Firewall Outsourcing:
Efficiency, Verifiability, Privacy

Rezwana Reaz and Mohamed G. Gouda
Department of Computer Science
University of Texas at Austin

Abstract

A firewall system is placed at the entry point of an enterprise network to examine the packets that attempt to enter the network and decide based on some underlying firewall $F$ whether to accept or reject these packets. A firewall system usually executes $O(n)$ steps to examine each incoming packet and determine whether to accept or reject this packet, where $n$ is the number of rules in $F$. The number of rules in a practical underlying firewall $F$ can be large, for example, several thousands. Therefore, the number of steps that need to be executed by a firewall system to determine whether to accept or reject an incoming packet can be several thousand steps. However, by outsourcing parts of the functionality of the firewall system to clouds, the number of steps to be executed by the system can be reduced from $O(n)$ to $O(1)$. In this paper, we design a class of outsourced firewall systems that take advantage of clouds to reduce the number of steps to be executed by the system from $O(n)$ to $O(1)$. Unfortunately, clouds can be unreliable causing two types of attacks to our class of outsourced systems: a verifiability attack and a privacy attack. The verifiability attack is that the cloud may not execute the outsourced functionality of the firewall system correctly. The privacy attack is that the cloud may leak the rules of the underlying firewall $F$ to potential attackers of the system. To defend against these two attacks we design two classes of outsourced systems, named Verifiable, and Private Systems. The verifiable system is designed by allowing the system to observe the steps that the cloud executes and verify in $O(1)$ steps that the observed steps are the ones that the cloud should indeed execute. The verifiable system defends against the verifiability attack but does not defend against the privacy attack. The private system is designed from the verifiable system by ensuring that the cloud cannot deduce the rules of the underlying firewall $F$ and so cannot leak these rules to potential attackers of the system.

1 Introduction

A firewall is a packet filter that is placed at the entry point of an enterprise network in the Internet. Packets that attempt to enter the enterprise network through the entry point are examined, one by one, by the firewall that is placed at the entry point. Examining a packet, the firewall determines whether to allow the packet to proceed into the enterprise network or to be rejected and prevented from entering the network. Therefore, the function of the firewall is to identify malicious packets that aim to attack the enterprise network and prevent these packets from entering the network.

A firewall is a sequence of rules where each rule consists of a sequence number, a predicate, and a decision. The sequence number of each rule is a unique integer in the range from 1 to $n$, where $n$ is the number of rules in the firewall. The predicate of each rule is defined using $t$ attributes $u_1, u_2, \ldots, u_t$. The decision of each rule is either “accept” or “reject”.

An example of a firewall $F$ that consists of three rules is as follows.

\begin{align*}
1 \quad ((u_1 \in [1, 4]) \land (u_2 \in [8, 9])) & \rightarrow \text{ reject} \\
2 \quad ((u_1 \in [2, 4]) \land (u_2 \in [7, 9])) & \rightarrow \text{ accept} \\
3 \quad ((u_1 \in [1, 9]) \land (u_2 \in [1, 9])) & \rightarrow \text{ reject}
\end{align*}
Note that the predicate of each rule in this firewall $F$ is defined using two attributes $u_1$ and $u_2$ whose integer values are taken from the integer interval $[1, 9]$. The first rule in $F$ is called rule 1, the second rule in $F$ is called rule 2, and so on.

A packet $p$ is defined as a tuple $(b_1, b_2, \ldots, b_t)$ of $t$ integers where each integer $b_i$ is taken from the domain of values of attribute $u_i$.

Now consider two packets $p_1$ and $p_2$ where $p_1$ is defined as the tuple $(3, 7)$ and $p_2$ is defined as the tuple $(2, 6)$. Packet $p_1$ does not match rule 1, but matches rule 2. So the first match rule in $F$ for packet $p_1$ is rule 2. Similarly, packet $p_2$ does not match rule 1 and rule 2. Rather it matches rule 3. So the first match rule in $F$ for packet $p_2$ is rule 3.

We adopt the notation $(F, p)$ to denote the sequence number of the first match rule in firewall $F$ for a packet $p$. For example, the sequence number $(F, p_1)$ of the first match rule in $F$ for packet $p_1$ is 2 and the sequence number $(F, p_2)$ of the first match rule in $F$ for packet $p_2$ is 3.

For any underlying firewall $F$, we define a Firewall System as a system that takes as input any packet $p$ and determines whether packet $p$ is accepted or rejected according to the underlying firewall $F$.

