Network Working Group
Internet Draft
Intended status: Informational
Expires: July 2015

N Teague
Verisign Inc
January 14, 2015

Open Threat Signaling using RPC API over HTTPS and IPFIX
draft-teague-open-threat-signaling-00

Abstract

This document defines a method by which a device or application may signal information relating to current threat handling to other devices/applications that may reside locally or in the cloud. The initial focus is ddos mitigation; however, the method may be extended to communicate any threat type. This will allow for a vendor or provider agnostic approach to threat mitigation utilising multiple layers of protection as the operator sees fit.

The dissemination of threat information will occur utilising JSON RPC API over HTTPS communications between devices/applications and will be augmented by IPFIX and UDP for signaling telemetry information relating to attacks and protected object data.

An open standards based approach to communication between on-premise DDoS mitigation devices and cloud based DDoS protection services allows for enterprises to have a wider range of options to better secure their environments without the limitations of vendor lock-in.

Status of This Memo

Copyright Notice

1. Introduction

There are many devices and applications dealing with threat handling that may be a discrete part of a larger strategy. These elements may be required from time to time to signal to an upstream component or provider that the capabilities of the device are exceeded and that an offramping of attack traffic to a more capable element or infrastructure is desired or required. Signaling the need to off-ramp is not the only necessary feature; however, it is also desirable to communicate the form that the threat takes in order to accelerate the next layer mitigation process.

Although many vendors and providers implement their own variation or invest in integrating disparate APIs, we are proposing the adoption of a standard method for elements to signal allowing greater integration among any CPE (Customer Premise Equipment) device, service or cloud provider. In addition to the goal of interoperability, the intent is to present a robust method capable of continued signaling in the event of congested ingress paths to the originator. Stateful transport exchanges between components may leverage recognised JSON API channels in order to pass white & black lists, export collector information, protected object attributes, signature updates, mitigation details etc. These exchanges can occur at regular intervals during times of relative inactivity and could continue during attacks up to the point where a signaling component or path becomes overwhelmed. In parallel, a UDP IPFIX channel will export data pertaining to protected objects as well as current and ongoing incidents. The receiver for export will be delivered to the signaling component via the JSON API channel, allowing for the upstream element to set the destination dynamically. The UDP export will focus more specifically on communicating the current state of the threat and the component dealing with it. Should the signaling component risk becoming degraded, the telemetry data passed from this node will communicate this risk while also ensuring an upstream device or provider has the required information to take over traffic handling without the need to relearn and re-detect.

2. Data Dictionary

The data dictionary refers to a set of attributes common across implementations. The dictionary is not exhaustive and expansion is encouraged. The object definitions, as presented in this draft, are intended to communicate an event. An event will include a number of attributes which will identify an attack profile, the targeted resource, any existing mitigation actions being undertaken, sla information and scope. In certain circumstances, such as initial registration and discovery, it may be desirable to export information regarding protected objects currently managed by the signaling component outside the scope of any threat or action. These may be identified as informational when the record is accompanied by an event key of 0.

The event attributes will appear:

- Access Token
- Key
- Time
- Type (category and subtype)
- Description
- Counter
- Scope
- SOS
- Thresholds

* Access Token - authentication token (e.g. pre-shared nonce)
* Key - the signaling component specific event identifier.
* Time - the time the event was triggered. The timestamp of the record may be used to determine the resulting duration.
* Type - determined from the attack definitions
* Description - textual notes
* Scope - refers to the status of started, ended or ongoing
* SOS - this allows for a signaling component to simply communicate that further filtering by additional infrastructure, provider or cloud is necessary. This negates the need to perform additional analytics on traffic characteristics and load. This field should be ignored where the scope identifies an attack as having ended. The SOS field is expected to communicate whenever the signaling component is overwhelmed but in certain circumstances this may need to be set for any or all events, it should therefore not be exclusively tied to signaling components health.
* Thresholds - load_factor1 % of max, load_factor2 % of max

The above event attributes will be augmented by additional data relating to the resource being attacked and the current handling.

The protected object attributes will appear:

- Access Token
- Key
- Label
- IPv/Prefix
  * version
  * address/prefix
  * protocol
  * port(s)
- SLA/QoS
- Mitigation status
- B/W threshold
- Counters
  *CurrentPps
  *CurrentBps

*PeakPps
*PeakBps
*TypicalPps
*TypicalBps

The Access Token and Key elements correspond to those found in the event notifier. Timestamp may be derived from the export-timestamp in the IPFIX header.

