INTERNET-DRAFT
Intended Status: Informational
Expires: April 17, 2016

T. Herbert
Facebook
L. Yong
Huawei USA

October 15, 2015

UDP Magic Numbers
draft-herbert-udp-magic-numbers-01

Abstract

This specification defines magic numbers in UDP which allow a node to determine or confirm the protocol contained in a UDP payload. This is primarily applicable for encapsulation and transport protocols encapsulated within UDP where intermediate devices, such as middle boxes, need to parse these protocols for providing service. Magic numbers can also be used to multiplex different UDP encapsulated protocols over the same UDP port.

Table of Contents

1 Introduction

Several transport and encapsulation protocols have been defined to be encapsulated within UDP [RFC0768]. In this model, the payload of a UDP packet contains a protocol header and payload for an encapsulated protocol. Transport protocols encapsulated in UDP include QUIC [QUIC], SCTP-in-UDP [RFC6951], and SPUD [I-D.hildebrand-spud-prototype]. Encapsulation protocols include Geneve [I-D.ietf-nvo3-geneve], VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], GUE [I-D.ietf-nvo3-gue], MPLS-in-UDP [RFC7510], and GRE-in-UDP [I-D.ietf-tsvwg-gre-in-udp-encap]. For various reasons, intermediate devices in a network may want to parse these protocols. For instance, a middlebox would need to parse an encapsulated transport protocol to implement a stateful firewall. To parse the encapsulated protocol in a UDP packet, a node must positively identify the encapsulated protocol.

The destination UDP port number is commonly used to interpret the contents of a UDP payload, however this is problematic in intermediate devices for several reasons:

- Port numbers can only be correctly interpreted by the endpoints. Interpretation by intermediate devices in the network may be incorrect ([RFC7605]]).

- Encapsulation and transport protocols will usually have assigned UDP ports, but they are not restricted to use only those.

- UDP encapsulated protocols may use a "substrate" protocol header as espoused in SPUD. Use of a substrate header may be common across several port numbers. Configuring each network device for each port that uses the substrate could be cumbersome.

This specification describes UDP magic numbers which allows network nodes to identify UDP encapsulated protocols without relying solely on UDP port numbers. A UDP magic number is a protocol specific, constant value which is logically inserted between the UDP header and the encapsulated protocol header. If a node matches the magic number in a packet to a known protocol’s magic number, then it can parse the encapsulated payload per the matched protocol. Each UDP encapsulated protocol uses a different magic number which allows multiplexing multiple encapsulated protocols over the same UDP port.

Note that the use of magic numbers is inherently probabilistic. It is possible that a UDP packet may have a payload that inadvertently matches a magic number. The magic number is defined to minimize the probability of this occurring (1/2^^64 assuming that UDP data has a random distribution), nevertheless the probability is non-zero. The consequences of incorrectly matching a UDP packet should be

considered for each UDP encapsulated protocol. An encapsulated protocol may include its own verification to ensure correct interpretation.

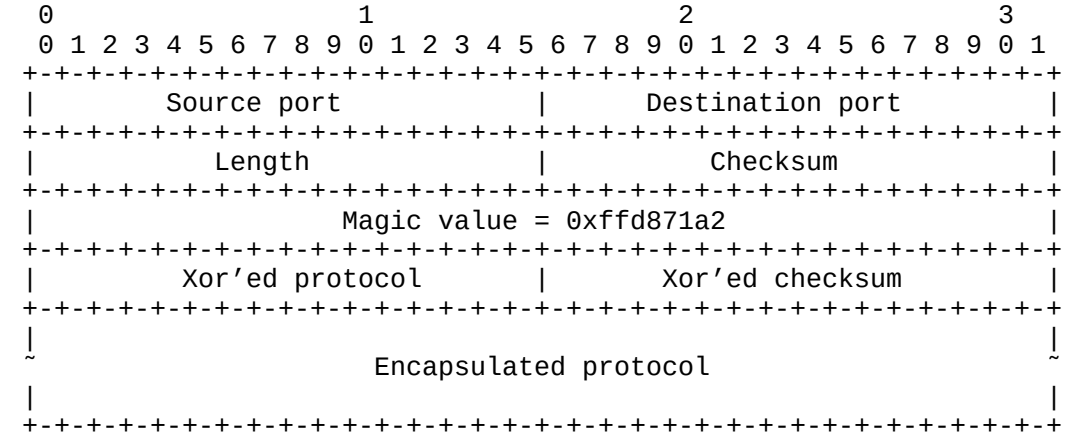
The use of magic numbers to identify UDP encapsulated protocols was specified in the SPUD prototype protocol ([I-D.hildebrand-spud-prototype]) and in "Session Traversal Utilities for NAT (STUN)" ([RFC5389]). This proposal generalizes the concept.

