

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: December 9, 2020

A. Khare, Ed.  
Juniper Networks, Inc.  
p. Bergeon, Ed.  
Nokia, Inc.  
J. Scudder  
Juniper Networks, Inc.  
L. Jalil  
M. Gallagher  
Verizon  
K. Kasavchenko  
NetScout  
June 7, 2020

BGP FlowSpec Payload Matching  
draft-khare-idr-bgp-flowspec-payload-match-06

## Abstract

The rise in frequency, volume, and pernicious effects of DDoS attacks has elevated them from fare for the specialist to generalist press. Numerous reports detail the taxonomy of DDoS types, the varying motivations of their attackers, as well as the resulting business and reputation loss of their targets.

BGP FlowSpec ([RFC 5575](#), "Dissemination of Flow Specification Rules") can be used to rapidly disseminate filters that thwart attacks, being particularly effective against the volumetric type. Operators can use existing FlowSpec components to match on pre-defined packet header fields. However recent enhancements to forwarding plane filter implementations allow matches at arbitrary locations within the packet header and, to some extent, the payload. This capability can be used to detect highly amplified attacks whose attack signature remains relatively constant while values in the packet header vary, as well as the burgeoning variety of tunneled traffic.

We define a new FlowSpec component, "Flexible Match Conditions", with similar matching semantics to those of existing components. This component will allow the operator to define bounded match conditions using bit offsets and a variety of match types.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

Internet-Draft

BGP FlowSpec Payload Matching

June 2020

working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 9, 2020.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Requirements Language . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Motivation . . . . .	<a href="#">3</a>
<a href="#">2.1.</a>	Machine analysis of DDoS attacks . . . . .	<a href="#">4</a>
<a href="#">2.1.1.</a>	Matching based on payload . . . . .	<a href="#">4</a>
<a href="#">2.1.2.</a>	Matching based on any protocol header field or across fields . . . . .	<a href="#">4</a>
<a href="#">2.2.</a>	Tunneled traffic . . . . .	<a href="#">5</a>
<a href="#">2.3.</a>	Non-IP traffic . . . . .	<a href="#">5</a>
<a href="#">3.</a>	Terminology . . . . .	<a href="#">5</a>
<a href="#">4.</a>	Defining the Search Boundary . . . . .	<a href="#">6</a>
<a href="#">4.1.</a>	Defining the Start of the Boundary . . . . .	<a href="#">6</a>
<a href="#">4.2.</a>	Defining the End of the Boundary . . . . .	<a href="#">7</a>
<a href="#">5.</a>	Specification . . . . .	<a href="#">8</a>
<a href="#">5.1.</a>	Value . . . . .	<a href="#">8</a>
<a href="#">5.1.1.</a>	Match Boundary . . . . .	<a href="#">8</a>

5.1.2.	Match Type . . . . .	8
5.1.2.1.	Bitmask match . . . . .	9
5.1.2.2.	Numeric range match . . . . .	9
5.1.2.3.	Regular expression string match types . . . . .	9
6.	Error Handling . . . . .	10

7.	Security Considerations . . . . .	10
8.	IANA Considerations . . . . .	10
9.	Acknowledgements . . . . .	11
10.	References . . . . .	11
10.1.	Normative References . . . . .	11
10.2.	Informative References . . . . .	11
10.3.	URIs . . . . .	12
	Authors' Addresses . . . . .	12

## 1. Introduction

BGP FlowSpec [[RFC5575](#)] can be used to rapidly disseminate filters that thwart attacks, being particularly effective against the volumetric type. Operators can use existing FlowSpec components to match on pre-defined packet header fields. However recent enhancements to forwarding plane filter implementations allow matches at arbitrary locations within the packet header and, to some extent, the payload. This capability can be used to detect highly amplified attacks whose attack signature remains relatively constant, while values in packet header vary. Varying values in packet headers generally make it challenging to mitigate such attacks.

We define a new FlowSpec component, "Flexible Match Conditions", with similar matching semantics to those of existing components. This component will allow the operator to define bounded match conditions using bit offsets and a variety of match types.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

## 2. Motivation

BGP FlowSpec couples both the advertisement of NLRI-specific match

conditions, as well as the forwarding instance to which the filter is attached. This makes sense since BGP FlowSpec advertisements are most commonly generated, or at least verified, by human operators. The operator finds it intuitive to configure match conditions as human-readable values, native to each address family.

It is much friendlier, for instance, to define a filter that matches a source address of 192.168.1.1/32, than it is to work with the equivalent binary representation of that IPv4 address. Further, it is easier to use field names such as 'IPv4 source address' as part of the match condition, than it is to demarc that field using byte and bit offsets.

However, there are a number of use cases that benefit from the latter, more machine-readable approach.

