and overlapping chains we described, one therefore has to modify the existing tools, and implement the algorithms that detect the desired patterns. An exception is the Margrave tool [9], which allows a user to write queries in a first-order language; however, this language is closer to a scripting language, and ultimately amounts to the user programming the desired detection mechanism. Margrave also reports running times an order of magnitude larger than the present work for checking shadowing anomalies. In contrast, our logic-based approach allows one to express arbitrary patterns over spatial sequences, and the same model checking algorithm can be used to efficiently evaluate any formula of Visibility Logic without modification.

V. Conclusion

Early experimental results on the use of Visibility Logic to the detection of patterns in firewall rules indicate that the approach is efficient. However, two factors unrelated to performance make the approach even more appealing. First, the approach is more generic than previous work: VL is general language for writing formulæ about sequences of regions, and existing anomalies studied in the literature are specific formulæ in that language. Second, while the present logic can be likened to Linear Temporal Logic in that it expresses constraints on single rulesets, it can generalize to express constraints on multiple *paths* across rulesets. This opens the way to a logic-based solution to the very relevant problem of verifying of *inter*-firewall anomalies, which is the subject of ongoing work.

References

[1] E. S. Al-Shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *INFOCOM*, 2004.

[2] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 10, pp. 2069–2084, 2005.

[3] R. Villemaire and S. Hallé, "Strong temporal, weak spatial logic for rule based filters," in *TIME*, C. Lutz and J.-F. Raskin, Eds. IEEE Computer Society, 2009, pp. 115–121.

[4] E. S. Al-Shaer and H. H. Hamed, "Firewall policy advisor for anomaly discovery and rule editing," in *IM*, ser. IFIP Conference Proceedings, G. S. Goldszmidt and J. Schönwälder, Eds., vol. 246. Kluwer, 2003, pp. 17–30.

[5] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "FIREMAN: A toolkit for firewall modeling and analysis," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2006, pp. 199–213.

[6] Y. Yin, R. Bhuvaneswaran, Y. Katayama, and N. Takahashi, "Analysis methods of firewall policies by using spatial relationships between filters," in *ICSCN*, pp. 348–354.

[7] A. Liu and M. Gouda, "Complete redundancy detection in firewalls," in *Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, ser. Lecture Notes in Computer Science, vol. 3654. Springer, 2005.

[8] M. G. Gouda and A. X. Liu, "Structured firewall design," *Computer Networks*, vol. 51, no. 4, pp. 1106–1120, 2007.

[9] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, "The Margrave tool for firewall analysis," in *LISA*, R. van Drunen, Ed. USENIX Association, 2010, pp. 1–18.