As an example, consider a firewall system defined for the underlying firewall $F$ presented above. This system takes as input any packet $p$, for example $(3, 4)$, and determines whether $F$ accepts or rejects $p$. Because the first rule in $F$ that matches $p$ is rule 3 and because this rule has a decision “reject”, packet $p$ is rejected by the firewall system.

Consider a firewall system whose underlying firewall has $n$ rules. This firewall system determines for any incoming packet $p$ whether to accept or reject $p$ by executing $O(n)$ steps. For example, any firewall system whose underlying firewall has 3 rules, can execute $O(3)$ steps to determine for any incoming packet $p$ whether to accept or reject $p$.

The number of rules in a practical underlying firewall $F$ can be large, for example, several thousands. Therefore, the number of steps that need to be executed by a firewall system to determine whether to accept or reject an incoming packet can be several thousand steps.

To reduce the number of steps that a firewall system needs to execute for any incoming packet, one can take advantage of a cloud $C$. The underlying firewall (which has $n$ rules) of the firewall system can be outsourced to cloud $C$. Any incoming packets to the firewall system is directed to cloud $C$. Cloud $C$ executes $O(n)$ steps to examine an incoming packet against the rules of the outsourced firewall and can determine whether to accept or reject the packet. Because cloud $C$ is not technically part of the firewall system, the firewall system ends up executing only $O(1)$ steps to determine whether to accept or reject any incoming packet.

Unfortunately, the added cloud $C$ is occasionally unreliable, causing two types of attacks: a verifiability attack and a privacy attack.

The verifiability attack, caused by cloud $C$, can be described as follows. When cloud $C$ executes the steps to compute the sequence number $(F, p)$ of the first match rule in the underlying firewall $F$ for any incoming packet $p$, $C$ may compute a wrong value $v$. In particular, the computed value $v$ can be the sequence number for a match (but not a first match) rule in $F$ for $p$. To defend against this attack, the firewall system is required to verify in $O(1)$ steps that the computed value $v$ by cloud $C$ is indeed the sequence number $(F, p)$ of the first match rule in $F$ for $p$.

The privacy attack, caused by cloud $C$, can be described as follows. If cloud $C$ knows the rules of the underlying firewall $F$, $C$ can leak the underlying firewall $F$ to any potential attacker of the system. To defend against this attack, cloud $C$ is required not to know the whole underlying firewall $F$. Rather cloud $C$ can know only part of $F$ and can use this part of $F$ to determine whether to accept or reject any incoming packet.

All prior work on designing firewall systems that take advantage of a cloud either defend against the verifiability attack or defend against the privacy attack, but do not defend against both attacks. For example, the firewall systems presented in [1] and [9] defend against the verifiability attack, but do not defend against the privacy attack. Also the firewall systems presented in [2–4,6–8] defend against the privacy attack but not the verifiability attack.
Our ultimate objective in this paper is to design an outsourced firewall system that takes advantage of a cloud and can defend against both the verifiability and privacy attacks caused by the cloud. Proofs of Lemmas and Theorems presented in this paper are presented in the Appendix.

2 Firewall Systems

In this section, we first introduce a design of a firewall system that is built on top of an underlying firewall $F$ and does not take advantage of any cloud. As shown in Fig. 1, this system consists of two units: a rule matching unit, and a decision unit. Assume that a packet $p$ attempts to pass this firewall system. First, packet $p$ is directed to the rule matching unit of the system. The rule matching unit uses the underlying firewall $F$ to compute the sequence number $\#(F, p)$ of the first match rule in $F$ for $p$. Then, the rule matching unit forwards the pair $p$ and $\#(F, p)$ to the decision unit of the system. Finally, the decision unit takes as input packet $p$ and the sequence number $\#(F, p)$ received from the rule matching unit and uses firewall $F$ to compute the decision (“accept” or “reject”) of the rule whose sequence number is $\#(F, p)$.

If the decision of the rule whose sequence number is $\#(F, p)$ is “accept”, then in this case the decision unit accepts packet $p$. Otherwise, the decision for the rule whose sequence number is $\#(F, p)$ is “reject” and in this case the decision unit rejects $p$.

The problem of this system is that it executes a total of $O(n)$ steps for each incoming packet $p$ to determine whether packet $p$ should be accepted or rejected by the system, where $n$ is the number of rules in the underlying firewall $F$.

