

US Patent & Trademark Office

Patent Public Search | Text View

United States Patent Application Publication

20250267124

Kind Code

A1

Publication Date

August 21, 2025

Inventor(s)

SANFORD; Cody et al.

SYSTEM AND METHOD FOR UTILIZATION OF FIREWALL POLICIES FOR NETWORK SECURITY

Abstract

Aspects of the present disclosure involve systems, methods, for encoding a firewall ruleset into one or more bit arrays for fast determination of processing of a received communication packet by a firewall device associated with a network. Through this bitmap, a number of computation operations needed to determine a processing rule for a received packet is significantly reduced compared to the traditional approach of using a hash or a longest prefix match technique. Rather, determining a processing rule for a received packet may include determining a bit value within one or more arrays. In one implementation, a firewall rule may be encoded into a 64-bit array of bit values in which each bit of the array corresponds to a particular processing rule for a particular network address. The firewall rule may be encoded into a bitmap array of bit values by asserting a particular bit within the array.

Inventors: SANFORD; Cody (Thornton, CO), WHELTON; Robert (Louisville, CO)

Applicant: Level 3 Communications, LLC (Denver, CO)

Family ID: 1000008589836

Assignee: Level 3 Communications, LLC (Denver, CO)

Appl. No.: 19/200580

Filed: May 06, 2025

Related U.S. Application Data

parent US continuation-in-part 18217949 20230703 PENDING child US 19200580

parent US continuation-in-part 17991322 20221121 parent-grant-document US 12095735 child US 18217949

us-provisional-application US 63283038 20211124

Publication Classification

Int. Cl.: H04L9/40 (20220101)

U.S. Cl.:

CPC H04L63/0236 (20130101); H04L63/0263 (20130101);

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATION [0001] This application is a continuation-in-part of U.S. patent application Ser. No. 18/217,949, filed Jul. 3, 2023, which is a continuation-in-part of U.S. patent application Ser. No. 17/991,322, filed Nov. 21, 2022, now U.S. Pat. No. 12,095,735, which claims priority under 35 U.S.C. § 119 (e) from U.S. Patent Application No. 63/283,038, filed Nov. 24, 2021, the contents of which are all incorporated herein by reference in their entirety.

TECHNICAL FIELD

[0002] Embodiments of the present disclosure generally relate to systems and methods for implementing a communications network, and more specifically for network firewall devices and policies associated with such firewall devices.

BACKGROUND

[0003] Providing secure communications between devices is an important component of many communication networks. Many communication network configurations therefore include one or more services to provide security to the network devices and/or devices utilizing the network to communicate. For example, a backbone or service network may offer a firewall service to customers that is configured to control or monitor incoming and outgoing traffic to/from a customer network or network device. Firewall devices are typically dispersed at entry points into/out of a network such that potentially harmful or malicious communications can be detected and, in some instances, discarded or blocked by the firewall. The purpose of a firewall is to act as a defense against malicious and unauthorized traffic from affecting a customer or secondary network, keeping the information that an organization does not want out, while allowing approved information to flow into and out of the network. Other security devices, systems, and/or services may respond to other threats such as denial-of-service attacks, phishing schemes to obtain sensitive information, malware distribution in or on the network, and the like. Through the collection of security procedures and devices, a network may be situated to respond to attacks and protect the communications within and transmitted through/from the network.

[0004] Generally speaking, firewall services maintain a list of network addresses and an associated rule for processing packets received from or intended for the corresponding network address. For example, a firewall service may receive a communication packet intended for a secondary network and obtain a source network address from the packet. The service uses the source network address to search a firewall policy to obtain a processing rule for packets from the source address.

Processing the packet may include dropping the packet, rerouting the packet, scrubbing the packet, allowing the packet to enter the secondary network, and the like. As some network addressing schemes may include billions of possible network addresses, significant transmission time may be spent in searching a firewall's entire ruleset to identify the processing rule to apply to a received packet, slowing the transmission speed of the packet into the secondary network. Techniques have been developed to improve the ruleset look-up procedure, such as through the use of hashing or longest prefix matching with a tree structure. However, these techniques still require the firewall

service to execute several computations to obtain the processing rule for a received communication packet.

[0005] It is with these observations in mind, among other, that aspects of the present disclosure were conceived.

SUMMARY

[0006] One aspect of the present disclosure relates to a method for providing a firewall service. The method may include the operations of encoding a firewall ruleset into one or more arrays comprising a string of bits, wherein each bit of the one or more arrays corresponds to one processing rule of the firewall ruleset and obtaining, utilizing a first portion of a network address included in a received communication packet, the one or more arrays from a data structure. The method may also include the operations of determining a bit value from the one or more arrays based on a second portion of the network address and processing the received communication packet based on the bit value from the one or more arrays.

[0007] Another aspect of the present disclosure relates to a network firewall device comprising a processing device, at least one interface receiving communication packets, and a non-transitory computer-readable medium encoded with instructions. When the instructions are executed, the processing device may perform the operations of encoding a firewall ruleset into one or more arrays comprising a string of bits, wherein each bit of the one or more arrays corresponds to one processing rule of the firewall ruleset and obtaining, utilizing a first portion of a network address included in a communication packet received via the at least one interface, the one or more arrays from a data structure. The processing device may also perform the operations of determining a bit value from the one or more arrays based on a second portion of the network address and processing the received communication packet based on the bit value from the one or more arrays.

[0008] Accordingly, in some embodiments, as discussed herein, the disclosed systems and methods provide a framework that extended from a binary decision framework to one capable of selecting from multiple possible actions based on IPv4 address evaluation, effectively transforming the system from a simple allow/deny mechanism to one that can map an address to any number of potential actions. Such enhanced mapping methodology provides significant advantages when applied to network prefixes, delivering runtime performance that substantially outpaces conventional approaches used in the field. The two predominant existing methods for such operations are the Longest Prefix Match (LPM) Trie, which operates with a runtime complexity of $O(N \log(2) N)$, and hash lookup, which functions at $O(C)$ where C represents a constant typically reduced to 1.

[0009] According to some embodiments, the disclosed methodology and system achieves a runtime of $O(C')$, where C' is demonstrably less than C , creating a meaningful performance improvement. It's worth noting that despite the hash algorithm's theoretical advantage in algorithmic efficiency, the constant time required for hash operations is typically long enough that the LPM Trie, despite its less efficient algorithmic runtime, often performs faster in practical applications, making it the generally preferred method in current implementations. The innovation described here is significant because the runtime of the extended method is sufficiently reduced to make it practically faster than the LPM Trie approach, representing a genuine advance in the field. Such improved method expands the system's applicability well beyond simple binary firewall policies, enabling an arbitrary number of different firewall policy decisions to be selected and applied based on IP address evaluation.

[0010] Furthermore, according to some embodiments, when combined with network prefix matching capabilities, the action determined can function as a routing decision, effectively allowing the system to serve as an IPv4 network router. Such dual functionality is particularly relevant given that contemporary advanced firewalls frequently provide both firewall policy enforcement and routing capabilities within a single implementation.

[0011] According to some embodiments, an additional application of the bitmap array structure

used in the disclosed method enables monitoring decisions where a target IP address can be associated with a specific rule that triggers the incrementation of a counter for the address. This counter information can then be logged in a separate data store, facilitating comprehensive reporting on addresses of concern that have been observed passing through the device. Such monitoring capability requires no additional explanation beyond the description provided in the patent documentation, as it follows naturally from the fundamental methodology.

[0012] Thus, the extended disclosed method represents a significant advancement in IP address-based decision making, offering improved performance characteristics while simultaneously expanding the range of possible actions beyond binary outcomes. By enabling policy enforcement, routing capabilities, and monitoring functions within a single framework, this innovation provides network security implementers with a more powerful and flexible tool for managing complex network environments. The performance improvements delivered by the disclosed mechanisms enable functionality for high-throughput scenarios where processing efficiency is critical to overall system performance. Indeed, the ability to map addresses to multiple possible actions rather than simple binary outcomes creates opportunities for more nuanced security policies and traffic management strategies. Such flexibility, combined with the performance advantages over existing methods, positions the extended the disclosed method as a valuable innovation in the field of network security and management.