* Label - textual label
* IPv/Prefix - identifies the protected object under attack including ip version, address, protocol, port
* SLA - expressed as the first three bits of the dscp field value. This will map to (lowest to highest) BE, CS1, CS2, CS3, CS4, and CS5. The purpose is for the upstream element or provider to be able to classify and handle attack traffic accordingly.
* Mitigation status - simple true or false to denote whether an active mitigation is occurring
* B/W threshold - event bandwidth as a % of overall capacity
* Rate/frequency - exports counters based upon current, peak and average bps/pps

The attack type identifier will be constructed from a category and a sub element. The category will be one of the high level types below with the sub element providing greater granularity into the event. This specific set of identifiers may be further expanded and a mechanism to update the attack dictionary across the JSON API channel or alternately for the elements to negotiate a standard set of definitions or an expanded set should be considered for a future iteration.

3. Attack/threat categories and sub elements

- Bandwidth - b/w exceeds available capacity or threshold
- Packet Rate - pps exceeds capacity or threshold
- Ipv4 Object - may be one or a combination of the following:
    - addr
    - protocol
    - src port
    - dscp
    - length
    - flags
    - ttl
    - martian
- Ipv6 Object - may be on or a combination of the following:
    - addr
    - protocol/next-header
    - src port
    - length
    - traffic class
    - hop limit
    - flow label
    - martian

- Packet Sanity - packets that fail basic sanity checks:
    - UDP packets with invalid UDP length
    - TCP packets with corrupt header
    - UDP/TCP with src/dst port 0
    - invalid version
    - invalid option
    - runt/giant/ping of death
    - land
    - fragments
- TCP - attacks against TCP:
    - syn abuse
    - ack abuse
    - fin abuse
    - rst abuse
    - psh abuse
    - urg abuse
    - window abuse
    - invalid TCP flags (null,xmas)
    - fragment abuse
    - invalid option
    - sockstress
- UDP - attacks against UDP:
    - flood abuse
    - fragment abuse
    - 0 payload
- ICMP - attacks against icmp:
    - flood

- Application - higher layer attacks:
    - hash collision
    - http
        - get flood
        - post flood
        - random/invalid url
        - slowloris
        - slow read
        - r-u-dead-yet (rudy)
        - url regex
        - malformed request
        - xss
    - https
        - ssl session exhaustion
    - dns
        - request spoofing
        - query flood
        - nxdomain flood
        - any flood
        - query regex
        - malformed query
        - response flood
        - dnssec abuse
    - sip
        - malformed request
    - sql
        - injection
- Amplification - amplified/amplifier attacks
    - dns
    - ntp
    - snmp
    - netbios
    - ssdp
    - chargen
    - qotd
    - bittorrent
    - kad
    - smurf
    - quake
    - steam
- Intrusion - potential intrusion or nuisance
    - port scan
    - buffer overflow
    - well know threat identifiers (CERT, emerging threats etc.)
- Custom - used for arbitrary definitions that may not yet be part of the standard
    - custom1
    - custom2
    - etc.

An event will be triggered based on the attack profile. E.g. application:http-slowloris and icmp:flood would be considered 2x separate events. The ability to roll individual events into a parent event id is also permissible. In these instances the ability to identify a parent event would be necessary. A device may use a threat data field in the export to communicate a sample payload for scrutiny by an upstream system or provider and on which a signature based filter may be based.

4. Threat enumeration

Threats will be identified using a 16 bit format split into 2x octets, the 1st octet will identify the category where there 2nd octet will relate to a specific sub type.

Category	ID	SubType	ID
Bandwidth	1		
Packet Rate	10		
IPv4 Object	11		
		addr	0b1
		protocol	0b10
		port	0b11
		src port	0b100
		dscp	0b101
		length	0b110
		flags	0b111
		ttl	0b1000
		martian	0b1001
IPv6 Object	100		
		addr	0b1
		protocol/nh	0b11
		src port	0b100
		length	0b101
		traffic class	0b110
		hop limit	0b111
		flow label	0b1000
		martian	0b1001

Packet Sanity	101		
		UDP length	0b1
		TCP corrupt header	0b10
		UDP/TCP src port 0	0b11
		invalid version	0b100
		invalid option	0b101
		runt/giant/ping of death	0b110
		land	0b111
		fragments	0b1000
TCP	110		
		syn abuse	0b1
		ack abuse	0b10
		fin abuse	0b11
		rst abuse	0b100
		psh abuse	0b101
		urg abuse	0b111
		window abuse	0b1000
		invalid TCP flags	0b1001
		fragment abuse	0b1010
		invalid option	0b1011
		sockstress	0b1100
UDP	111		
		flood abuse	0b1
		fragment abuse	0b10
		0 payload	0b11
ICMP	1000		
		flood	0b1
Application	1001		
		hash collision	0b1
		http - get flood	0b10
		http - post flood	0b11
		http - random/invalid url	0b100
		http - slowloris	0b101
		http - slow read	0b110
		http - r-u-dead-yet (rudy)	0b111
		http - malformed request	0b1000
		http - xss	0b1001
		https - ssl session exhaustion	0b1010
		dns - request spoofing	0b1011
		dns - query flood	0b1100
		dns - nxdomain flood	0b1101
		dns - query regex	0b1110
		dns - malformed query	0b1111
		dns - response flood	0b10000
		dns - dnssec abuse	0b10001
		sip - malformed request	0b10010
		sql - injection	0b10011