(Note that the rule matching unit executes $O(n)$ steps to compute the sequence number $\#(F, p)$ of the first match rule in $F$ for $p$. Also the decision unit executes $O(1)$ steps to find the decision of the rule whose sequence number is $\#(F, p)$. Thus, the system executes a total of $O(n)$ steps for each incoming packet $p$ to determine whether $p$ should be accepted or rejected.)
To solve this problem, we design an outsourced firewall system that takes advantage of a cloud and executes $\mathcal{O}(1)$ steps for any incoming packet $p$ to determine whether $p$ should be accepted or rejected.

The outsourced system has two units: a rule matching unit and a decision unit. The rule matching unit is executed by a cloud $C_1$, that is not part of the firewall system, and the decision unit is executed by the firewall system itself.

Thus, the number of steps that are executed by cloud $C_1$ does not contribute to the number of steps that are executed by the outsourced system. Therefore, the number of steps executed by the outsourced system for each incoming packet $p$ to determine whether $p$ is accepted or rejected is of $\mathcal{O}(1)$.

It turns out that this outsourced system still is vulnerable to an attack caused by the fact that cloud $C_1$ is occasionally unreliable. So when $C_1$ executes the rule matching unit for a packet $p$ to get the sequence number $\#(F, p)$, $C_1$ may compute a wrong value $\nu$. In particular, the computed value $\nu$ can be the sequence number for a match (but not a first match) rule in $F$ for $p$. When $C_1$ delivers the wrong value $\nu$ to the decision unit, the decision unit cannot check in $\mathcal{O}(1)$ steps whether $\nu$ is wrong.

To defend against this attack, we design a new outsourced system, called the Verifiable System. The verifiable system has two identical rule matching units and one decision unit. The two rule matching units are executed by two clouds $C_1$ and $C_2$. We assume that neither of the two clouds is always reliable. We also assume that the two clouds cannot collude.

Since cloud $C_1$ is occasionally unreliable, then when $C_1$ computes the sequence number $\#(F, p)$ of the first match rule in $F$ for any incoming packet $p$, $C_1$ may compute a wrong value $\nu_1$.

We show in Lemma 1 below that if cloud $C_1$ produces a value $\nu_1$ and cloud $C_2$ produces a value $\nu_2$ where $\nu_1 \neq \nu_2$, then the decision unit of the verifiable system can conclude that cloud $C_1$ that produced the larger value $\nu_1$ is lying. And so the decision unit rejects packet $p$.

We also show in Lemma 2 below that if cloud $C_1$ produces a value $\nu_1$ and cloud $C_2$ produces a value $\nu_2$ where $\nu_1 = \nu_2$, then the decision unit can conclude that $\nu_1$ is equal to the sequence number $\#(F, p)$ and $\nu_2$ is also equal to the sequence number $\#(F, p)$.

The verifiable system turns out to be vulnerable to an attack caused by the fact that each cloud $C_i$ needs to know the the underlying firewall $F$ in order to execute the rule matching unit being hosted by $C_i$. Because $C_i$ is occasionally unreliable, $C_i$ can leak the underlying firewall $F$ to any potential attacker of the system.

To defend against this attack, we design another verifiable system, called the Private System. In this system, each $C_i$ does not need to know the whole $F$ to execute the rule matching unit that is being hosted in $C_i$. Rather, each $C_i$ knows only part of $F$ called the concise version $CF$ of $F$. This concise version $CF$ of $F$ is the same as $F$ except that the decisions of all the rules in $F$ are dropped. For example, if the underlying firewall $F$ is as follows:

$$
1. (u_1 \in [1, 4]) \land (u_2 \in [8, 9]) \quad \rightarrow \quad \text{reject}
$$

$$
2. (u_1 \in [2, 4]) \land (u_2 \in [7, 9]) \quad \rightarrow \quad \text{accept}
$$

$$
3. (u_1 \in [1, 9]) \land (u_2 \in [1, 9]) \quad \rightarrow \quad \text{reject}
$$

then the concise version $CF$ of $F$ is as follows:

$$
1. (u \in [1, 4]) \land (v \in [8, 9])
$$

$$
2. (u \in [2, 4]) \land (v \in [7, 9])
$$

$$
3. (u \in [1, 9]) \land (v \in [1, 9])
$$

The private system is our desired outsourced system.

In the following three sections, we elaborate on our discussion of outsourced firewall system, verifiable system, and private system that defend against the verifiability and privacy attacks caused by clouds $C_1$ and $C_2$. 

---

4
3 Outsourced Firewall Systems

In this section, we modify the firewall system presented in Fig. 1 to make it execute $O(1)$ steps for each incoming packet $p$ to determine whether $p$ is accepted or rejected according to the underlying firewall $F$. We achieve this modification by taking advantage a cloud $C$ to host the rule matching unit in the firewall system in Fig. 1. Thus, the resulting system is an outsourced system where the rule matching unit is hosted in cloud $C$. Therefore, the and the number of steps that are executed by this outsourced system for any packet $p$, to determine whether $p$ is accepted or rejected is only of $O(1)$.