[0013] In some embodiments, the application of network prefix matching capabilities further enhances the utility of the system by enabling routing decisions based on the same fundamental methodology, creating a unified approach to both security policy enforcement and network traffic direction. By consolidating these functions within a single methodological framework, the extended disclosed systems and methods can streamline implementation and reduces computational overhead compared to maintaining separate systems for security and routing functions. The monitoring capabilities enabled by the bitmap array structure provide valuable visibility into network traffic patterns and potential security concerns, allowing system administrators to track and analyze addresses of interest without requiring significant additional computational resources. This monitoring functionality integrates seamlessly with the core addressing methodology, creating a comprehensive solution for network management and security enforcement. The technical advancement represented by this extension to the disclosed method addresses a significant challenge in network security and management implementations-namely, the need to balance performance with flexibility in decision-making processes based on IP address evaluation. By delivering superior runtime performance while simultaneously expanding the range of possible outcomes beyond binary decisions, this innovation provides a more powerful and efficient approach to IP-based network management.

[0014] As discussed herein, the practical implications of the disclosed technological advancements are considerable, particularly in environments where network traffic volumes are substantial and processing efficiency directly impacts overall system performance and user experience. For example, by reducing the computational overhead associated with IP address-based decision making, the extended method enables security policies and routing decisions without incurring the performance penalties that would typically accompany such enhanced functionality. The unified approach to security policy enforcement, routing decisions, and monitoring capabilities represents a holistic solution to network management challenges, potentially simplifying implementation and maintenance compared to disparate systems handling these functions separately. This consolidation of functionality within a single methodological framework creates opportunities for more coherent and consistent network management strategies, potentially reducing configuration errors and improving overall system reliability.

Description

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The various features and advantages of the technology of the present disclosure will be apparent from the following description of particular embodiments of those technologies, as illustrated in the accompanying drawings. It should be noted that the drawings are not necessarily to scale; however, the emphasis instead is being placed on illustrating the principles of the technological concepts. The drawings depict only typical embodiments of the present disclosure and, therefore, are not to be considered limiting in scope

[0016] FIG. 1 schematic diagram illustrating an exemplary network operating environment in accordance with some embodiments of the instant disclosure.

[0017] FIG. 2A illustrates a flowchart of a method for utilizing an encoded firewall rule for processing a communication packet in accordance with some embodiments of the instant disclosure.

[0018] FIG. 2B illustrates a flowchart according to some embodiments of the instant disclosure;

[0019] FIG. 3 illustrates a flowchart of a method for encoding a firewall rule into a bitmap array in accordance with some embodiments of the instant disclosure.

[0020] FIG. 4 is a diagram illustrating encoding a firewall rule into a bitmap array in accordance with some embodiments of the instant disclosure.

[0021] FIGS. 5A and 5B illustrates a flowchart of a method for utilizing a bitmap of a firewall rule for processing a communication packet into or out of a secondary network in accordance with some embodiments of the instant disclosure.

[0022] FIG. 6 is a diagram illustrating a logic equation for utilizing various bitmap arrays to process a received communication packet in accordance with some embodiments of the instant disclosure.

[0023] FIG. 7 is a flowchart of a method for reducing bitmap arrays for a firewall device or devices to a single bitmap array in accordance with some embodiments of the instant disclosure.

[0024] FIG. 8 is a diagram illustrating an example of a computing system which may be used in implementing embodiments of the present disclosure.

DETAILED DESCRIPTION

[0025] Aspects of the present disclosure involve systems, methods, and the like, for encoding a firewall ruleset into one or more bit arrays, also referred to herein as a “bitmap”, for fast determination of processing of a received communication packet by a firewall device associated with a network. Through the bitmap encoding, all 4 billion Internet Protocol version 4 (IPv4) addresses may be mapped to a single bit within a 500 Megabyte (MB) data structure, with each bit in the data structure corresponding to a binary “drop” or “pass” rule for a received packet associated with an IP address. Through this bitmap, a number of computation operations needed to determine a processing rule for a received packet is significantly reduced compared to the traditional approach of using a hash or a longest prefix match technique. Rather, determining a processing rule for a received packet may include determining a bit value within one or more arrays of bits that indicates a processing rule for the received packet.

[0026] In one particular implementation, a firewall rule may be encoded into a 64-bit array of bit values (or bitmap) in which each bit of the array is a one value or a zero value that corresponds to a particular processing rule for a particular network address. For example, a firewall rule may indicate that packets from a particular source address be blocked from entering a secondary network. The firewall rule may be encoded into a bitmap array of bit values by asserting a particular bit within the array. In one particular implementation, the bitmap array may be indexed through a bitmap address that comprises a first portion of the source network address. For example, the first 26 bits of the source address may be used as a bitmap address (or indexing address at which a particular bitmap array is stored). The remaining portion of the source address may indicate a bit position within the bitmap array that is asserted in response to the firewall rule. For

example, the remaining 6 bits of the source address may provide a bit position within the bitmap array at which a bit may be asserted or de-asserted corresponding the firewall rule to encode the rule in the bitmap array.

[0027] In some instances, the firewall service may maintain different bitmap arrays that correspond to different firewall rules. For example, a blacklist bitmap array may store encoded firewall rules for addresses from which communications are blocked, a whitelist bitmap array may store encoded firewall rules for addresses from which communications are allowed to pass, and a threat bitmap array may store encoded firewall rules for addresses for which communications may be noted as a potential threat and further processing of the communication may be performed. Each of the different arrays may store the encoded firewall rules through the assertion or de-assertion of a bit within the respective arrays based on the type of rule. More or fewer bitmap arrays corresponding to other types of processing rules may also be maintained by the firewall device.

[0028] In other instances, the firewall service may condense the bitmap arrays corresponding to the different firewall rules into a single such array. Condensing of the bitmap arrays of the firewall rules may reduce the storage requirement for the firewall device by reducing the number of arrays stored at the device. In one implementation, the different bit map arrays discussed above may be generated, such as the whitelist array, the blacklist array, and/or the threat array, and precompiled into a single bitmap array prior to loading to the firewall device or other computing device. In one particular example, a union may be generated from a threat bitmap array and a blacklist bitmap array and a difference between the union set and a whitelist array may be determined. The resulting array may be stored as a new threat bitmap array that corresponds to the firewall rules for the firewall device. Further, because the firewall rules for the device have been reduced to a single array, a smaller portion of the memory of the device is needed to store the array in comparison to storing each of the whitelist, blacklist, and threat bitmap array discussed above.

[0029] Upon receipt of a communication either intended for a secondary network or received from the secondary network, the firewall device may utilize either the source or destination IP address of the received communication to access one or more of the bitmap arrays storing the encoded firewall rules to determine a bit value within the one or more arrays. The bit values from the bitmap arrays may indicate a processing rule for the received communication. In this manner, determining a firewall rule for processing the received communication may be one or more bitwise operations, executed with fewer operations and faster than a traditional firewall rule lookup, such as using hashing values or longest prefix matching with a Trie tree structure. As such, a firewall device may process received communications at a faster rate, significantly reducing the latency of communication packet transmission introduced by traditional firewall device.

[0030] Beginning at FIG. 1, an exemplary operating environment **100** for providing security services, such as a firewall security service, to one or more secondary networks **108** is illustrated. In general, the environment **100** provides for establishing communication sessions between network users or customers and for providing one or more network services to such network users over a backbone network, illustrated as IP network **102**. For example, customers may utilize the network **102** to communicate using communication devices, such as telephone devices and/or mobile communication devices. In another example, content from a content delivery network (CDN) may be provided to and/or from one or more customers of the network **102** through the operating environment **100** discussed herein. In still another example, the network environment **100** may facilitate communications between networks managed or administered by separate entities, such as communications between IP network **102** and secondary network **104**. The IP network **102** may also provide access services to one or more customers, such as access to the Internet **110**. With specific reference to FIG. 1, the environment **100** includes the IP network **102**, which may be provided by a wholesale network service provider. However, while the environment **100** of FIG. 1 shows a configuration using the IP network **102**, it should be appreciated that portions of the network may include non IP-based routing. For example, network **102** may include

devices utilizing time division multiplexing (TDM) or plain old telephone service (POTS) switching. In general, the network **102** of FIG. **1** may include any communication network devices known or hereafter developed.

