XEP-0205: Best Practices to Discourage Denial of Service Attacks

Peter Saint-Andre
mailto:stpeter@jabber.org
xmpp:stpeter@jabber.org
https://stpeter.im/

2009-01-07
Version 1.0

Status	Type	Short Name
Active	Informational	N/A

This document recommends a number of practices that can help discourage denial of service attacks on XMPP-based networks.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

A key factor in the reliability and security of network infrastructure is its resilience in the face of denial of service attacks (see RFC 4732 [1]). Although the existing network of servers and clients that communicate via the Extensible Messaging and Presence Protocol (XMPP; see RFC 3920 [2]) has not yet been subject to such attacks, that is no cause for complacency. Therefore this document specifies a set of best practices that server implementations and deployments can follow in order to reduce the likelihood of denial of service attacks on the Jabber network.

2 Potential Attacks

RFC 4732 defines denial of service as follows:
A Denial-of-Service (DoS) attack is an attack in which one or more machines target a victim and attempt to prevent the victim from doing useful work. The victim can be a network server, client or router, a network link or an entire network, an individual Internet user or a company doing business using the Internet, an Internet Service Provider (ISP), country, or any combination of or variant on these.
The authors list a number of attacks, of which the following seem most likely against XMPP systems:

1. Exploiting poor software quality (e.g., buffer overlow attacks).
2. Exhausting application or operating system resources (e.g., memory, CPU cycles, disk space, configured maximum simultaneous connections).
3. Triggering lockouts and quota exhaustion (e.g., quotas associated with the bandwidth limits common in numerous XMPP server implementations.
4. Hijacking the TCP connection of a client or server (see Improving TCP's Robustness to Blind In-Window Attacks [3]).
5. Attacking the Domain Name System (DNS) infrastructure on which XMPP typically depends.
6. Poisoning blacklists.
7. Amplifying network traffic (this could be done through reflectors such as Multi-User Chat [4] and Publish-Subscribe [5] services).

[1] RFC 4732: Internet Denial-of-Service Considerations <http://tools.ietf.org/html/rfc4732>.
[2] RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc3920>.
[3] Improving TCP's Robustness to Blind In-Window Attacks <http://tools.ietf.org/html/draft-ietf-tcpm-tcpsecure>. Work in progress.
[4] XEP-0045: Multi-User Chat <http://xmpp.org/extensions/xep-0045.html>.
[5] XEP-0060: Publish-Subscribe <http://xmpp.org/extensions/xep-0060.html>.

3 Potential Solutions

Numerous potential solutions have been suggested to deal with the threat of denial of service attacks against XMPP servers, including the following:

1. Limiting the number of connections that a server will accept from a given IP address at any one time. Such a limit may help to prevent automated processes from exhausting the server's resources (such as available ports or XML parser processing resources).

2. Limiting the number of connection attempts (via the TCP binding specified in RFC 3920 or via the BOSH [6]) that a server will accept from a given IP address in a given time period. Such a limit may help to prevent automated processes from exhausting the server's resources (such as available ports or XML parser processing capacity).

3. Limiting the number of authentication attempts for a given Jabber ID in a given time period. While such a limit may seem beneficial, in fact it might result in locking out the legitimate owner of a Jabber ID if a malicious entity attempts a large number of illegitimate authentication attempts for the Jabber ID; therefore such a limit is not recommended except as described in Section 7.3.5 of rfc3920bis [7], and it is instead recommended to limit the number of connections and connection attempts on a per-IP basis.

4. Disallowing unauthenticated connections from clients and from peer servers; as mentioned below, this is required by RFC 3920.

5. Limiting the number of XMPP resource identifiers allowed to an account at any one time. This may help to prevent a rogue account from creating an unlimited number of sessions and therefore exhausting the resources of the server's session manager.

6. Limiting the absolute size in bytes of XML stanzas accepted by the server, or of particular aspects of an XML stanza (e.g., attribute values, element names, XML character data). Limits on particular aspects of an XML stanza are probably not needed, as long as it is possible to limit the absolute size of each XML stanza, since such a limit may help to prevent exhaustion of server resources (e.g., XML parser processing capacity).

7. Limiting the number of bytes or XML stanzas that a server will accept over a given TCP connection or for a given JabberID in a given time period. Such a limit, which helps

[6] XEP-0124: Bidirectional-streams Over Synchronous HTTP <http://xmpp.org/extensions/xep-0124.html>.

[7] rfc3920bis: proposed revisions to Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/draft-ietf-xmpp-3920bis>. (work in progress)

to prevent rogue accounts or hijacked clients from flooding the server, is common in existing XMPP server implementations and often goes by the name ”karma”.