As shown in Fig. 2, this outsourced system consists of two units: the rule matching unit which is executed by cloud $C$, and the decision unit which is executed by the firewall system itself. Both the rule matching unit and the decision unit use the underlying firewall $F$.

![Figure 2: A firewall system with outsourcing](image.png)

Assume that a packet $p$ attempts to pass this outsourced system whose underlying firewall is $F$. First, packet $p$ is directed to the rule matching unit which is hosted in cloud $C$. The rule matching unit uses firewall $F$ to compute the sequence number $\#(F, p)$ of the first match rule in $F$ for $p$. Second, the rule matching unit forwards the pair $p$ and $\#(F, p)$ to the decision unit. Third, the decision unit takes as input packet $p$ and the sequence number $\#(F, p)$ received from the rule matching unit and uses firewall $F$ to compute the decision (“accept” or “reject”) of the rule whose sequence number is $\#(F, p)$.

If the decision of the rule whose sequence number is $\#(F, p)$ is “accept”, then in this case the decision unit accepts packet $p$. Otherwise, the decision for the rule whose sequence number is $\#(F, p)$ is “reject” and in this case the decision unit rejects $p$.

Correctness of this outsourced system is guaranteed iff cloud $C$ which hosts the rule matching unit...
is always reliable. If cloud \( C \) is occasionally unreliable, then for any incoming packet \( p \), cloud \( C \) can compute a pair (packet \( p \), \( v \)) and send it to the decision unit where \( v \) is a sequence number for a match (but not necessarily a first match) rule in \( F \) for \( p \). The decision unit cannot detect in \( O(1) \) steps that the received value \( v \) is wrong. Thus, the decision unit may end up applying a possible wrong decision (accept instead of reject or reject instead of accept) to the incoming packet \( p \).

Based on this discussion, correctness of the outsourced system in Fig. 2 can be stated as follows.

**Theorem 1** In the outsourced system in Fig. 2 and under the assumption that the cloud in this system is always reliable, the outsourced system takes as input any packet \( p \) and determines in \( O(1) \) steps whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

### 4 Verifiable Firewall Systems

The verifiable system is obtained from the outsourced system presented in Section 3 by performing the following three modifications to that system. First, cloud \( C \) in the outsourced system is replaced by two identical clouds \( C_1 \) and \( C_2 \). We assume that \( C_1 \) and \( C_2 \) are occasionally unreliable but they cannot collude. Second, the rule matching unit that is hosted in cloud \( C \) is replaced by two identical rule matching units that are hosted in clouds \( C_1 \) and \( C_2 \) as shown in Fig. 3. Third, the decision unit in the outsourced system is replaced by a verifiable decision unit in the verifiable system. Next, we describe the tasks that need to be performed by the verifiable decision unit.

![Figure 3: Verifiable firewall system](image)

When a packet \( p \) attempts to pass the verifiable system whose underlying firewall is \( F \), packet \( p \) is first directed to each of the rule matching units hosted in clouds \( C_1 \) and \( C_2 \). Each rule matching unit uses firewall \( F \) to compute the sequence number \( #(F, p) \) of the first match rule in \( F \) for \( p \). Let \( v_1 \) denote the sequence number \( #(F, p) \) computed by the rule matching unit in cloud \( C_1 \) and let \( v_2 \) denote the sequence number \( #(F, p) \) computed by the rule matching unit in cloud \( C_2 \).
The rule matching unit in $C_1$ forwards packet $p$ and the sequence number $v_1$ to the verifiable decision unit. Similarly, the rule matching unit in $C_2$ forwards packet $p$ and the sequence number $v_2$ to the verifiable decision unit.

After the decision unit receives packet $p$ and the two sequence numbers $v_1$ and $v_2$, the decision unit checks whether the two sequence numbers $v_1$ and $v_2$ are equal. There are two possible outcomes of this check.

First Outcome: The two sequence numbers $v_1$ and $v_2$ are not equal. In this case, the decision unit concludes that the cloud, say $C_1$, that has sent the higher sequence number $v_1$ is lying and drops packet $p$.

