

上网行为管理 协议分析系列

---- MSN 协议分析

相关主题: 上网行为管理,PatrolFlow,信息安全网关,带宽管理,流量控制,P2P 控制,BT 下载,邮件监控,IM 控制,游戏监管,聊天监控,内容审计,多链路负载均衡,Web 推送,防火墙,防毒墙

正文内容:

一、概要介绍

msn messenger 通常使用端口 1863 进行通信(在实际中用 sniff 跟踪发现 msn 通信都是用 1863 端口进行通信的)。在 msn messenger 工作中,本机客户端与三种服务器通过协议进行通信和数据交换。(dispatch 服务器、notification 服务器 tchboard 服务器)。在本机客户端和各服务器之间主要通过两种形式的进行通信,一种是命令,另一种是消息。

dispatch 服务器主要用于初始化连接服务器。用户首先利用地址 messenger.hotmail.com 和端口 1863 连接 dispatch 服务器,然后根据再返回的 ip 地址和端口来连接 notification 服务器。notification 服务器是 msn messenger 的主工作区,几乎所有的操作都要与 notification 服务器进行连接和信息交换,包括用户的状态改变、聊天请求以及来信通知。实际在工作中 notification 服务器的 ip 地址和端口固定为 64.4.13.195:1863。command: 多数数据是以标准的命令格式发送的。标准的命令格式主要由三部分组成,以命令标识符开始,然后是参数,以换行为结束。参数之间以空格区分。

message: 是一种独特的命令方式。它以 msg 开头,每个消息的第一行的末尾以一个数字来表示消以下部分息的字节数(包括 mime 头和主体部分)。第二行为 mime 的头,一般形式为 mime-version: 1.0,以换行结束。下一行所代表的是要发送消息的类型,定义的格式为 content-type: */*; charset=utf-8,其中 */*代表消息类型,charset=utf-8 是完全可选的,与是否使用该参数与定义的消息类型有关。随后 mime 头以两个换行结束,用于区分消息主体。transaction id: 在客户端向服务器端发送的每个命令和消息中都包含一个 transaction id。其位于命令标识符和 msg 后面,服务器端收到客户端的相应命令和消息后,回应客户端时把对应的 transaction id 返回给客户端,客户端根据 transaction id 来判断服务器回应的是哪个请求。每次客户端向服务器发送一次命令或消息后,transaction id 自动加 1。

二、启动 msn

初始化连接

(1) 连接的第一步是连接 dispatch 服务器。打开一个 tcp socket,通过 ip 地址 messenger.hotmail.com 和 1863 端口连接 dispatch 服务器。

(2) 当连接 dispatch 服务器成功后,本机客户端向 dispatch 服务器发送一个 ver 命令,以 msnp7 msnp6 msnp5 msnp4 cvr0 (协议版本)作为参数。服务器收到请求后,同样返回 ver 命令,如果参数为 0,表示协商失败。

(3) 本机客户端收到回复后,向服务器发送一个无参数的 inf 请求,请求一个认证算法。当 dispatch 服务器收到后返回 md5。

(4) 本机客户端根据返回的参数向服务器发送 usr 命令,其中传入两个参数,第一个为服务器返回的 md5,第二个参数为客户申请登陆的电子邮件地址。服务器根据收到的请求返回一个 xfr 命令,把 notification 服务器的 ip 地址和端口作为参数返回给客户端。

(5) 最后客户端根据 ip 地址和端口去连接 notification 服务器。以下是客户端和 dispatch 服务器的实际通信过程。

connect: messenger.hotmail.com 1863

```
>>> ver 0 msnp7 msnp6 msnp5 msnp4 cvr0
<<< ver 0 msnp7 msnp6 msnp5 msnp4 cvr0
>>> inf 1
<<< inf 1 md5
>>> usr 2 md5 i [email]example@passport.com[/email]
<<< xfr 2 ns 64.4.12.132:1863 0
disconnect
```

连接 notification 服务器

连接 notification 服务器成功后，前三步执行的操作与 dispatch 服务器相同。

(6) 同样本机客户端根据返回的参数向 notification 服务器发送 `usr` 命令，其中传入两个参数，第一个为 notification 服务器返回的 md5，第二个参数为客户登陆的电子邮件地址。服务器收到请求后返回 `usr`，包含两个参数 `md5 s #.#`。 `#.#` 为 md5 hash。

(7) 客户端根据收到的答复后再次发送一个 `usr`，参数为 `md5 s #`。其中 `#` 为上次服务器返回的 md5 hash 的小写 16 进制。服务器收到后返回 `usr`，参数为 `ok user@host name 1`。`user@host` 为用户登陆的电子邮件地址，`name` 为用户的映射名。

(8) 当用户成功登陆 `msn` 后，客户端向服务器发送一个 `chg` 请求，请求服务器修改新登陆用户的状态，参数为 `nln`。参数 `nln` 表示在线，这是每个用户登陆后需要做的第一步初始化用户状态。以下为 notification 服务器与客户端之间的通信过程。

```
connect: 64.4.12.132 1863
>>> ver 3 msnp7 msnp6 msnp5 msnp4 cvr0
<<< ver 3 msnp7 msnp6 msnp5 msnp4 cvr0
>>> inf 4
<<< inf 4 md5
>>> usr 5 md5 i [email]example@passport.com[/email]
<<< usr 5 md5 s 1013928519.693957190
>>> usr 6 md5 s 23e54a439a6a17d15025f4c6cbd0f6b5
<<< usr 6 ok [email]example@passport.com[/email] my%20screen%20name 1
>>> chg 7 nln
<<< chg 7 nln
continue session . . .
```