### [2.1.](#) Machine analysis of DDoS attacks

Launching a DDoS is easier and more cost-effective than ever. The will to attack matters more than wherewithal. Those with the inclination can initiate one from the comfort of their homes [[1](#)], or even buy DDoS-as-a-Service [[2](#)], complete with 24x7 support and flexible payment plans.

Despite their effectiveness, such attacks are easily thwarted - once identified. The challenge lies in fishing out a generally unvarying attack signature from a data stream. Machine analysis may prove superior here, given the size of input involved. The resulting pattern may not lie within a well-defined field; even if it happens to, it may be a more straight-forward workflow to have machine analysis result in a machine-readable filter.

Below we illustrate the need for the suggested approach with two use cases.

#### [2.1.1.](#) Matching based on payload

A vast majority of volumetric DDoS attacks are of reflection/amplification nature. They can often be identified by the UDP source port of a service that reflects and amplifies the attack traffic. However, there exist DDoS attack methodologies such as SSDP Diffraction or Bittorrent amplification where values in most of layer

3 and layer 4 header fields, including source and destination UDP ports, are varied. That makes it challenging if not impossible to classify and mitigate a DDoS attack based on existing Flow Specification components. At the same time these attacks very often have a constant pattern in payload. Using the pattern in payload as a matching criteria would help in mitigating such DDoS attacks.

#### 2.1.2. Matching based on any protocol header field or across fields

BGP FlowSpec [[RFC5575](#)] defines 12 Flow Specification component types that can be used to match traffic. However, a DDoS attack might result in illegitimate traffic of a specific pattern in a layer 3 or layer 4 header, and this pattern would not have a respective component type. Examples are Time to Live field of IP header or Window field of TCP header. In order to avoid extending BGP FlowSpec [[RFC5575](#)] with all theoretically possible component types, this document proposes divorcing the search boundary from having to align with header fields. This allows flexibly matching patterns

regardless of whether they have a currently matching component type as well as patterns that span fields.

#### 2.2. Tunneled traffic

Tunnels continue to proliferate due to the benefits they provide. They can help reduce state in the underlay network. Tunnels allow bypassing routing decisions of the transit network. Traffic that is tunneled is often done so to obscure or secure. Common tunnel types include IPsec [[RFC4301](#)], Generic Routing Encapsulation (GRE) [[RFC2890](#)], Virtual eXtensible Local Area Network (VXLAN) [[RFC7348](#)], GPRS Tunneling Protocol (GTP) [[GTPv1-U](#)], et al.

By definition, transit nodes that are not the endpoints of the tunnel hold no attendant control or management plane state. These very qualities make it challenging to filter tunneled traffic at non-endpoints. Often though, the forwarding hardware at these transit-only nodes is capable of reading the byte stream that comprises the protocol being tunneled. Despite this capability, it is usually infeasible to filter based on the content of this passenger protocol's header since BGP FlowSpec does not provide the operator a way to address arbitrary locations within a packet.

### [2.3.](#) Non-IP traffic

Not all traffic is forwarded as IP packets. Layer 2 services abound, including flavors of BGP-signaled Ethernet VPNs such as BGP-EVPN, BGP-VPLS, FEC 129 VPWS (LDP-signaled VPWS with BGP Auto-Discovery).

Ongoing efforts such as [[I-D.ietf-idr-flowspec-l2vpn](#)] offer one approach, which is to add layer 2 fields as additional match conditions. This may suffice if a filter needs to be applied only to layer 2, or only to layer 3 header fields.

## [3.](#) Terminology

**Header** Subset of datagram or packet that contains information that is required for delivery from source to destination.

**Payload** Remaining subset of datagram or packet that contains the information that is being transported.

**Field** A priori defined subset of the header with established semantics, acceptable value type and length.

**Type** How the encoded bits that comprise a field are interpreted. A well-defined type can be used to enforce notions of ordering, upper and lower bounds, and correctness. For instance, using a

signed integer type to count the number of packets received by a given forwarding element could result in negative values. To avoid that, the field should be typed as a zero-based counter. In order to avoid premature rollover, the counter should be sized appropriately. To prevent retrogression, the values should always be accumulated as it is impossible to receive fewer packets in toto. Defining this example field as an unsigned 64-bit field with monotonically incrementing values ensure it meets the appropriate objectives.

**Maximum Readable Length** The packet length in bits that a forwarding implementation can parse and make available for filtering. Abbreviated as MRL.

## [4.](#) Defining the Search Boundary

Based on this set of definitions, the flexible match operator requires three inputs to demarcate the search extents and the search term itself:

- o Where the match should begin: Where in the datagram or packet the search for matching values is initiated. This allows skipping over parts of the packet that are not of interest.
- o Where the match should end: Where in the datagram or packet the search ends.
- o What should be matched: A variety of search types, including exact numeric matches, matching a range of numeric values, and string-based matches.