Second Outcome: The two sequence numbers $v_1$ and $v_2$ are equal. In this case, the decision unit concludes that the two clouds $C_1$ and $C_2$ have sent the same sequence number $v_1$ and that this sequence number is $\#(F, p)$. For this outcome, the decision unit uses the underlying firewall $F$ to compute the decision of the rule whose sequence number is $\#(F, p)$. If the decision of this rule is “accept”, then the decision unit accepts packet $p$. Otherwise, the decision for this rule is “reject” and so the decision unit rejects $p$.

Correctness of the first outcome follows from Lemma 1, discussed next.

**Lemma 1** Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet. If $v_1$ and $v_2$ are not equal, then the decision unit concludes (correctly) that the cloud, say $C_1$, that has sent the higher sequence number $v_1$ is lying.

Correctness of the second outcome follows from Lemma 2, discussed below and the fact that the two clouds $C_1$ and $C_2$ are occasionally unreliable, but they can’t collude.

A cloud $C_i$ is occasionally unreliable iff when $C_i$ sends a pair $(\text{packet } p, \text{value } v_i)$ to the decision unit, then $v_i$ is the sequence number of any match rule of $p$ that is not necessarily the sequence number of the first match rule of $p$. The two clouds $C_1$ and $C_2$ can’t collude iff when $C_1$ sends the pair $(p, v_1)$ and $C_2$ sends the pair $(p, v_2)$, then $C_1$ and $C_2$ can’t guarantee that $v_1$ equals $v_2$.

**Lemma 2** Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively, have sent to the verifiable decision unit for the same packet $p$. If $v_1$ and $v_2$ are equal, then the decision unit concludes (correctly) that the two clouds $C_1$ and $C_2$ have sent the same sequence number $v_1$ and that this sequence number is $\#(F, p)$.

Based on these discussions, correctness of the verifiable system can be stated as follows.

**Theorem 2** Under the assumption that the two clouds in the verifiable systems are occasionally unreliable but they cannot collude, the verifiable system takes as input any packet $p$ and determines in $\mathcal{O}(1)$ steps whether $p$ is accepted or rejected according to the underlying firewall $F$.

## 5 Private Firewall Systems

The verifiable system that is discussed in the previous section is still vulnerable to an attack, called the privacy attack, caused by the fact that cloud $C_i$ which hosts a rule matching unit of the verifiable system, knows the the underlying firewall $F$. Because $C_i$ is occasionally unreliable, $C_i$ can leak the underlying firewall $F$ to any potential attacker of the system.

To defend against this attack, we design another verifiable system, called the Private Firewall System as follows. The private system is obtained from the verifiable system by replacing each rule matching unit that uses firewall $F$ by a rule matching unit that uses a concise version $CF$ of $F$. The concise version $CF$ of $F$ is sufficient for the rule matching unit to compute the sequence number of the first match rule in $F$ for any incoming packet $p$.

The concise version $CF$ of $F$ is the same as $F$ except that the decisions of all the rules in $F$ are removed. For example, if the underlying firewall $F$ is as follows:
to determine whether to accept or reject

In this paper, we present outsourced firewall systems that use clouds to examine each incoming packet

6 Related Work

F
C
potential attackers of the system. By contrast, no cloud

Theorem 3 Each cloud C_i in the verifiable system knows the underlying firewall F and can leak F to potential attackers of the system. By contrast, no cloud C_i in the private system knows the underlying firewall F and so cannot leak F to potential attackers of the system.

6 Related Work

In this paper, we present outsourced firewall systems that use clouds to examine each incoming packet

p
to determine whether to accept or reject p. When part of the tasks of a firewall system is executed
in a cloud, the cloud (either partially or completely) knows the underlying firewall $F$ of the firewall system. For each incoming packet $p$, the cloud determines whether to accept or reject $p$ according to the rules of the underlying firewall $F$ which is stored in the cloud. However, the cloud that executes part of the tasks of the firewall system can be occasionally unreliable which makes the outsourced system vulnerable to two types of attacks.

The first attack is that the cloud can compute a wrong decision (accept instead of reject or reject instead of accept) for some incoming packet. To defend against this attack, the firewall system is required to verify the correctness of the decision computed by the cloud for each incoming packet. This attack is called the verifiability attack to an outsourced firewall system.

The second attack is that the cloud can know the rules of the underlying firewall $F$ and can leak these rules to potential attackers of the system. To defend against this attack, the firewall system is required not to store the rules of the underlying firewall $F$ in the clear in the cloud. Rather, the rules of the underlying firewall $F$ are stored encrypted in the clouds. Therefore, the cloud does not know the rules of the underlying firewall $F$ and cannot leak them to potential attackers of the system. This attack is called the privacy attack to an outsourced firewall system.