8. Limiting the number of XML stanzas that a connected client may send to different recipients in a given time period. Such a limit may help to prevent automated processes (e.g., bots) from sending unsolicited bulk communications (known in the instant messaging domain as ”spim”).

9. Limiting or prohibiting the sending of certain stanzas based on payload, type, or other appropriate features.

10. Restricting access to services (such as multi-user chat and publish-subscribe) that enable traffic amplification.

11. More strictly limiting the proposed restrictions depending on connection type, authentication type, or user class.

4 Recommendations

The following recommendations are presented roughly in order of interaction (e.g., recommendations related to the number of TCP connections or connection attempts are presented before recommendations related to authentication, which are presented before recommendations related to XML stanza handling).

4.1 Simultaneous Connections

A server implementation SHOULD enable a server administrator to limit the number of connections that it will accept from a given IP address at any one time. [8]
If an entity attempts to connect but the maximum number of connections has been reached, the receiving server MUST NOT allow the new connection to proceed. There are no XMPP errors associated with this behavior, since it occurs at the binding (TCP or HTTP) level before an XML stream is initiated.

[8] Alternatively, it is possible to limit the number of connections at the TCP layer rather than at the XMPP application layer. Care must be taken in doing so, since limits at the TCP layer might result in an inability to access non-XMPP services.

4.2 Connection Attempts

A server implementation SHOULD enable a server administrator to limit the number of connection attempts that it will accept from a given IP address in a given time period (e.g., one hour). [9]

If an entity attempts to connect but the maximum number of connection attempts has been reached, the receiving server MUST NOT allow the new connection to proceed. There are no XMPP errors associated with this behavior, since it occurs at the binding (TCP or HTTP) level before an XML stream is initiated.

4.3 Unauthenticated Connections

In accordance with RFC 3920, a server MUST NOT process XML stanzas (i.e., <message/>, <presence/>, or <iq/>) from clients that have not yet provided appropriate authentication credentials, and MUST NOT process XML stanzas from peer servers whose identity it has not either authenticated via SASL (see RFC 4422 [10]) or verified via server dialback.

4.4 Simultaneous Resources

A server implementation SHOULD enable a server administrator to limit the number of resources it will allow an account to bind at any one time.

If a connected client attempts to bind a resource but has already reached the configured number of allowable resources, the receiving server MUST return a <resource-constraint/> stanza error, where the XMPP <error/> element SHOULD also include an application-specific error condition of <resource-limit-exceeded/>, as shown in the following example:

Listing 1: Resource number limit violation

```
<iq type='error' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
    <resource>someresource</resource>
  </bind>
  <error type='cancel'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <resource-limit-exceeded xmlns='urn:xmpperrors'/>
  </error>
</iq>
```

[9] Alternatively, it is possible to limit the number of connections at the TCP layer rather than at the XMPP application layer. Care must be taken in doing so, since limits at the TCP layer might result in an inability to access non-XMPP services.

[10] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.

4.5 Stanza Size

A server implementation SHOULD enable a server administrator to limit the size of stanzas it will accept from a connected client or peer server.

If a connected client or peer server sends a stanza that violates the upper limit, the receiving server SHOULD NOT process the stanza and instead SHOULD return a <not-allowed/> stanza error, where the XMPP <error/> element SHOULD also include an application-specific error condition of <stanza-too-big/>, as shown in the following example:

Listing 2: Stanza size limit violation: stanza error

```
<message from='shakespeare.lit' to='iago@shakespare.lit/evilos'>
  <error type='modify'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <stanza-too-big xmlns='urn:xmpp:errors'/>
  </error>
</message>
```

Note: In the case of a stanza size limit, the server SHOULD NOT include the original payload in the error stanza.

Alternatively (e.g., if the sender has sent a seriously large stanza), the server MAY instead return a <policy-violation/> stream error:

Listing 3: Stanza size limit violation: stream error

```
<stream:error>
  <policy-violation xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  <stanza-too-big xmlns='urn:xmpp:errors'/>
</stream:error>
</stream:stream>
```

4.6 Multiple Recipients

A server implementation SHOULD enable a server administrator to limit the number of XML stanzas that a connected client may send to distinct recipients within a given time period.