#### [4.1.](#) Defining the Start of the Boundary

While intuitive to grasp, determining the search boundary requires explication. A canonical forwarding engine parses an incoming packet header and identifies it as belonging to a single Network Layer Reachability Information (NLRI), or address family. The contents of the header are parsed with address family specificity, in order to extract a forwarding lookup key. In the case of IPv4 unicast forwarding, this key is the IPv4 destination address. The key is used to look up the corresponding action in an address family specific forwarding table.

This does not preclude implementations from exposing additional packet headers to the operator, both encapsulating and encapsulated, to provide additional forwarding functionality. For instance, common stateless load balancing techniques involve reading fields in additional headers in order to increase entropy and preserve flow

ordering. As another example, in the case of Ethernet encapsulated IPv4 packets, a forwarding engine could allow filtering using the source or destination MAC address even though the forwarding decision is ultimately based only on the IPv4 header.

As yet another example, consider that a Virtual eXtensible Local Area Network (VXLAN) [[RFC7348](#)] packet has the following headers:

- o Outer Ethernet Header: Source MAC address of the originating VXLAN Tunnel End Point (VTEP). – Outer IPv4/IPv6 Header: Source IP address of the originating VXLAN Tunnel End Point (VTEP).
- o Outer UDP Header: Random source port used to generate entropy for load balancing, and destined to the IANA-assigned VXLAN port 4789.
- o VXLAN Header: Used to identify a specific VXLAN overlay network.
- o Inner Ethernet Header and payload: Original MAC frame encapsulated.

Forwarding at the tunnel midpoints, i.e., not the where tunnel imposition or disposition occur, makes use of the outer IPv4 header. In order to differentiate itself, a midpoint may provide the ability to parse and take the VXLAN header into account. This functionality could be used to implement access control or perform traffic telemetry.

In order to normalize behavior across forwarding implementations, the beginning of the search space MUST be aligned with the FlowSpec AFI/SAFI to which the flexible match rule belongs. For instance, with FlowSpec for IPv4 traffic, the match can only start at the first bit of the IPv4 header. Even if the forwarding implementation has the capability to read outer and inner headers, the start of the search extent is anchored at the IPv4 header.

#### [4.2.](#) Defining the End of the Boundary

Similarly, the end of the search boundary MUST be the lesser of either the last bit in a packet or the Maximum Readable Length ([Section 3](#)) that a forwarding implementation can parse from a packet and make available for filtering. As the MRL will be implementation-dependent, it needs to be known to the flexible filtering rules engine. That can be communicated out-of-band via configuration or signaled using future BGP or IGP extensions.

It is not required that all nodes in a flexible filtering domain be required to have a common or minimum MRL. This does not obviate the need for a rules engine to take MRL into account when creating

flexible filters. This is especially important as the rules engine

may not have direct BGP peering with all FlowSpec enforcers and may not receive a BGP Notification if it advertises a flexible match that exceeds the MRL of a given node.

## 5. Specification

We define a new FlowSpec component, Type TBD, named "Flexible Match Conditions".

Encoding: <type (1 octet), length (1 octet), value>

### 5.1. Value

The Value field contains the match boundary, match type, and term to match.

Encoding: <match boundary (2 octets), match type (1 octet), match term>

#### 5.1.1. Match Boundary

The match boundary is encoded as:

```

+-----+-----+-----+-----+-----+-----+-----+-----+ |u|bit
                                     o| byte offset | +-----+-----+-----+-----+-----+-----+

```

u - Currently unused. MUST be zero.

bit offset - The number of bits to ignore in the packet being matched, from the start of the search boundary.

byte offset - The number of bytes to ignore.

#### 5.1.2. Match Type

Currently the following match types are defined:

Value	Match Type
0	Bitmask match
1	Numeric range match
2	POSIX Regular expression (regex) string match
3	PCRE Regular expression (regex) string match

Match Types

Match types 0 and 1 MUST be implemented. All other types are optional.

#### [5.1.2.1.](#) Bitmask match

This is encoded as {prefix, mask}, of equal length.

prefix - Provides a bit string to be matched. The prefix and mask fields are bitwise AND'ed to create a resulting pattern.

mask - Paired with the prefix field to create a bit string match. An unset bit is treated as a 'do not care' bit in the corresponding position in the prefix field. When a bit is set in the mask, the value of the bit in the corresponding location in the prefix field must match exactly.

#### [5.1.2.2.](#) Numeric range match

This is encoded as {low value, high value}, treated as an inclusive range.

low - The low value of the desired numeric range. This value MUST be numerically lower than the high value.

high - The high value of the desired numeric range. This value MUST be numerically higher than the low value.