In this paper, we present an outsourced system, called the Private System. This system takes advantage of two clouds to execute the rules of the underlying firewall $F$ and defends against both the verifiability and privacy attacks. Similarly, several firewall systems [1–4, 6–9] have been presented in the literature that take advantage of one or more clouds. Each one of these systems defends against either the verifiability attack or the privacy attack but not both. By contrast, the Private System presented in the current paper defends against both the verifiability attack and privacy attacks.

In the firewall systems presented in [1] and [9], the rules of the underlying firewall $F$ are stored in the clear in the cloud. Each incoming packet to the enterprise network is directed in the clear to the cloud. For each incoming packet $p$, the cloud determines whether to accept or reject $p$ according to the rules of the underlying firewall $F$ which are stored in the cloud. If the cloud determines to accept $p$, then the cloud forwards $p$ to the entry point of the enterprise network. Therefore, the firewall systems presented in [1] and [9] defend against the verifiability attack.

Whereas the firewall system in [9] executes the verification steps online, the firewall system in [1] executes the verification steps offline. Moreover, because the rules of the underlying firewall $F$ are stored in the clear in the cloud, the cloud can leak these rules to potential attackers of the system. Therefore, the firewall systems presented in [1] and [9] do not defend against the privacy attack.

In the firewall systems presented in [2–4, 6–8], the rules of the underlying firewall $F$ are encrypted before they are stored in the cloud. Each incoming packet to the enterprise network is directed to the cloud. For each incoming packet $p$, the cloud determines whether to accept or reject $p$ according to the encrypted rules of the underlying firewall $F$ which are stored in the cloud. If the cloud determines to accept $p$, then the cloud forwards $p$ to the entry point of the enterprise network. Because the rules of the underlying firewall $F$ which are stored in the cloud are encrypted, the cloud cannot know the rules of the underlying firewall $F$ and so cannot leak these rules to potential attackers of the system. Therefore, the firewall systems presented in [2–4, 6–8] defend against the privacy attack.

Moreover, none of the firewall systems in [2–4, 6–8] verifies that packet $p$ that has been forwarded to the entry point of the enterprise network from the cloud is indeed accepted according to the underlying firewall $F$. Therefore the firewall systems presented in [2–4, 6–8] do not defend against the verifiability attack.

7 Concluding Remarks

In this paper, we design an outsourced firewall system for an underlying firewall $F$ by outsourcing part of the functionality of the firewall system to be executed by a cloud. We show that the outsourced system executes $O(1)$ steps to determine whether to accept or reject each incoming packet. Because
the cloud used in this system is occasionally unreliable, the outsourced system is vulnerable to two
types of attacks: a verifiability attack and a privacy attack. The verifiability attack is that the cloud
may not execute the outsourced functionality correctly of the firewall system. The privacy attack is
that the cloud may leak the rules of the underlying firewall $F$ to potential attackers of the system.

We develop method to design an outsourced system that defends against these attacks in two
stages. In the first stage, we extend the outsourced system to a verifiable system that defends against
the verifiability attack. This system is designed from the outsourced system by allowing the system to
observe the steps that the cloud executes and verify in $O(1)$ steps that the observed steps are the ones
that the cloud should indeed execute. The verifiable system defends against the verifiability attack
but does not defend against the privacy attack.

In the second stage, we extend the verifiable system to a private system that solves both verifiability
attack and privacy attack. This system is designed from the verifiable system by ensuring that the
cloud cannot deduce the rules of the underlying firewall and so cannot leak these rules to potential
attackers of the system. Therefore, the private system is our desired outsourced firewall system that
takes advantage of cloud(s) to execute its steps and defends against both the verifiability and privacy
attacks caused by cloud(s).

The outsourcing techniques presented in this paper are developed for firewalls. The problem of
extending these techniques to outsource other middleboxes such as Intrusion Detection Systems (IDS),
NAT, load balancers etc merits further research. Several middlebox outsourcing techniques [4,7,9] have
been presented in the literature, but none of these techniques defends against both the verifiability
and privacy attacks that are caused by outsourcing middleboxes to cloud(s). In this paper, we develop
methods to defend against both the verifiability and privacy attacks for firewall outsourcing. However,
the problems of extending these techniques for outsourcing middleboxes that solve both the verifiability
and privacy attacks requires further research.

The model of firewalls discussed in this paper has been generalized into firewall expressions in [5].
The problem of extending the outsourcing techniques discussed in this paper to outsource firewall
expressions merits further research.

**APPENDIX**

Proofs of Lemma 1, Lemma 2, Theorem 1, Theorem 2 and Theorem 3 are given below.