[0031] The IP network **102** includes numerous components such as, but not limited to gateways, routers, route reflectors, and registrars, which enable communication and/or provides services across the IP network **102**, but are not shown or described in detail here because those skilled in the art will readily understand these components. Communications between the IP network **102** and other entities or networks, such as secondary network **104**, may also be managed through network environment **100**. In some instances, secondary network **104** may be a home network and can include personal communication devices such as, but not limited to, a personal computer or a telephone connected to a router/firewall. In other instances, secondary network **104** may include a larger-scale network, such as a virtual private network (VPN) or an enterprise network, which may connect to the network **102** in various geographic locations. In general, the secondary network **104** may include any type of communication device that receives a multimedia signal, such as an audio, video or web-based signal, and presents that signal for use by a user of the communication device. The communication and networking components of the secondary network **104** enable a user at the secondary network to communicate via the IP network **102** to other communication devices, such as another customer network, or receive services from the IP network, such as security services and/or access to the Internet **110**. In some embodiments, multiple communication devices in diverse locations that are owned or operated by a particular entity or customer may be connected through the IP network **102**.

[0032] Networks, such as the Internet **110** and secondary network **104**, may connect to IP network **102** through one or more interface devices. Interface devices may include, but are not limited to, provider edge devices **112** and/or Internet gateway device **114**. For ease of instruction, only two external networks **104**, **110** are shown communicating with the IP network **102**; however, numerous such networks, and other devices, may be connected with the network, which is equipped to handle enormous numbers of simultaneous calls and/or other IP-based communications. As discussed, the IP network **102** may provide one or more telecommunication or other network services to the secondary network **104**. For example, an operator of secondary network **104** may purchase, and the IP network **102** may provide, one or more security services for protection of the communications provided to and transmitted from the secondary network. Such security services may be provided via the provider edge **112** of the network **102** or any other device of the network, as explained in more detail below.

[0033] In one particular example, the IP network **102** may provide a firewall service to the secondary network **104** from one or more firewall devices **116**. In general, the firewall service provides gatekeeping functions for communications into or out of the secondary network **104**. In one example, the firewall service may be included in the provider edge device **112** of the IP network **102** that receives communications associated with the secondary network **104**. In another example, the firewall service may be implemented by a firewall device **116** or firewall server of the IP network **102**. The firewall service may block, re-route, deny, flag, etc. communications entering or leaving the secondary network **104** based on a set of firewall rules, also referred to as a firewall ruleset. Management and configuration of the firewall rules of the firewall service is discussed in more detail below, including encoding the firewall ruleset into one or more bitmap arrays in which each bit of the arrays corresponds to a rule of the ruleset.

[0034] The firewall device **116** may, in some instances, be a server device of the IP network **102** configured to provide the firewall services to communications associated with the secondary network **104**. In other implementations, the firewall device **116** may be a virtual machine hosted on a plurality of computing devices of the network **102**, such as a plurality of application servers in communication with or within the network. In some embodiments, the firewall device **116** may include, three network interfaces comprising a management interface used to administer the device

including configuration of the operating system (OS) of the device, an encoding and processing application running on the OS, and updating a threat feed and network data from upstream systems. In some embodiments, such interfaces can be, but are not limited to, separate physical interfaces, each with a different physical network link, virtual interfaces (e.g., virtual local area network (VLAN), for example), and the like, or some combination thereof. The firewall device **116** may further include a private side interface (an interface facing the secondary network **104**) that connects to a device of the secondary network, such as a router or other network device. An Internet side interface (an interface facing into IP network **102**) may also be included that connects to a device associated with the Internet **110**, i.e. a router that can be the gateway, default route, to the Internet as provided to the secondary network **104** without the firewall device being present. Further, although illustrated as connected to a single secondary network **104**, the firewall device **116** may be connected to or otherwise associated with any number of devices, networks, etc. connected to the network **102** to firewall services to the devices and networks.

[0035] In one implementation, a security management system **118** may communicate with the IP network **102** and/or the secondary network **104** to manage the firewall device **116**. Although shown as instantiated within the IP network **102**, the security management system **118** may be separate from the IP network. In general, the security management system provides configurations and updates to the devices and services of the firewall device **116** in response to threat data or information to update the firewall ruleset encoded into the one or more bitmap arrays.

[0036] In general, the firewall device **116** may be deployed between the secondary network **104** and the gateway **114** to the Internet **110**. In one implementation, the firewall device **116** is deployed using network layer **3** (IP) routing such that communications to and from the secondary network **104** are routed to the firewall device **116**, regardless of how the device is physically connected within the network **102**. In another implementation, the firewall device **116** may be physical connected between the secondary network **104** and the Internet **110** to process all communications between the secondary network and the Internet gateway **114**. Those of ordinary skill in the art will appreciate the various ways in which the firewall device **116** may receive communications associated with the secondary network **104** for application of the firewall ruleset associated with the secondary network.

[0037] FIG. **2A** illustrates a flowchart of a method **200** for utilizing an encoded firewall rule for processing a communication packet in accordance with one embodiment. The operations of the method **200** may be executed by the security management system **118** and/or the firewall device **116** of network **102**. Through the method **200**, a firewall ruleset may be encoded into a data structure, referred to as a bitmap, that improves the processing of communications into or out of the secondary network **104** to which the firewall device **116** is associated.

[0038] Beginning at operation **202**, one or more firewall policy rules may be received for inclusion in a firewall ruleset for enforcement by the firewall device **116**. In general, the firewall rule includes a processing rule for a received communication, such as accept the communication packet, deny the communication packet, reroute the communication packet, mark the communication packet as a potential threat. As explained above, a firewall device **116** typically inspects or analyzes each packet that travels through it to decide if it should allow the packet to pass through the firewall based on a sequence of rules (or ruleset) pertaining to the values of the one or more fields in the packet. For example, a packet may include a source IP address and destination IP address. A firewall rule may utilize either the source IP address and/or the destination IP address values to determine whether the packet is allowed to pass through the firewall or is denied. As such, a decision portion of a firewall rule determines what happens if the value obtained from the communication packet matches to address value of the rule. The rule then typically employs an accept or deny action on the packet, with the possibility of additional actions, such as an instruction to log the action, mark the packet, re-route the packet, and the like.

[0039] As discussed above, a firewall policy is generally made up of an ordered list of these rules

such that as a packet is processed by the firewall device **116**, the firewall attempts to match some aspect of the packet to the rule one rule at a time, from beginning of the rule list to the end. Matching the packet means that the firewall evaluates a packet based on the fields in the rule tuple to determine if the fields match the values identified in each individual rule of the ruleset. A rule does not necessarily need to contain a value for all possible fields and can sometimes contain an “any” variable in a field to indicate that the rule is a “do not care” condition for that variable. In general, these rules are processed in order until the firewall device finds a match and takes the appropriate action identified by the decision portion of the rule.

[0040] The received firewall rule may therefore include a value, such as a source IP address or destination IP address, known as the tuple of the rule and a processing action for communications that match the tuple value. Although many processing actions may be included in the received firewall policy rule, the present disclosure discusses the processing actions of accept, deny, and mark as a potential threat. However, the operations and systems described herein may be applied to any processing rule included in any type of firewall rule supported by firewall device **116**. For example, the firewall device **116** may be configured to route a received communication packet to a scrubbing environment of the network **102** based on a firewall policy rule. Other processing actions may also be undertaken by the firewall device **116**.