1.1 Terminology

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 Magic number format

The UDP magic number is a sixty-four bit value that includes a fixed constant, an encapsulated protocol type, and a checksum. The magic number within a UDP packet is diagrammed below.



The fields of the magic number are:

- Magic value: A fixed constant of 0xffd871a2. This value is the same for all encapsulated protocol types.
- Xor'ed protocol: Indicates the protocol type of the encapsulated protocol. The value in the field is a protocol type number exclusive or'ed with 0x36b4.
- Xor'ed Checksum: Indicates the standard one's complement

checksum over the magic number (including the Magic value and Xor'ed Protocol fields). The value in this field is the calculated checksum exclusive or'ed with 0x5ce9.

- o Encapsulated protocol: This contains the header and payload of the encapsulated protocol. The type for the protocol is indicated in the Xor'ed protocol field.

2.1 Magic value

The first byte of the Magic value field is 0xff and the other three bytes are a randomly chosen constant.

For UDP encapsulated protocols that allow magic number use to be optional, the magic number must be clearly distinguishable from a valid header. Each such protocol must declare that a header which would match the associated magic number is invalid. The value of 0xff as the first byte in the magic number was chosen as a likely value that would indicate an invalid header. It is common that the first byte of a protocol header contains a version number, and most protocols have not gotten past version zero. So if a magic number is received by a node that does not yet support magic numbers, the UDP payload would likely be interpreted as a protocol header with a bad version number; this should result in dropping the packet and not misinterpreting it. In this way, the use of magic numbers can be enabled for many existing protocols with forward compatibility.

2.2 Protocol types

Protocol types can generally refer to any encapsulation, transport, substrate, or application specific protocol that is encapsulated in UDP for which intermediate devices might need to parse. A protocol type number is encoded in UDP magic numbers to allow intermediate devices to distinguish different payload types while still using a common magic number format.

Protocol types are indicated by sixteen bits numbers, and the space is divided into three regions.

Numbers 0-49151 are reserved to mirror the assigned UDP port number space. If a port number is assigned to a UDP encapsulated protocol, that same number can be used as the protocol type number. This is allowed for convenience, there is no required correlation between protocol type numbers and UDP port numbers.

Numbers 49152-57343 are reserved for assigned protocol types.

Numbers 57344-65535 are reserved for private protocol types.

The Xor'ed protocol field in a magic number is a protocol type number exclusive or'ed with 0x36b4.

2.3 Magic number checksum

The magic number checksum is calculated as the standards one complement checksum computed over the sixty-four bit magic number where the Xor'ed checksum field is set to zero for the purposes of calculation. The checksum calculation covers the Magic value and the Xor'ed protocol fields. The Xor'ed checksum field is set to the result of the calculation exclusive or'ed with 0x5ce9.

Note that the magic number checksum is performed over constant fields and is itself a constant value per protocol type. An implementation should not need to perform this calculation when processing packets. Appendix A demonstrates how the checksum is applied to create a magic number constant for Generic UDP Encapsulation.

The magic number checksum may be used to validate the presence of a well formed UDP magic number. This is demonstrated in Appendix B.

3 Usage

This section describes the processing of UDP magic numbers on end hosts and intermediate devices.

3.1 End hosts

The use of UDP magic numbers is enabled on a per port basis. Magic numbers may be required for every UDP packet sent on a port, or may be optional. If a UDP port is assigned to a single protocol, the magic number in packets sent to that port is the one assigned to the protocol. If different encapsulated protocols are multiplexed on the same UDP port, magic numbers for those protocols will be used.