Amplification	1010		
		dns	0b1
		ntp	0b10
		snmp	0b11
		netbios	0b100
		ssdp	0b101
		chargen	0b110
		qotd	0b111
		bittorrent	0b1000
		kad	0b1001
		smurf	0b1010
		quake	0b1011
		steam	0b1100
		Intrusion	0b1011
		port scan	0b1
		buffer overflow	0b10
		well known - emerging threats	0b11
		well known - us-cert	0b100
		well known - idefence	0b101
...	...	...	...
Custom	11111111		
		custom1	0b1
		custom2	0b10
		custom3	0b11
		custom4	0b100
		custom5	0b101
		custom6	0b110
		custom7	0b111
		custom8	0b1000
		...	...

5. JSON RPC API over HTTPS communication

The JSON API channel is expected to be opened at regular intervals for the exchange of command and control data. The signaling component will authenticate using a standard user/role:password or api-key and request URL:{scheme}://{host}:{port}/ocs/api/cloudinfo using a POST method with a request body of {"device_ip":"<device ip>", "load_factor1":"<alias>", "load_factor2":"<alias>"...}. The upstream element will use the access token plus ip address to verify the originators credentials as valid signaling component. The upstream element may then pass to the requesting component the IPFIX ID token, the IPFIX destination address, white lists and mitigation information.

```
METHOD:POST
URL:{scheme}://{host}:{port}/ocs/api/cloudinfo
Request Body:
{"device_ip":"<device ip>", "load_factor1": "<alias>",
"load_factor2": "<alias>" }
Response Body:
{
"access_token":"<Access-Token>",
"export_host":"<ip>",
"whitelist_ips":["<ip1>","<ip2>"..],
"device_threshold_config": {"load_factor1": "% of max",
"load_factor2": "% of max", "load_factor3":"% of max"}
"mitigation_info":{"status":"<status(Inactive,Monitoring,Mitigating)>"
,"swing_flag":"<true or false>",
"blacklistaddrs":["<ip1>","<ip2>"..]}
"custom":"arbitrary data"
}
```

Request Body:

The device_ip attribute simply details the signaling components source address. The device may also communicate the aliases assigned to local load factors of interest e.g. cpu, memory, state table etc.

Response Body:

The access_token will be used for basic authentication of IPFIX exports to the upstream collector.

The export_host will communicate the ipv4/ipv6 addr of the upstream IPFIX collector.

The whitelist_ips attribute will allow for an provider or cloud instance to white list certain ip addresses from which all traffic should be accepted to ensure that any proxied traffic where the original address is obscured is not mistaken for a new attack signature.

The device_threshold_config is an extensible object which allows the upstream element to set thresholds of common metrics at which to trigger an SOS=true. The common metrics are described as load_factors and may be device specific, these should be expressed as a % of the maximum capacity.

The mitigation_info object is also extensible and will communicate the current status, offramp/restoration status and any relevant black list information.

The status attribute of the mitigation_info object as 3x states:

* Inactive - no IPFIX messages have been received in the last
health-refresh-timeout period.
* Monitoring - IPFIX messages are being received
* Mitigating - the upstream is actively mitigating a threat

The swing attribute of the mitigation_info object is set either true or false:

* True - the attack has abated or reduced (if volumetric) to a level deemed within the capacity of the original signaling component.
* False - the attack mitigation should continue to be handled by the upstream element.

The blacklistaddrs attribute is a simple set of ipv4 or ipv6 addresses and allows an upstream element to communicate known bad actors or compromised hosts to the signaling component.

The custom field may be used for the upstream element to communicate an arbitrary object. This could include a portal url in the cloud or some other yet to be standardised data.