[0041] In one particular implementation, the firewall policy rule may be received from a third-party source of firewall rule information. The third-party source may analyze Internet traffic data and identify possible sources of malicious or dangerous traffic, such as a source of frequent denial of service (DOS) attacks on a network, a common source of packets that contain a known virus, a common source of known spamming communications, and the like. In another implementation, network **102** may include a security threat analysis system to identify sources of potentially malicious data packets, which may or may not be the security management system **118**. The third-party source or network threat analyzer may generate a firewall rule for identified sources of potentially malicious communication packets and provide the generated firewall rule to the firewall device **116**. In another implementation, the security management system **118** may manage the distribution of firewall rules to one or more firewall devices of the network **102** such that the firewall rule to be encoded may be received from the security management system.

[0042] Regardless of the source of the firewall rule, the firewall device **116** or security management system **118** may, at operation **204**, encode the policy rule into one or more bitmap arrays. FIG. 3 illustrates a flowchart of one method **300** for encoding a firewall rule into a bitmap array in accordance with one embodiment. In particular, an address associated with the firewall rule may be obtained at operation **302**. As mentioned above, the firewall rule may include a tuple that, in some instances, may be a network address. The network address may be a source address. For example, a particular network address may be identified as a source of potentially malicious communication packets such that a firewall rule may identify the source address and a block action. In another example, the firewall rule may be configured to allow all traffic originating from the secondary network **104** to pass through the firewall device **116**. As such, the firewall rule may include the source address as any address of devices of the secondary network **104** and an allow action. In a similar manner, the firewall rule may identify a destination address for one or more devices within the secondary network **104** for which all traffic may be blocked. In another example, a particular destination address of the Internet **110** may be identified as a device associated with a scam such that the firewall rule may identify the destination address and a block action. Any combination of source/destination addresses and processing action may be included in the firewall rule.

[0043] At operation **304**, a bitmap array for the firewall policy rule may be determined. As described above, the firewall device **116** may maintain different bitmap arrays that correspond to different firewall processing actions. For example, a blacklist bitmap array may store encoded firewall rules for addresses from which communications are blocked, a whitelist bitmap array may store encoded firewall rules for addresses from which communications are allowed to pass, and a

threat bitmap array may store encoded firewall rules for addresses for which communications may be noted as a potential threat and further processing of the communication may be performed. Thus, depending on the type of firewall rule being encoded (block, allow, mark as a threat, etc.), a bitmap array corresponding to that processing action may be determined. For example, for a received firewall rule that includes a block action, a blacklist bitmap array may be determined as the type of bitmap array for the received firewall rule. Similarly, a whitelist bitmap array may be determined for a received firewall rule that includes an allow processing rule. Other types of bitmap arrays may be maintained by the firewall device **116** for other types of processing actions included in one or more firewall rules.

[0044] According to some embodiments, the processing in operation **304** can involve functions for encoding that is to be used to set addresses (e.g., all or at least a portion) within a network address to the bitmap index (obtained in operation **208**, discussed infra).

[0045] By way of a non-limiting example, according to some embodiments, the operation **304** can be used to store arbitrary data in a bitmap corresponding to a specific IPv4 network address.

[0046] For example, given a data value, `bitfield_value`, to be associated with a particular IPv4 Network Address, the size of this data can be specified as `bitfield_size` from which the `number_of_bitfields`, `bitfield_index_bits`, `ipv4_address_to_data_map_index_bits`, and `bitfield_mask` can be calculated based on the machine architecture word size. A sequence of IPv4 Addresses can be computed from the IPv4 Network Address using standard CIDR prefix processing routines available in common libraries for most programming languages. For each address contained within the given IPv4 Network Address as the `target_ipv4_address`, the functioning can involve first obtaining the `ipv4_address_to_data_map_index` and `ipv4_address_to_data_map_value`; then calculating the `bitfield_index` and `bitfield_index_shift` are obtained from the target address. Next shifting the `bitfield_mask` and `bitfield_value` to the proper bitfield location in the `ipv4_address_to_data_map_value` by left shifting both by `bitfield_index_shift`. Next, AND the inverse of the left-shifted `bitfield_mask` with the `ipv4_address_to_data_map_value` (as discussed infra), determining a new value with the bitfield set to all zeros. In some embodiments, such new value with the left-shifted `bitfield_value` to set the bitfield to the given data value can involve obtaining the replacement `ipv4_address_to_data_map_value`, which is stored in the `ipv4_address_to_data_map` at the `ipv4_address_to_data_map_index`.

[0047] At operation **306**, a bitmap address may be determined from the network address included in the received firewall rule. In one particular implementation, the bitmap address may correspond to the first 26 bits of the address included in the received firewall rule. Generally, an IP address comprises a 32-bit string of binary values. FIG. 4 is a diagram illustrating an example of an IP address included in a firewall rule that may be used to encode a firewall rule into a bitmap array in accordance with one embodiment. As mentioned above, the address value **402** in the firewall rule may be a source address or a destination address. Regardless, a portion of the address **402** included in the firewall rule may be obtained and used to determine a bitmap address at which a bitmap array is stored. In the illustrated example, the first 26 bits of the address **402** identified in the firewall rule is used as the bitmap address **404** for the particular bitmap array. For example, the firewall rule may include a tuple value of a network address **402** of: [0048] [00100101].[11111111].[01010101].[11111111].

[0049] This address **402** may correspond to a source address or a destination address, as identified in the firewall rule. From a portion of this address **402**, a bitmap address **404** may be identified. In one example, the first 26 bits of the network address **402** in the firewall rule may be identified as the bitmap address **404**, comprising the bit string [0050] [011111111101010101111111].

[0051] This bitmap address value **404** may correspond to an address location within the bitmap data structure of the firewall device **116** at which the determined bitmap array to be updated with the received firewall rule may be stored. For example, at address location

[0111111110101011111111] of the firewall device **116** data structure, one or more bitmap arrays may be stored corresponding to different firewall policy actions, such as a blacklist array, a whitelist array, a threatlist array, and the like. In this manner, a particular bitmap array within the firewall device **116** data structure may be identified through the address **402** included in the received firewall rule. The identified bitmap array may then be updated to encode the received firewall rule into the determined bitmap array, as explained below. Further, although discussed herein as setting the first 26 bits of the IP address of the firewall rule as the bitmap address, the bitmap may include any number of bits from the network address, either contiguous or noncontiguous, as the bitmap address used to access a bitmap array from the data structure.

[0052] At operation **308**, an address bit within the identified bitmap array may be identified from a second portion of the address identified in the received firewall rule. In one particular implementation, the address bit value may be the Nth bit within the identified bitmap array, where N is a bit position indicated by the last 6 digits of the network address in the firewall rule.

Returning to the illustration of FIG. **4**, a second portion of the network address **402** comprising a number of bits may be set as the address bit **406** for the identified bitmap array stored at and obtained from the data structure at the bitmap address **404**. In the illustrated example, the bit address **406** corresponds to the last six bits of the network address **402**, or the remaining portion of the network address after the bitmap address **404** is determined. In general, however, the bit address may be any number of bits from the network address, either contiguous or noncontiguous.

[0053] The bit address **406** generally indicates a bit position within the identified bit array. For example and as described above, a type of bit array may be identified corresponding to a processing action of the received firewall rule, such as a blacklist array, whitelist array, etc. A particular bitmap array may be accessed in the data structure at the bitmap address **404** (determined from the first portion of the network address included in the firewall rule). To determine which bit in the identified bitmap array to assert, the bit address **406** may be obtained from a second portion of the network address **402**. In the example shown, the bit address **406** comprises six bits such that the corresponding bitmap array may be 64 bits long (as the six bits of the bit address can provide up to 64 different bit locations within the array). The second portion of the network address **402** illustrated in FIG. **4** is the bit string [001001], or a value of 9. Thus, in this example, N=9 such that the bit address or position within the identified bitmap array to be asserted to encode the received firewall rule is bit position 9. The bit address **406** may identify any bit position within the bitmap array, such as any position 0-63 for 64-bit array. Bit addresses comprising a smaller portion of the network address **402** may be utilized for smaller arrays and bit addresses comprising a larger portion of the network address may be utilized for longer arrays.