3.1.1 Required magic numbers

If magic numbers are required for a UDP port, a sender must set the magic number in any packets sent to the destination port. A receiver must check for a valid magic number. If the magic number is valid, that is the Magic value is correct and the protocol type is supported by the receiver for the port, then the packet is accepted. Otherwise, the magic number is not matched so the packet is dropped.

3.1.2 Optional magic numbers

When magic numbers are optional for a UDP port, a receiver must check if a magic number is present in a received packet. If a magic number

is matched for a protocol type supported by the receiver, then the packet must be accepted and the Encapsulated protocol in the packet is processed according to the protocol type. If the magic number is not matched, the packet is still accepted and the UDP payload is processed as a protocol type implied by the port number.

If it is not feasible in a protocol to distinguish a magic number from a valid header (MPLS-in-UDP for instance), UDP magic numbers cannot be optional on the protocol's port number. They can be used on a separate port number for which magic numbers would be required.

3.1.3 Use with DTLS

UDP magic numbers are intended to occupy the first bytes of the UDP payload to facilitate interpretation at middleboxes. When they are used with DTLS [RFC6347], the magic number must precede the DTLS headers. The protocol type in the magic number would refer to the payload type contained in DTLS.

3.2 Intermediate devices

Intermediate devices may match magic numbers in two ways:

- Match both the destination port and magic numbers associated with the port.

- Match magic numbers across a range (possibly all) of ports.

Matching both the port and magic number is recommended. This is feasible in cases where a UDP encapsulated protocol has an assigned port number. Matching the port number and magic number significantly reduces the possibility of misinterpreting a packet.

Matching just the magic number and not a port may be done when UDP encapsulated protocols are used on unassigned ports, or configuring port numbers on intermediate devices is prohibitive.

In either case, if a middlebox is able to match a magic number it may parse the encapsulated payload of the packet for the associated protocol.

If a middle box does not match a magic number for a packet it should follow default processing for UDP packets. If magic numbers are known to be required for a port, a middlebox may perform some alternative processing when the magic number is not present. This alternative processing should not be more restrictive than had the packet been sent to another arbitrary UDP port. In particular, if UDP packets for other ports would not be dropped, failure to match a magic number

should not result in the packet being dropped.

4 Security Considerations

UDP magic numbers are not a security mechanism and should not increase security risk.

5 IANA Considerations

IANA will be requested to create a "UDP Magic Number Protocol Type" registry to allocate protocol types. This shall be a registry of 16-bit values along with descriptive strings. The allocation ranges are described in section 2.2.

6 References

6.1 Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

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

[RFC1776] Crocker, S., "The Address is the Message", RFC 1776, DOI 10.17487/RFC1776, April 1 1995, <http://www.rfc-editor.org/info/rfc1776>.

[TRUTHS] Callon, R., "The Twelve Networking Truths", RFC 1925, DOI 10.17487/RFC1925, April 1 1996, <http://www.rfc-editor.org/info/rfc1925>.

6.2 Informative References

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, May 2013, <http://www.rfc-editor.org/info/rfc6951>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[QUIC] Roskind, J., "QUIC: Multiplexed Stream Transport Over UDP", http://www.ietf.org/proceedings/88/slides/slides-88-

tsvarea-10.pdf

[I-D.hildebrand-spud-prototype] Hildebrand, J. and Trammell, B. "Substrate Protocol for User Datagrams (SPUD) Prototype", draft-hildebrand-spud-prototype-03 (work in preogress), March 2015.

[I-D.ietf-nvo3-geneve] Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-00, May 2015.

[I-D.ietf-nvo3-vxlan-gpe] Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X, Elzur, U., Garg, P., and Melman, D. "Generic Protocol Extension for VXLAN" draft-ietf-nvo3-vxlan-gpe-00draft-quinn-vxlan-gpe-00, February 2015

[I-D.ietf-nvo3-gue] Herbert, T., Yong, L., and Zia, O., "Generic UDP Encapsulation", draft-ietf-nvo3-gue-01 (work in progress), June 2015.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, April 2015, <http://www.rfc-editor.org/info/rfc7510>.