**Lemma 1**

*Statement of Lemma 1*: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively,
have sent to the verifiable decision unit for the same packet. If $v_1$ and $v_2$ are not equal, then the decision
unit concludes (correctly) that the cloud, say $C_1$, that has sent the higher sequence number $v_1$ is lying.

*Proof*: Suppose $v_1 > v_2$ and so the decision unit concludes that cloud $C_1$ is lying. We prove, by
contradiction, that this conclusion reached by the decision unit is correct. Assume that cloud $C_1$, that
has sent the higher sequence number $v_1$ is not lying. This assumption is contradicted by the fact that
rules $v_1$ and $v_2$ are also match rules in $F$ for $p$ and the fact that $v_1 > v_2$, which implies that rule $v_1$
is not the first match rule in $F$ for $p$. Therefore, the conclusion, that has been reached by the decision
unit that cloud $C_1$ is lying, is correct.

**Lemma 2**

*Statement of Lemma 2*: Let $v_1$ and $v_2$ be the two sequence numbers that clouds $C_1$ and $C_2$, respectively,
have sent to the verifiable decision unit for the same packet $p$. If $v_1$ and $v_2$ are equal, then the decision
sequence number of any match rule of \( p \) that there are \( k \) match rules for \( p \) exists only one match rule for they have sent a sequence number of any of the last \( k \) numbers \( F, p \) that this sequence number is \(#(F, p)\). Assume that there are \((k + 1)\) match rules for packet \( p \) in \( F \), where \( k > 0 \). Thus, when the two sequence numbers \( v_1 \) and \( v_2 \) are equal, then either the two clouds have sent the sequence number \(#(F, p)\) or they have sent a sequence number of any of the last \( k \) match rules for \( p \). (Note that when \( k = 0 \), there exists only one match rule for \( p \) in \( F \). In this case, if \( v_1 \) and \( v_2 \) are equal, then both clouds have sent the sequence number \(#(F, p)\).)

Let \( q \) be the probability that the two clouds have sent the sequence number \(#(F, p)\). So the probability that the two clouds have sent a sequence number of any of the last \( k \) match rules for \( p \) is \((1 - q)\). The following probability analysis shows that \( q \) is 1.

Consider the following three predicates, \( E, G, \) and \( H \), as follows.

- \( E: v_1 = v_2 \)
- \( G: v_1 = #(F, p) \land v_2 = #(F, p) \)
- \( H: v_1 = i_1 \land v_2 = i_2 \), where \( i_1 \) and \( i_2 \) are randomly selected values from the set of \( k \) sequence numbers of the last \( k \) match rules in \( F \) for \( p \).

The probability of the two sequence numbers \( v_1 \) and \( v_2 \) are equal is computed as follows.

\[
P(E) = (P(E|G) \times P(G)) + (P(E|H) \times P(H)) \tag{1}
\]

where

\[
P(E) = P(v_1 = v_2) = 1,
\]

\[
P(E|G) = P((v_1 = v_2)|(v_1 = #(F, p) \land v_2 = #(F, p))) = 1,
\]

\[
P(G) = q,
\]

\[
P(E|H) = P((v_1 = v_2)|(v_1 = i_1 \land v_2 = i_2)) = 1/k, \text{ where } i_1 \text{ and } i_2 \text{ are randomly selected values from the set of } k \text{ sequence numbers of the last } k \text{ (} > 0 \text{) match rules in } F \text{ for } p,
\]

\[
P(H) = (1 - q).
\]

Substituting these probabilities into Equation 1, we conclude that \( q \) is 1 for any value of \( k \). Therefore, when the two sequence numbers \( v_1 \) and \( v_2 \) are equal, the conclusion reached by the decision unit that this sequence number is \(#(F, p)\) is correct.

**Theorem 1**

**Statement of Theorem 1:** In the outsourced system in Fig. 2 and under the assumption that the cloud in this system is always reliable, the outsourced system takes as input any packet \( p \) and determines in \( O(1) \) steps whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

**Proof:** Consider an outsourced system, shown in 2 that consists of a rule matching unit and a decision unit. Both the rule matching unit and the decision unit use the same underlying firewall \( F \). When a packet \( p \) attempts to pass this outsourced system whose underlying firewall is \( F \), packet \( p \) is directed to the rule matching unit which is hosted in cloud \( C \). The rule matching unit uses firewall
\( F \) to compute the sequence number \( \#(F, p) \) of the first match rule in \( F \) for \( p \). Then, the rule matching unit forwards the pair \( p \) and \( \#(F, p) \) to the decision unit. Next, the decision unit takes as input packet \( p \) and the sequence number \( \#(F, p) \) received from the rule matching unit and uses firewall \( F \) to compute the decision ("accept" or "reject") of the rule whose sequence number is \( \#(F, p) \).