[0054] At operation **310**, the received firewall rule may be encoded into the identified bitmap array by asserting a bit within the array corresponding to the bit address **406** value obtained from the network address **402**. For example, a bitmap array **408** may be identified from a firewall rule (such as a blacklist array) and accessed at bitmap address **404** obtained from the first portion of the network address **402**. To encode the firewall rule, the bit address **406** may be obtained from the second portion of the network address and a value of the second portion may be determined. In the example shown, the second portion equals “9”, such that the bit address or position within the bitmap array **408** to be asserted is the ninth bit position. Asserting the Nth bit within the bitmap array **408** encodes the firewall rule into the data structure. In an alternate embodiment, the bitmap array **408** may comprise a series of one values and encoding the firewall rule includes de-asserting the bit at the Nth bit position to a zero value. Regardless, the bitmap array **408**, after updated to include the encoded firewall rule, may be stored in the data structure. Additional firewall rules may be encoded in a similar manner such that the data structure stores an encoding of each firewall rule of the ruleset associated with the firewall device.

[0055] The firewall device **116** may utilize the encoded firewall ruleset to process communications associated with the secondary network **104**. For example and returning to the method **200** of FIG.

2A, a communication packet may be received at the firewall device **116** at operation **206**. The communication may be received from a device of the secondary network **104** for transmission to a device of the Internet **110** or may be received from a device of the Internet intended for a device of the secondary network. Based on a source or destination address included in the received communication, the firewall device **116** may obtain an encoded firewall rule for processing the received packet at operation **208**. In some embodiments, more particularly, the firewall device **116** may utilize a portion of the bits of the network address included in the received communication to identify a storage location within the data structure at which one or more bitmap arrays may be stored. In one example, the firewall device **116** may extract or otherwise determine the first 26 bits of the network address to determine the bitmap address value. One or more bitmap arrays may be accessed from the data structure using the bitmap address. Further, a processing action may be determined from a bit within the one or more bitmap arrays, such as a blocking action from a blacklist array, an allow action from a whitelist array, a marking action from a threatlist array, and the like. The bit within a particular bitmap array that determines a processing action by the firewall device **116** may be determined from a second portion of the network address in the received communication. In particular, the second portion may provide a bit position value within a bitmap array. An asserted bit at the determined bit position within the bitmap array may instruct the firewall device **116** on a processing action for the received communication.

[0056] According to some embodiments, the processing in operation **208** can involve sub-operations **250-258**, as discussed in FIG. 2B. Accordingly, as discussed herein, operation **208** can involve operations for the network sub-portion of the target address being used to lookup the bitmap that contains the rules that apply to the network that hosts the target address.

[0057] According to some embodiments, the disclosed operations/sub-operations function to determine a value associated with a given IPv4 address. This value can be used, for example, as the index to the memory location of a bitmap containing the rules that apply to the network that includes the given IPv4 address. For example, such network belongs to a particular customer who has specified the rules applicable to addresses on their network. It should be understood that such example is non-limiting; and for purposes of the example, and the discussion herein, the IPv4 address to do this lookup will be referred to as the `target_ipv4_address`.

[0058] In some embodiments, the operations can use the new bitmap structure called the `ipv4_address_to_data_map` that stores the associated values. On a typical modern computer architecture, this can be an array of unsigned 64-bit integers. The array itself can be indexed based on the `target_ipv4_address`, and how much of the address that is used can be based on the size of the data to store. The upper bound of the array can be $2^{\lceil \frac{\text{target_ipv4_address}}{64} \rceil}$, with a lower bound of $2^{\lceil \frac{\text{target_ipv4_address}}{64} \rceil - 26}$. The `ipv4_address_to_data_map` can be initialized as a data structure completely filled with 0s. This results in a necessary assumption for any use case, that the data value 0 for any IPv4 address can be treated as a null or unassigned value. In short, the data structure can be an array of 64-bit words divided into a number of equal sized bit fields that each contain a data value for a particular ipv4 address. A bit field (bitfield) can be a subset of bits within the 64-bit word that is evenly divisible into the word size, which is a power of two selected from the integer factors of the word size. Possible field sizes for a 64-bit word are 1, 2, 4, 8, 16, 32 and 64. The number of fields stored in a word can be equal to the word divided by the field size. In the case of a 64-bit word, the number of fields for each possible field size are 64, 32, 16, 4, 2, and 1, respectively.

[0059] In some embodiments, a first step is to determine the size in bits of the data desired to be stored. For example, to store an integer value from 0 to 40 requires a field size of 8 bits with a maximum value of 64, being the next largest available bit field size that can store integers up to 40. This value is called the `bitfield_size`. The number of bit fields within the word is called `number_of_bitfields`, which is 8 for this example (8 bit fields of 8 bits each comes to a total of 64-bits as the word size.)

[0060] The number of bit fields determines how many bits need to be reserved from target_ipv4_address to index the data. Reserve the lowest $\log_2(\text{bitfield_size})$ bits of the target_ipv4_address. Continuing with the example, with 8 fields, 3 bits are required from the target_ipv4_address. These bits are referred to as bitfield_index_bits.

[0061] This determines the number of elements in the array allocated to store the bitmap. To store values for all IPv4 address requires $2^{\lceil 32/\text{number_of_bitfields} \rceil}$ elements, this being the length of the array referred to as the ipv4_address_to_data_map_length. In the example, the ipv4_address_to_data_map has 536870912 elements. Each element is 64-bits, which is 8 bytes. The total size of this array is $\text{ipv4_address_to_data_map} * \text{word_size_in_bytes}$, equalling $536870912 * 8 = 4294967296$ bytes. The bits required to index this array are obtained from the $\log_2(2^{\lceil 32/\text{number_of_bitfields} \rceil})$ high order bits of the target address. This value is called ipv4_address_to_data_map_index_bits.

[0062] The proceeding values can all be precalculated and/or hardcoded for a particular application for speed. The array can be allocated from shared memory and initialized with all elements set to zero.

[0063] Accordingly, as depicted in FIG. 2B, operation **250** the target_ipv4_address can be shifted right by the bitfield_index_bits. In the example, this is $\text{target_ipv4_address} \gg 3$. This establishes a map index, known as the ipv4_address_to_data_map_index, which specifies the memory location of the word containing bit field data array. The data at this index is called the ipv4_address_to_data_map_value, which is a word as specified by the host machine architecture.

[0064] In operation **252**, the index to the bit field within the retrieved data map value is determined. This is obtained by masking the lowest order bitfield_index_bits bits from the address—e.g., $\text{target_ipv4_address} \& 0x07$ for the given example. This value is called the bitfield_index.

[0065] In operation **254**, the bit field value is determined. According to some embodiments, to determine the bit field value from the ipv4_address_to_data_map_value the word is shifted and masked. The shift is calculated by multiplying the bitfield_index by the number of fields number_of_bitfields. This is $\text{bitfield_index} * 8$ in the example. This shift value is called bitfield_index_shift. In operation **256**, the mask, known as the bitfield_mask is obtained by setting all bits to 1 in a bit field of bitfield_size, which is 0xFF in the example case. And, in operation **258**, the bitfield_mask is applied to the result of shifting the ipv4_address_to_data_map_value by bitfield_index_shift with a logical AND operator. The complete operation in pseudocode is: $(\text{ipv4_address_to_data_map_value} \gg \text{bitfield_index_shift}) \& \text{bitfield_mask}$. This produces the final stored map value (stored_map_value) associated with the given target address, as in operation **258**.