5.1 JSON API Example interaction:

```
      +-+-+                    +-+-+
      | D |---------HTTPS-------| C |
      +-+-+                    +-+-+
        |                        |
        |                      +-+-+
        +-----------IPFIX-------| I |
                               +-+-+
```

D = DDoS mitigation device 192.0.2.1
C = Cloud provider 198.51.100.1
I = IPFIX receiver 203.0.113.1

D initiates an https connection to C:
URL: https://user:password@198.51.100.1:443/ocs/api/cloudinfo

D posts:

{
"device_ip":"192.0.2.1"
}

C responds:

{
"access_token":"abc123",
"export_host":"203.0.113.1",
"whitelist_ips":["203.0.113.254","203.0.113.253"],
"device_threshold_config":
{"load_factor1": "85", "load_factor2": "75", "load_factor3": "85"},
"mitigation_info":
{"status":"Inactive","swing_flag":"True", "blacklistaddrs":[""]},
"custom":{"portal_url":"https://portal.cloud.net/Mitige?=192.0.2.1"}
}

Upon receipt of the response body the device 192.0.2.1 would now send event exports to the IPFIX receiver at 203.0.113.1 and would authenticate using the ID "abc123". Periodically a new token may be exchanged or an alternate IPFIX destination (export_host) set. In these instances the signaling component should start using the new credentials or destination immediately. The component will whitelist the ip addresses of 203.0.113.254 and 203.0.113.253. The SOS flag will be set to true should the component cpu=85% or mem=85% or bandwidth=75%.

6. IPFIX export

IPFIX was selected for this channel due to its push nature, extensible templates and its existing availability on ddos and security platforms. Leveraging an existing protocol will result in minimal retooling and hopefully lower any barrier to adoption.

An attack will trigger the creation of an incident record on the component which in turn will trigger IPFIX export to an upstream device or provider with details of the attack parameters. Due to the unreliable nature of UDP event data sets will repeat at regular intervals for the duration of the attack.

An attack may generate different data exports which will communicate various facets of the threat, the target and the overall incident. The event data set will define the base key and this will be used to link other records such as protected objects and threat profile data sets. Corresponding data sets referencing the same key will be considered part of the same event when combined with the component id.

An IPFIX event data export may be used as a heartbeat between elements. It is recommended that the signaling component periodically send heartbeats upstream to verify its status during periods of relative inactivity, failure by the upstream to receive these heartbeats may then trigger an alert or further investigation into why they never reached their destination.

The template for events will contain 8x fields as detailed:

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Set ID = 2            |       Length                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Template ID n           |       Field Count = 8         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|        Access Token         |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Key               |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Time              |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Type              |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|         Description         |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Scope             |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           SOS               |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|        Thresholds           |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The template for protected object will contain 16x fields as detailed:

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Set ID = 2            |       Length                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Template ID n           |       Field Count = 16        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|         Access Token        |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|             Key             |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|            Label            |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|          IP version         |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|          Address/Prefix     |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Protocol          |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|            Port             |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           SLA Code Point    |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|         Mitigation Status   |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|          B/W Threshold      |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Current Pps       |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Current Bps       |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|            Peak Pps         |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|            Peak Bps         |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Typical Pps       |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Typical Bps       |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The template for attack and threat identification will contain
4x fields as detailed:

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Set ID = 2             |       Length                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Template ID n            |       Field Count = 4         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|       Access Token           |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|           Key                |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|      Threat Identifier       |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1|        Threat Data           |       Field Length = n        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note - where no threat data is required to aid in mitigation (ie the identifier is enough) the Threat Data field may be set to null.

7. Security Considerations

The protocol described here serves as a security mitigation tool. Potential vulnerabilities of this system are addressed by the use of encrypted channels for communication between the elements and the use of low overhead control signals in case there is denial of service or congestion affecting the paths between the elements. The security considerations of [RFC7011] and [RFC5405] apply to the IPFIX and UDP based channels respectively. Additional security considerations will be added to subsequent drafts.

8. IANA Considerations

There are not expected to be requests to the IANA in relation to this memo.

9. Contributors

This document represents a collaborative effort by engineers at Verisign and Juniper to create a candidate for an open standards effort supporting communication between on-premise DDoS mitigation devices and cloud based DDoS mitigation services. A standards based approach allows businesses to have a wider range of options to better secure their complex environments without the limitation of vendor lock-in. The companies have published a draft specification through the Internet Engineering Task Force (IETF) to encourage community participation and further development of these proposals toward becoming an open standard.

10. Acknowledgements

The following people are acknowledged for their technical contributions in the development of this document:
Aziz Mohaisen, Jon Shallow, Suresh Bhogavilli, Jeshmi Raman, Malathy Poruran

11. Normative References

[RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Echange of Flow Information" https://tools.ietf.org/html/rfc7011 September 2013

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP usage Guidelines for Application Designers" https://tools.ietf.org/html/rfc5405 November 2008

12. Authors' Addresses

Nik Teague
Verisign Inc.
12061 Bluemont Way
Reston, VA 20190
US

Phone: +1 703 948 3200
Email: nteague@verisign.com
URI: http://www.verisigninc.com