If the decision of the rule whose sequence number is \( \#(F, p) \) is "accept", then in this case the decision unit accepts packet \( p \). Otherwise, the decision for the rule whose sequence number is \( \#(F, p) \) is "reject" and in this case the decision unit rejects \( p \).

Correctness of this outsourced system is guaranteed iff cloud \( C \) which hosts the rule matching unit is always reliable. If cloud \( C \) is occasionally unreliable, then for any incoming packet \( p \), cloud \( C \) can compute a pair (packet \( p \), value \( v \)) and send it to the decision unit where \( v \) is a sequence number for a match (but not necessarily a first match) rule in \( F \) for \( p \). The decision unit cannot detect in \( \mathcal{O}(1) \) steps that the received value \( v \) is wrong. Thus, the decision unit may end up applying a possible wrong decision (accept instead of reject or reject instead of accept) to the incoming packet \( p \).

**Theorem 2**

**Statement of Theorem 2:** Under the assumption that the two clouds in the verifiable systems are occasionally unreliable but they cannot collude, the verifiable system takes as input any packet \( p \) and determines in \( \mathcal{O}(1) \) steps whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

**Proof:** The two clouds \( C_1 \) and \( C_2 \) in the verifiable system are occasionally unreliable. Therefore, when each cloud \( C_i \) sends a pair (packet \( p \), value \( v_i \)) to the decision unit, then \( v_i \) can be the sequence number of any match rule of \( p \) that is not necessarily the sequence number of the first match rule of \( p \). We proved in Lemma 1 that when the two sequence numbers \( v_1 \) and \( v_2 \) are not equal, then the decision unit of the verifiable system concludes (correctly) that the cloud, say \( C_1 \), that has sent the higher sequence number \( v_1 \) is lying.

However, the two clouds \( C_1 \) and \( C_2 \) cannot collude. Therefore, when \( C_1 \) sends the pair \( (p, v_1) \) and \( C_2 \) sends the pair \( (p, v_2) \), then \( C_1 \) and \( C_2 \) can’t guarantee that \( v_1 \) equals \( v_2 \). Using this fact we prove in Lemma 2 that if \( v_1 \) and \( v_2 \) are equal, then the decision unit of the verifiable system concludes (correctly) that the two clouds \( C_1 \) and \( C_2 \) have sent the same sequence number \( v_1 \) and that this sequence number is \( \#(F, p) \).

Therefore, when the two clouds \( C_1 \) and \( C_2 \) send the sequence numbers \( v_1 \) and \( v_2 \) respectively, to the decision unit, the decision unit verifies in \( \mathcal{O}(1) \) steps whether \( v_1 \) and \( v_2 \) are equal and so they are equal to the sequence number \( \#(F, p) \). Moreover, the decision unit finds the decision of the rule whose sequence number is \( \#(F, p) \) in \( \mathcal{O}(1) \) steps. Thus, we can conclude that the verifiable system takes as input any packet \( p \) and determines in \( \mathcal{O}(1) \) steps whether any of the two clouds is lying and whether \( p \) is accepted or rejected according to the underlying firewall \( F \).

**Theorem 3**

**Statement of Theorem 3:** Each cloud \( C_i \) in the verifiable system knows the underlying firewall \( F \) and can leak \( F \) to potential attackers of the system. By contrast, no cloud \( C_i \) in the private system knows the underlying firewall \( F \) and so cannot leak \( F \) to potential attackers of the system.

**Proof:** Recall that the two clouds \( C_1 \) and \( C_2 \) in the private system know the concise version \( CF \) of the underlying firewall \( F \). Therefore, each cloud \( C_i \) knows the predicate of the first match rule in \( F \) for any incoming packet \( p \). But the decision of the rule whose sequence number is \( \#(F, p) \) is not known to any of the two clouds. Rather the decision of the rule whose sequence number is \( \#(F, p) \) is only known to the verifiable decision unit.

Moreover, the decision ("accept" or "reject") taken for each packet \( p \) by the verifiable decision unit is not known to any of the two clouds. Thus, knowing only packet \( p \) and the predicate of the
first match rule for $p$, neither $C_1$ nor $C_2$ can deduce the decision of the first match rule in $F$ for $p$. Therefore, the rules in the underlying firewall $F$ remain private to the private system.

References