#### [5.1.2.3.](#) Regular expression string match types

Not every forwarding plane that supports filtering via FlowSpec is a hardware-accelerated Network Processor Unit (NPU) or Application-Specific Integrated Circuit (ASIC). Software-only forwarding planes, while less performant, may be able to filter on more complex match types.

There is a plethora of regular expression engines and their supported flavor. The two flavors this match type refers to are:

##### [5.1.2.3.1.](#) POSIX regular expression string match

This type refers to extended regular expression (ERE) as defined by [\[IEEE.1003-2.1992\]](#).

##### [5.1.2.3.2.](#) PCRE regular expression string match

This type refers to Perl compatible regular expression as defined by

## 6. Error Handling

Malicious, misbehaving, or misunderstanding implementations could advertise semantically incorrect values. Care must be taken to minimize fallout from attempting to parse such data. Any well-behaved implementation SHOULD verify that the minimum packet length undergoing a match equals (match start header length + byte offset + bit offset + value length).

## 7. Security Considerations

This document introduces no additional security considerations beyond those already covered in [RFC5575] .

## 8. IANA Considerations

IANA is requested to assign a type from the First Come First Served range of the "Flow Spec Component Types" registry:

Type Value	Name	Reference
TBD	Flexible Match Conditions	this document

IANA is requested to create a "BGP Flow Spec" group. The existing "Flow Spec Component Types" registry is to be a member of the "BGP Flow Spec" group.

IANA is requested to create a new registry, called "Flow Spec Match Type" within the (newly created) "BGP Flow Spec" group.

Reference: this document

Registry Owner/Change Controller: IESG

Registration procedures:

+-----+-----+

Range	Registration Procedures
0-127	IETF Review
128-249	First Come First Served
250-254	Experimental
255	Reserved

Note: a separate "owner" column is not provided because the owner of all registrations, once made, is "IESG".

IANA is requested to perform the following new allocations within the "Flow Spec Match Type" registry:

Value	Description	Reference
0	Bitmask match	this document
1	Numeric range match	this document
2	POSIX regular expression (regex) string match	this document
3	PCRE regular expression (regex) string match	this document

## 9. Acknowledgements

We wish to thank Philippe Bergeon, Ron Bonica, Jeff Haas, Sudipto Nandi, Brian St Pierre and Rafal Jan Szarecki for their valuable comments and suggestions on this document.

## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", [RFC 5575](#), DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.

## 10.2. Informative References

- [GTPv1-U] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 10.3.0, September 2011.
- [I-D.ietf-idr-flowspec-l2vpn] Weiguo, H., Eastlake, D., Litkowski, S., and S. Zhuang, "BGP Dissemination of L2 Flow Specification Rules", [draft-ietf-idr-flowspec-l2vpn-15](#) (work in progress), May 2020.

Khare, et al. Expires December 9, 2020 [Page 11]

---

Internet-Draft BGP FlowSpec Payload Matching June 2020

- [IEEE.1003-2.1992] Institute of Electrical and Electronics Engineers, "Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Standard 1003.2, 1992.
- [RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", [RFC 2890](#), DOI 10.17487/RFC2890, September 2000, <<https://www.rfc-editor.org/info/rfc2890>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", [RFC 7348](#), DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.

## 10.3. URIs

- [1] <https://github.com/649/Memcrashed-DDoS-Exploit>

- [2] <https://www.facebook.com/PutinStresser/photos/a.1687498801469198/2024483917770683/?type=3>
- [3] <https://www.pcre.org/original/pcre.txt>

#### Authors' Addresses

Anurag Khare (editor)  
Juniper Networks, Inc.  
2251 Corporate Park Drive  
Herndon, Virginia 20171  
US

Email: [anuragk@juniper.net](mailto:anuragk@juniper.net)

Philippe Bergeon (editor)  
Nokia, Inc.  
600 March Road  
Ottawa, Ontario K2K2E6  
CA

Email: [philippe.bergeon@nokia.com](mailto:philippe.bergeon@nokia.com)

Khare, et al.	Expires December 9, 2020	[Page 12]
---------------	--------------------------	-----------

---

Internet-Draft	BGP FlowSpec Payload Matching	June 2020
----------------	-------------------------------	-----------

John Scudder  
Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, CA 94089  
US

Email: [jgs@juniper.net](mailto:jgs@juniper.net)

Luay Jalil  
Verizon

Email: [luay.jalil@one.verizon.com](mailto:luay.jalil@one.verizon.com)

Michael Gallagher  
Verizon

Email: michael.gallagher@verizon.com

Kirill Kasavchenko  
NetScout

Email: Kirill.Kasavchenko@netscout.com