[I-D.ietf-tsvwg-gre-in-udp-encap] Crabbe, E., Yong, L., Xu, X., and Herbert, T. "GRE-in-UDP Encapsulation", draft-ietf-tsvwg-gre-in-udp-encap-07, July 2015

[RFC7605] Touch, J., "Recommendations on Using Assigned Transport Port Numbers", BCP 165, RFC 7605, DOI 10.17487/RFC7605, August 2015, <http://www.rfc-editor.org/info/rfc7605>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[EVILBIT] Bellovin, S., "The Security Flag in the IPv4 Header", RFC 3514, DOI 10.17487/RFC3514, April 1 2003, <http://www.rfc-editor.org/info/rfc3514>.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, DOI 10.17487/RFC5513, April 1 2009, <http://www.rfc-editor.org/info/rfc5513>.

[RFC5514] Vyncke, E., "IPv6 over Social Networks", RFC 5514, DOI 10.17487/RFC5514, April 1 2009, <http://www.rfc-

editor.org/info/rfc5514>.

Appendix A: Example of creating a UDP magic number

This section demonstrates how a magic can be created for a UDP encapsulated protocol. For this example we consider Generic UDP Encapsulation (GUE), and assume that the assigned port number is used as the protocol type number.

The assigned port number for GUE in 6080 or 0x17c0 in hexadecimal. So the value of the Xor'ed protocol field is:

0x17c0 ^ 0x36b4 = 0x2174

To compute the magic checksum we first sum the words of the Magic value and the Xor'ed protocol field value computed above:

0xffd8 + 0x71a2 + 0x2174 = 0x192ee

The result is folded and then complemented:

(0x92ee + 1) ^ 0xffff = 0x6d10

So the value in the Xor'ed checksum field is:

0x6d10 ^ 0x5ce9 = 0x31f9

Thus the full 64 bit magic number value for GUE is:

0xffd871a2:0x217431f9

Appendix B: Checking magic numbers

This section provides some guidelines for how to check magic numbers.

B.1 Matching a single magic number

When a port supports precisely one protocol type there is only one magic number to check. This will be a common case at a receiver where magic numbers are enabled for encapsulated protocols that have assigned ports. Receiver processing in pseudo code may be:

```
dataptr = UDP_payload_ptr
good_magic = false
PROTO_MAGIC_NUMBER = Pre computed 64 bit value for protocol type

if (UDP_payload_length >= 8 &&
    memcmp(UDP_payload_ptr, PROTO_MAGIC_NUMBER, 8) == 0) {
```

```
            /* Magic number matched, skip it for further processing */
            dataptr += 8
            good_magic = true
    }

    if (good_magic || magic_numbers_are_optional)
            process_packet(dataptr)
    else
        /* Handle bad packet */
```

B.2 Matching against a set of magic numbers

A host needs to check against a set of magic numbers when different encapsulated protocols are multiplexed over a single port, and an intermediate device checks against a set when matching magic numbers across a range of ports. In either case, the typical method is to check the first four bytes of the UDP payload against the constant magic number value. If this is a match then the protocol type number is extracted and a lookup is performed to find a context. If a context is found, the checksum field in the packet is compared against a precomputed value in the context. In pseudo code this is:

```
dataptr = UDP_payload_ptr;
good_magic = false;

if (UDP_payload_length >= 8 &&
    *(u32 *)UDP_payload_ptr == 0xffd871a2) {
       proto = *(u16 *)(UDP_payload_ptr + 4) ^ 0x36b4
       checksum = *(u16 *)(UDP_payload_ptr + 6)
       ctx = protocol_lookup(proto)
       if (ctx && checksum == ctx->checksum) {
              /* Protocol found and matched */
              good_magic = true
              dataptr += 8;
       }
}

if (good_magic)
       process_as_protocol(dataptr, proto);
else
       /* Handle bad packet */
```

B.3 Magic number validation

A node can validate that a magic number is well formed for any protocol. This requires checking the Magic value is correct and verifying the checksum. In pseudo code this would be:

```
good_magic = false
u16 checksum(start, len) /* Checksum function */
if (UDP_payload_length >= 8 &&
    *(u32 *)UDP_payload_ptr == 0xffd871a2) {
        csum = checksum(UDP_payload_ptr, 6)
        if (csum ^ 0x5ce9 == *(u16 *)(UDP_payload_ptr + 6))
               good_magic = true
}
```

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA
US

EMail: tom@herbertland.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75024
US

Email: lucy.yong@huawei.com