If a connected client sends too many stanzas to distinct recipients in a given time period, the receiving server SHOULD NOT process the stanza and instead SHOULD return an <unexpected-request/> stanza error, where the XMPP <error/> element SHOULD also include an application-specific error condition of <too-many-stanzas/>, as shown in the following example:

Listing 4: Stanza recipients violation: stanza error

```
<message from='iago@shakespeare.lit/evilos' to='juliet@capulet.lit'>
  <error type='wait'>
    <unexpected-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <too-many-stanzas xmlns='urn:xmpp:errors'/>
  </error>
</message>
```

4.7 Bandwidth Limits

A server implementation SHOULD enable a server administrator to limit the amount of bandwidth it will allow a connected client or peer server to use in a given time period.
Bandwidth limits in existing XMPP server implementations (often called ”karma”) can be somewhat complex, since they dynamically respond to usage patterns, take into account temporary traffic bursts, enable the server administrator to modify recovery times and penalty lengths, etc. An example of low average bandwidth limits would be 1k-2k per second, of medium limits 4k-6k per second, of high limits 8k-10k per second. For details, see documentation for existing implementations.

4.8 Stanza Limits

A server implementation SHOULD enable a server administrator to limit the types of stanzas (based on payload etc.) it will allow a connected client send over an active connection. Possible restrictions include:

- Forbidding XMPP protocol extensions that typically involve the sending of large stanzas, such as IQ-Based Avatars [11] and In-Band Bytestreams [12]
- Prohibiting In-Band Registration [13] when directed to the server itself (i.e., disallowing open registration of new accounts).
- Limiting the number of roster requests that a user may request in a given time period.
- Not supporting offline message storage (or limiting the size of offline storage).
- Limiting the number of presence subscription requests that a user may send in a given time period.
- Limiting the number of stanzas that a user may generate in a given time period that are intended for unknown or non-existent JIDs, that are intended for JIDs at remote domains, or that are intended for distinct JIDs.

[11] XEP-0008: IQ-Based Avatars <http://xmpp.org/extensions/xep-0008.html>.
[12] XEP-0047: In-Band Bytestreams <http://xmpp.org/extensions/xep-0047.html>.
[13] XEP-0077: In-Band Registration <http://xmpp.org/extensions/xep-0077.html>.

4.9 Service Restrictions

An implementation of a service that enables traffic amplification (e.g., multi-user chat or publish-subscribe) SHOULD enable an administrator of that service to limit (based on JabberID or other characteristics) what entities may send information through the service. (See XEP-0045 regarding methods of banning users from multi-user chat rooms and XEP-0060 regarding methods for prohibiting users from interacting with publish-subscribe nodes.)
In fact, the "session manager" of an XMPP presence server also acts as just such an amplifier, since presence information is broadcast to all subscribers. Any such service SHOULD enable an administrator to limit the number of stanzas sent through the service in any given period of time (presence changes in a session manager, presence changes or messages in a chatroom, published items in a pubsub service).

5 Implementation Considerations

Implementations MAY enable administrators to configure appropriate exceptions to some of the recommendations specified in this document. Examples include:

- Less strict limits for server administrators compared to entities associated with registered accounts, and for entities associated with registered accounts compared to anonymous entities.
- Less strict limits for entities that authenticate via strong authentication methods (e.g., TLS + SASL EXTERNAL) compared to entities that authenticate via weaker authentication methods (e.g., TLS + SASL PLAIN or server dialback).
- Less strict limits for connections from known IP addresses.
- Less strict limits for connections made via the TCP binding compared to connections made via the HTTP binding.

6 Security Considerations

This entire document is about security.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [14].

[14] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

8 XMPP Registrar Considerations

The XMPP Registrar [15] includes <resource-limit-exceeded/> and <too-many-stanzas/> in its registry of application-specific error conditions (see <http://xmpp.org/registrar/errors.html>), where the element is qualified by the 'urn:xmpp:errors' namespace as described in Application-Specific Error Conditions [16].
The registry submission is as follows:

```
<condition>
  <ns>urn:xmpp:errors</ns>
  <element>resource-limit-exceeded</element>
  <desc>
    The account is not allowed to bind more resources at this time.
  </desc>
  <doc>XEP-0205</doc>
</condition>
<condition>
  <ns>urn:xmpp:errors</ns>
  <element>too-many-stanzas</element>
  <desc>
    A connected client has attempted to send multiple stanzas to
    too many different intended recipients in a given time period.
  </desc>
  <doc>XEP-0205</doc>
</condition>
```

9 Acknowledgements

Special thanks to Chris Mullins for calling attention to the need for a specification on this topic. Thanks also to Thiago Camargo, Bruce Campbell, Dave Cridland, Gustavo Felisberto, Justin Karneges, Alexey Melnikov, Pedro Melo, Kevin Smith, Michal Vaner, and Matthew Wild for their suggestions.

[15] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <http://xmpp.org/registrar/>.

[16] XEP-0182: Application-Specific Error Conditions <http://xmpp.org/extensions/xep-0182.html>.