[0066] By way of a non-limiting example: [0067] target_ipv4_address: 1. 1. 1. 1 [0068] bit representation: 0b0000 0001 0000 0001 0000 0001 0000 0001 [0069] #machine_architecture is 64-bit and so: [0070] word_size=64 [0071] bitfield_size=8 [0072]

number_of_bitfields=word_size/bitfield_size=8 [0073] bitfield_index_bits= $\log_2(\text{bitfield_size})=3$ [0074] bitfield_index_mask=0b0000 0111=0x7 [0075] ipv4_address_to_data_map_index_bits=32-bitfield_index_bits=29 [0076] ipv4_address_to_data_map: Array[u64int] [0077]

ipv4_address_to_data_map_length= $(2^{\lceil 32/\text{number_of_bitfields} \rceil} * \text{word_size_in_bytes}) - 1$ [0078]

ipv4_address_to_data_map_length=536870911 [0079] # For the sake of this example, assume the ipv4_address_to_data_map has been populated with some values [0080]

ipv4_address_to_data_map_index=target_ipv4_address>>bitfield_index_bits [0081]

ipv4_address_to_data_map_index=0b0000 0001 0000 0001 0000 0001 0000 0001>>3 [0082]

ipv4_address_to_data_map_index=0b0000 0000 0010 0000 0010 0000 0010 0000 [0083] decimal

ipv4_address_to_data_map_index=2105376 [0084] ipv4_address_to_data_map_value= [0085]

ipv4_address_to_data_map [ipv4_address_to_data_map_index] [0086]

=ipv4_address_to_data_map [2105376] [0087] =0x00 00 00 00 00 00 01 00 [0088]

bitfield_index=target_ipv4_address & bitfield_index_mask [0089] =0b0000 0001 0000 0001 0000 0001 0000 0001 & 0b0000 0111 [0090] =1 [0091]

bitfield_index_shift=number_of_bitfields*bitfield_index [0092] =8*1 [0093] =8 [0094]
 bitfield_mask=0xFF [0095] stored_map_value=
 (ipv4_address_to_data_map_value>>bitfield_index_shift) & bitfield_mask= [0096] (0x00 00 00 00
 00 00 01 00>>8) & 0xFF [0097] =(0b0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
 0000 0000 0001 0000 0000>>8) & 0b1111 1111 [0098] =(0b0000 0000 0000 0000 0000 0000 0000 0000
 0000 0000 0000 0000 0000 0000 0000 0001) & 0b1111 1111 [0099] =0b0000 0001 [0100] =1
 [0101] Turning back to FIG. 2A, based on the encoded rules stored in the one or more bitmap
 arrays, the firewall device **116** may process the received communication at operation **210** through a
 processing action, such as a deny action, allow action, re-route action, etc. In this manner, each rule
 of a firewall ruleset may be encoded as a single bit value within a data structure such that
 determining the processing action for a received communication may occur quickly by the firewall
 device **116**, reducing a latency at the device for processing the communication.
 [0102] FIGS. 5A and 5B illustrates a flowchart of a more detailed method **500** for utilizing a
 bitmap of a firewall rule for processing a communication packet into or out of a secondary network
104 in accordance with one embodiment. The operations of the method **500** may be performed by
 the firewall device **116** (or any other component of the network **102**) to process a received
 communication based on a ruleset encoded into the bitmap or data structure through the methods
 described above. The operations may reduce determining an applicable firewall rule to a received
 communication packet to bitwise computations, increasing the processing speed of the received
 communication by the firewall device **116**.
 [0103] At operation **502**, the firewall device **116** may receive a communication packet associated
 with a secondary network **104**. In the implementation in which the firewall device **116** provides
 firewall services to multiple secondary networks or devices, the firewall device may determine a
 secondary network or device for which the communication is associated. For example, each
 communication may include a header that comprises a source network address and a destination
 network address. These addresses may be IP networks addresses, although other network address
 standards are also contemplated. The firewall device **116** may include a database of addresses
 associated with any secondary network or device receiving a firewall service from the firewall
 device. Upon receiving a communication, the firewall device **116** may compare the source address
 and/or the destination address to the databases of addresses to determine which secondary network
 or device is associated with the received communication (either as a sender of the packet or a
 receiver of the packet). The firewall device **116** may further determine which interface the packet is
 received. For example, the firewall device **116** may include an Internet/network communication
 port or interface that faces the IP network **102** and the Internet **110** to receive communications from
 and send communications to the Internet. The firewall device **116** may include another port or
 interface that faces the secondary network **104** to receive communications from and send
 communications to the secondary network. Thus, the firewall device **116** may determine through
 which interface the communication is received.
 [0104] If the communication packet is received on the Internet interface as determined at operation
504, the firewall device **116** may set the network address as the source address included in the
 header of the packet at operation **506**. Because the communication packet is received on the
 Internet interface, the source address included in the header indicates the Internet device that
 initiated transmission of the communication to the secondary network **104**. If the communication
 packet is not received on the Internet interface, the firewall device **116** may set the network address
 as the destination address included in the header of the packet at operation **508**. Communication
 packets received on the secondary network **104** interface are intended to be transmitted to a device
 of the network **102** or the Internet **110** such that the destination address in the header of the packet
 may be set as the network address for analysis by the firewall device **116**. In the example illustrated
 in FIG. 4, the set network address is network address **402** used to determine a corresponding
 firewall rule for the received communication.

[0105] At operation **510**, the firewall device **116** may set the bitmap address as a first portion of the network address determined above. As described above, the bitmap address may be the first 26 bits of the network address, although any portion of the network address may be set as the bitmap address. For a firewall device **116** that services multiple secondary networks or devices, the firewall device may determine a corresponding data structure of encoded firewall rules to apply to a received communication. For example, the firewall device **116** may store multiple data structures of encoded firewall rules, indexed by an identifier of a secondary network **104** or device for which the firewall ruleset applies. Thus, the firewall device **116** may identify a particular data structure of an encoded firewall ruleset for application to a given received communication.

[0106] In a similar manner, the firewall device **116** may, at operation **512**, determine an address bit value as a second portion of the network address. In one particular example, the address bit value may correspond to the last six bits of the network address, although any sized portion of the network address may be used as the bit value. The bit value provides a bit position within one or more bit arrays to which a firewall rule may be encoded, as described above.

[0107] The firewall device **116** may access or obtain one or more stored bitmap arrays from a data structure using the bitmap address determined above. For example, the first 26 bits of the network address may provide a 26-bit address within the data structure at which one or more bitmap arrays are stored. The bitmap arrays may correspond to different communication packet processing actions, such as a blacklist array, a whitelist array, and a threatlist array. Thus, at operation **514**, the firewall device **116** may access a threatlist array at the bitmap address. The threatlist array may, in one implementation, be a 64-bit array that stores encoded firewall rules for identifying a received communication as a potential threat. To determine if the particular received communication should be processed as a potential threat, the firewall device **116** may determine a bit position within the threatlist array corresponding to the address bit value determined above. In another implementation, the threatlist array may encode firewall rules provided by a threat feed provider as a list of known threats for blocking. In a similar manner, at operation **516**, the firewall device **116** may also access a blacklist array at the bitmap address. The blacklist array may, in one implementation, be a 64-bit array that stores encoded firewall rules for identifying a received communication that is to be blocked or denied. In some implementations, the blacklist may be provided by the customer of the system to specify additional addresses or addresses to be blocked. A whitelist array may, at operation **518**, be accessed at the bitmap address. The whitelist array may, in one implementation, be a 64-bit array that stores encoded firewall rules for identifying a received communication that is to be allowed to be transmitted to the destination address. Additional or fewer bitmap arrays may be stored or otherwise accessed through the bitmap address for other communication packet processing actions.

[0108] In FIG. 5B, the firewall device **116** may determine, at operation **520**, if the Nth bit in the blacklist array or the Nth bit in the threatlist array is asserted. An asserted bit in either of the threatlist array or the blacklist array may indicate a “deny” processing action for the received communication. However, if the Nth bit in either the threatlist or the blacklist is un-asserted, the firewall device **116** may redirect the communication packet to an egress port at operation **526**. Sending the packet to the egress port of the firewall device **116** prepares packet to be transmitted to the destination address or network as indicated by the destination address. Further, even if the Nth bit in the blacklist array or the Nth bit in the threatlist array is asserted, the firewall device **116** may include a rule to automatically let all communications from a particular address to pass, as encoded into the whitelist array. Thus, the firewall device **116** may determine, at operation **522**, if the Nth bit in whitelist bitmap array is un-asserted. If the Nth bit of the whitelist bitmap array is un-asserted, the firewall device **116** may drop or deny the communication packet at operation **524** as a communication packet that is potentially malicious. If instead the Nth bit of the whitelist bitmap array is asserted, the firewall device **116** may redirect the communication packet to the egress port at operation **526** for transmission to the destination device indicated by the destination address in

the packet header.

[0109] FIG. 6 is a diagram illustrating a logic equation for utilizing various bitmap arrays to process a received communication packet in accordance with one embodiment. The logic equation **600** is a bitwise operation of some of the operations of the method **500** discussed above with reference to FIGS. 5A and 5B. The firewall device **116** may execute the bitwise logic equation **600** to determine a processing action for a received communication packet, such as part of operations **520-526** discussed above with reference to method **500**. The logic equation **600** of FIG. 6 may include a bitwise OR operation of a blacklist array **602** with a threatlist array **604**. This operation may generate an array in which each bit of the array is asserted if a bit in the same position in either the blacklist array **602** or the threatlist array **604** is asserted. The resultant array may then be bitwise AND operation with an inverted whitelist array **606** to generate a combined array for determining the processing action of a received communication. The AND operation may generate an array in which each bit of the array is asserted only if a bit in the same position in both the resultant array from the OR operation and the inverted whitelist array is asserted. The firewall device **116** may use the array resulting from the bitwise operations **600** to determine a processing operation for a received communication packet. For example, the firewall device **116** may determine if the Nth bit in the array generated from the bitwise operations **600** of FIG. 6 is asserted. If the Nth bit is asserted, the firewall device **116** may block or deny the communication packet. In this manner, the bitmap arrays may be combined through one or more bitwise operations such that the firewall device **116** may determine the Nth bit of a single array comprising the combination of the various bitmap arrays at the bitmap address of the data structure. Through the combination of the bitmap arrays using the bitwise operations, a time for determining a processing action for a given communication packet may be further reduced to a single bit lookup in the combined array of the various bitmap arrays of the data structure.

[0110] Although discussed above with reference to a deny or accept processing action, it should be appreciated that other processing actions may be encoded into a bitmap array. For example, the threatlist array may correspond to a re-direct processing action for which, if the Nth bit of the threatlist array is asserted, the firewall device **116** may re-direct a received communication packet to a pre-designated device or destination. For example, an asserted bit within the threatlist array may correspond to re-directing a received communication to a scrubbing device or environment of the network **102**. In another example, the threatlist array may correspond to setting a bit or bits within the header of the communication that indicates the communication is suspect. The secondary network **104** may process the received communication differently based on the set bit or bits within the header. In this manner, any number of bitmap arrays may be stored in the data structure of the firewall device **116** with encoded firewall rules each corresponding to a type of processing of a received communication packet.

[0111] In some instances and to reduce the memory usage for maintaining the bitmap arrays for the firewall device, the threatlist, blacklist, and whitelist bitmap arrays associated with a customer may be compiled into a single bitmap array. In one implementation, compiling the bitmap arrays may reduce the memory consumption at the firewall device from 1.5 gigabytes (GB) to 0.5 GB or less. FIG. 7 is a flowchart of a method **700** for reducing bitmap arrays for a firewall device or devices to a single bitmap array. The operations of the method **700** may be performed prior to or during a loading of the bitmap array of the firewall rules to the firewall device. Through the operations, the bitmap array of the firewall rules may be reduced to a 64-bit integer. In one instance, the firewall device may execute the operations of method **700** to generate the single bitmap array corresponding to the firewall rules. In other instances, another computing device may perform the operations to generate the bitmap array and provide the condensed array to the firewall device. Regardless, the firewall device may store the condensed bitmap array in less memory than devices that utilize the whitelist array, blacklist array, and threat array as described above.

[0112] Beginning in operation **702**, a blacklist bitmap array, a whitelist bitmap array, and a

threatlist bitmap array associated with a network may be determined, as described above. Each of the bitmap arrays may, in one implementation, be a 64-bit array that stores encoded firewall rules for processing a received communication. The blacklist array may encode “deny” firewall rules for a received communication, the whitelist array may encode “allow” firewall rules for a received communication, and the threatlist array may encode rules that identify a potential threat from a received communication. Each of the bitmap arrays may be similar to that discussed above with reference to FIG. 4 and may be generated as described herein. However, as each bitmap array comprises a 64-bit array, the memory requirement for storing the bitmap arrays may be 1.5 GB or more. In addition, the bitmap arrays determined may be associated with a particular network or with a particular IP prefix associated with a network or customer to a network. Thus, each of the bitmap arrays may correspond to a particular IP prefix for controlling the processing of communications intended for that prefix or network. In this manner, the condensed bitmap array may be generated for a particular network or firewall device such that each network or customer may have a unique bitmap array for the corresponding firewall policy. In other instances, a condensed bitmap array may correspond to more than one network or prefix, such as instances in which a firewall policy is applied to several networks through a firewall device or network of devices.

[0113] To reduce the memory requirement for the bitmap arrays, the three arrays may be combined or condensed into a single bitmap array that represents the information contained in each of the arrays. For example, in operation **704**, an array of a union operation of the blacklist bitmap array and the threatlist bitmap array may be generated or otherwise determined. In general, the union operation combines elements of two sets. The union of the blacklist array and the threatlist array generates a single array in which each bit that is asserted in either the blacklist array and the threatlist array is asserted in the union array. As such, the union array includes an asserted bit at each location corresponding to a mitigation action, such as a block or a re-direct action for the communication. At operation **706**, a difference array between the union array and the whitelist bitmap array may be generated or otherwise determined. Through the difference operation, the asserted bits within the whitelist bitmap array may become de-asserted within the union array. The difference operation thus removes the bits corresponding to an “allow” rule from the union set. At operation **708**, the difference array generated in operation **706** may be set as a new threatlist bitmap array for the network or device. This new threatlist bitmap array represents the firewall policy for the network or device.

[0114] The firewall device may utilize this new threatlist bitmap array to process communications similar to that described above. However, rather than accessing each of the blacklist array, whitelist array, and the threatlist array as described, the device may access the new threatlist bitmap array to determine a bit position within the array and process the communication accordingly. In particular, once the corresponding bit location within the array is determined as described above, the firewall device may determine if the bit at the location is asserted. If asserted, the firewall device may take some processing action on the communication, such as a re-direct or block action. If the bit at the determined location is non-asserted, the firewall device may allow the communication to pass through the firewall device. In this manner, the firewall rules may be encoded within the new threatlist array and used to process communications through the firewall device. Further, as the bitmap arrays for the device are condensed to a single array, even less system memory is needed to store the firewall policy.

[0115] FIG. 8 is a block diagram illustrating an example of a computing device or computer system **800** which may be used in implementing the embodiments of the components of the network disclosed above. For example, the computing system **800** of FIG. 8 may be the security management system **118** or the firewall device **116** discussed above. The computer system (system) includes one or more processors **802-806**. Processors **802-806** may include one or more internal levels of cache (not shown) and a bus controller or bus interface unit to direct interaction with the

processor bus **812**. Processor bus **812**, also known as the host bus or the front side bus, may be used to couple the processors **802-806** with the system interface **814**. System interface **814** may be connected to the processor bus **812** to interface other components of the system **800** with the processor bus **812**. For example, system interface **814** may include a memory controller **818** for interfacing a main memory **816** with the processor bus **812**. The main memory **816** typically includes one or more memory cards and a control circuit (not shown). System interface **814** may also include an input/output (I/O) interface **820** to interface one or more I/O bridges or I/O devices with the processor bus **812**. One or more I/O controllers and/or I/O devices may be connected to the I/O bus **826**, such as I/O controller **828** and I/O device **830**, as illustrated.

[0116] I/O device **830** may also include an input device (not shown), such as an alphanumeric input device, including alphanumeric and other keys for communicating information and/or command selections to the processors **802-806**. Another type of user input device includes cursor control, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to the processors **802-806** and for controlling cursor movement on the display device.

[0117] System **800** may include a dynamic storage device, referred to as main memory **816**, or a random access memory (RAM) or other computer-readable devices coupled to the processor bus **812** for storing information and instructions to be executed by the processors **802-806**. Main memory **816** also may be used for storing temporary variables or other intermediate information during execution of instructions by the processors **802-806**. System **800** may include a read only memory (ROM) and/or other static storage device coupled to the processor bus **812** for storing static information and instructions for the processors **802-806**. The system set forth in FIG. **8** is but one possible example of a computer system that may employ or be configured in accordance with aspects of the present disclosure.

[0118] According to one embodiment, the above techniques may be performed by computer system **800** in response to processor **804** executing one or more sequences of one or more instructions contained in main memory **816**. These instructions may be read into main memory **816** from another machine-readable medium, such as a storage device. Execution of the sequences of instructions contained in main memory **816** may cause processors **802-806** to perform the process steps described herein. In alternative embodiments, circuitry may be used in place of or in combination with the software instructions. Thus, embodiments of the present disclosure may include both hardware and software components.

[0119] A machine readable medium includes any mechanism for storing or transmitting information in a form (e.g., software, processing application) readable by a machine (e.g., a computer). Such media may take the form of, but is not limited to, non-volatile media and volatile media and may include removable data storage media, non-removable data storage media, and/or external storage devices made available via a wired or wireless network architecture with such computer program products, including one or more database management products, web server products, application server products, and/or other additional software components. Examples of removable data storage media include Compact Disc Read-Only Memory (CD-ROM), Digital Versatile Disc Read-Only Memory (DVD-ROM), magneto-optical disks, flash drives, and the like. Examples of non-removable data storage media include internal magnetic hard disks, SSDs, and the like. The one or more memory devices may include volatile memory (e.g., dynamic random access memory (DRAM), static random access memory (SRAM), etc.) and/or non-volatile memory (e.g., read-only memory (ROM), flash memory, etc.).

[0120] Computer program products containing mechanisms to effectuate the systems and methods in accordance with the presently described technology may reside in main memory **816**, which may be referred to as machine-readable media. It will be appreciated that machine-readable media may include any tangible non-transitory medium that is capable of storing or encoding instructions to perform any one or more of the operations of the present disclosure for execution by a machine or

that is capable of storing or encoding data structures and/or modules utilized by or associated with such instructions. Machine-readable media may include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more executable instructions or data structures.

[0121] Embodiments of the present disclosure include various steps, which are described in this specification. The steps may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware, software and/or firmware.

[0122] Various modifications and additions can be made to the exemplary embodiments discussed without departing from the scope of the present disclosure. For example, while the embodiments described above refer to particular features, the scope of this disclosure also includes embodiments having different combinations of features and embodiments that do not include all of the described features. Accordingly, the scope of the present disclosure is intended to embrace all such alternatives, modifications, and variations together with all equivalents thereof.

Claims

1. A method comprising: encoding a firewall ruleset into a plurality of arrays each comprising a string of bits, wherein each bit of the plurality of arrays corresponds to one processing rule of the firewall ruleset, the encoding comprising utilizing at least one target address based on a determine mapping value; combining the plurality of arrays into a new array comprising a string of bits; obtaining, utilizing a first portion of a network address included in a received communication packet, the new array from a data structure; determining a bit value from the new array based on a second portion of the network address; and processing the received communication packet based on the bit value from the new array.
2. The method of claim 1, further comprising: shifting the target address a value in a direction according to a bitfield index of bits; determining a map index based on shifting; and determining a bit field index based on a value associated with the map index, the bit field index being utilized as a basis for the determined bit value.
3. The method of claim 2, further comprising: determining a bit field value based on the shifting and map index; determining a bit field mask, the bit field mask corresponding to a set bit value according to a bit field size, the bit value being based on the bit field mask; and determining the mapping value based on the bit field mask.
4. The method of claim 1 further comprising: storing the new array at a firewall device, wherein storing the new array consumes less memory space than a storing of the plurality of arrays.
5. The method of claim 1 wherein combining the plurality of arrays into the new array comprises: determining a union set of a blacklist array and a threatlist array, the blacklist array corresponding to a block processing rule of the firewall ruleset and the threatlist array corresponding to a re-direct processing rule of the firewall ruleset; and determining a difference set of the union set and a whitelist array, the whitelist array corresponding to an allow processing rule of the firewall ruleset, wherein combining the plurality of arrays into the new array further comprises: setting the difference set as the new array.
6. The method of claim 1 wherein the network address is an Internet Protocol (IP) address and the first portion comprises a first twenty-six bits of the network address.
7. The method of claim 1 wherein encoding the firewall ruleset comprises: asserting a bit of the string of bits of at least one of the plurality of arrays at a first bit position, the first bit position corresponding to a value equal to a second portion of the network address, wherein the second portion comprises a last six bits of the network address.
8. The method of claim 1, further comprising the first portion and the second portion being

variable, such that a bitmap field value associated with the network address is more than one bit.

9. The method of claim 1, wherein the first portion and the second portion are inversely proportional to an overall size of the network address.

10. The method of claim 1 wherein processing the received communication packet comprises: blocking the received communication to a destination address if the bit value from the new array is asserted.

11. The method of claim 1 wherein processing the received communication packet comprises: transmitting the received communication to a destination address if the bit value from the new array is de-asserted.

12. A network device comprising: a processor configured to: encode a firewall ruleset into a plurality of arrays each comprising a string of bits, wherein each bit of the plurality of arrays corresponds to one processing rule of the firewall ruleset, the encoding comprising utilizing at least one target address based on a determine mapping value; combine the plurality of arrays into a new array comprising a string of bits; obtain, utilizing a first portion of a network address included in a received communication packet, the new array from a data structure; determine a bit value from the new array based on a second portion of the network address; and process the received communication packet based on the bit value from the new array.

13. The network device of claim 12, wherein the processor is further configured to: shift the target address a value in a direction according to a bitfield index of bits; determine a map index based on shifting; determine a bit field index based on a value associated with the map index, the bit field index being utilized as a basis for the determined bit value; determine a bit field value based on the shifting and map index; and determine a bit field mask, the bit field mask corresponding to a set bit value according to a bit field size, the bit value being based on the bit field mask, wherein the processor is further configured to determine the mapping value based on the bit field mask.

14. The network device of claim 12 wherein the processor is further configured to: store the new array at a firewall device, wherein storing the new array consumes less memory space than a storing of the plurality of arrays.

15. The network device of claim 12 wherein the processor is further configured to: determine a union set of a blacklist array and a threatlist array, the blacklist array corresponding to a block processing rule of the firewall ruleset and the threatlist array corresponding to a re-direct processing rule of the firewall ruleset; and determine a difference set of the union set and a whitelist array, the whitelist array corresponding to an allow processing rule of the firewall ruleset, wherein combining the plurality of arrays into the new array further comprises: setting the difference set as the new array.

16. The network device of claim 12 wherein the network address is an Internet Protocol (IP) address and the first portion comprises a first twenty-six bits of the network address.

17. The network device of claim 12 wherein the processor is further configured to: assert a bit of the string of bits of at least one of the plurality of arrays at a first bit position, the first bit position corresponding to a value equal to a second portion of the network address, wherein the second portion comprises a last six bits of the network address.

18. The network device of claim 12, wherein the first portion and the second portion are variable, such that a bitmap field value associated with the network address is more than one bit, wherein the first portion and the second portion are inversely proportional to an overall size of the network address.

19. The network device of claim 12 wherein the processor is further configured to: block the received communication to a destination address if the bit value from the new array is asserted.

20. The network device of claim 12 wherein the processor is further configured to: transmit the received communication to a destination address if the bit value from the new array is de-asserted.