三、语音对话

通信过程

(1) 本机客户端向 notification 服务器 (64.4.13.195:1863) 请求两个新 `tchboard` 服务器地址，其中一个用于发送请求，另一个用于接受回复。如下面表格所示：

```
secondary connection port range protocol type direction info
6891-6900 tcp inbound sending
6891-6900 tcp outbound receiving
```

客户端通过 `tcp` 向服务器发送 `xfr` 命令，参数为 `sb`。notification 服务器收到请求后，同样以 `xfr` 命令返回，参数中包 `tchboard` 服务器的 ip 地址和端口以及 `tchboard` 服务器的登陆序号。如下例所示：

```
>>> xfr 10 sb
<<< xfr 10 sb 64.4.12.193:1863 cki 16925950.1016955577.17693
```

(2) 客户端根据返回的 ip 地址和端口好，通过 `tcp` 协议连接两 `tchboard` 服务器。

(3) 连接成功后，客户端分别向两 `tchboard` 服务器发送 `usr` 命令，第一个参数为实际连接的

电子邮件地址, 第二个参数为第二步返回 tchboard 服务器登陆序号。如果发送成功, tchboard 服务器返回 usr 命令, 第一个参数为 ok。如下例所示:

```
>>> usr 1 [email]example@passport.com[/email] 16925950.1016955577.17693
<<< usr 1 ok [email]example@passport.com[/email] mike
```

(4) 当邀请对方对话时, 把对方的注册的电子邮件地址作为参数传递到 cal 命令中, 然后发送 tchboard 服务器, 收到该请求后, 服务器返回一个 id 号。同时服务器又向客户端发送 joi 命令, 把被请求方的电子邮件地址作为第一个参数。如下例所示:

```
>>> cal 2 [email]name_123@hotmail.com[/email]
<<< cal 2 ringing 11752099
<<< joi [email]name_123@hotmail.com[/email] name_123
```

以上步骤都是本机客户端与两 tchboard 服务器之间同时进行, 都执行了以上所有的步骤。

(5) 当客户要进行语音对话时, 本机客户端向发送请求 tchboard 服务器发送一个请求(假设 tchboard 服务器的 ip 地址: 64.4.12.192, 端口: 1863)。该请求的格式如下例所示:

```
msg 4 n 277
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
application-name: 请求服务类型
application-guid: {5d3e02ab-6190-11d3-bbbb-00c04f795683}
session-protocol:sm1
context-data:
requested:sip_a; capabilities: sip_a,sip_v
invitation-command: invite
```

.....

(6) 服务器 64.4.12.192 收到该请求后, 回复 ack, 表示确定已收到该请求。

(7) 当被请求方接受你的语音对话邀请后, 回复 tchboard 服务器(假设为 ip 地址: 64.4.12.159, 端口: 1863) 向本机客户端发送回复, 其回复格式如下:

```
msg [email]name_123@hotmail.com[/email] name_123
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
invitation-command: accept
```

.....

ip-address:返回被请求方的 ip 地址

(8) 当本机收到该回复后, 向服务器 64.4.12.159 发送一个消息, 其格式如下:

```
msg 4 a 237
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
invitation-command: accept
```

.....

ip-address:本机的 ip 地址和端口

(9) 服务器 64.4.12.159 收到该消息后, 返回一个 ack 命令, 表示确认已收到。

(10) 然后由被请求方根据本机客户端的 ip 地址和端口进行 udp 连接, 进行数据传递。

四、发送文件

通信过程

(1) 文件操作的通信过程与语音对话前四步相同, 但发送文件操作只申请一 tchboard 服务器(假设 ip 地址: 64.4.12.164, 端口: 1863)。

(2) 当客户要进行文件发送时, 本机客户端向服务器 64.4.12.164 发送一个请求, 其格式如下:

```
msg 4 n 277
```

```
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
application-name: file transfer
application-guid: {5d3e02ab-6190-11d3-bbbb-00c04f795683}
invitation-command: invite
invitation-cookie: 33267
application-file: readme.txt
application-filesize: 60904
```

(3) 服务器 64.4.12.164 收到该请求后, 等被请求方用户接受该请求后返回给本机客户端一个答复, 其格式如下:

```
msg [email]example@passport.com[/email] tim 179
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
invitation-command: accept
invitation-cookie: 33267
launch-application: false
request-data: ip-address:
```

注意: 其中最后一行包含了对本机 ip 地址和端口号的请求。

(4) 本机收到该回复后, 立刻向服务器 64.4.12.164 发送一个消息, 其格式如下:

```
msg 4 n 238
mime-version: 1.0
content-type: text/x-msmsgsinvite; charset=utf-8
invitation-command: accept
invitation-cookie: 33267
ip-address: 10.44.102.65
port: 6891
authcookie: 93301
launch-application: false
request-data: ip-address:
```

注意: 该消息中包含了本机的 ip 地址和端口。

(5) 然后由被请求方根据本机客户端的 ip 地址和端口进行 tcp 连接, 进行文件发送。

五、视频对话

通信过程

视频对话的通信过程与语音对话相同, 同样申请两 tchboard 服务器, 最后通过 udp 进行连接, 进行数据传送。

六、发送即时消息

发即时消息

发送即时消息的通信过程与语音对话前四步相同, 但只申请一 tchboard 服务器 (假设 ip 地址: 64.4.12.174, 端口: 1863)。完成上述四步操作, 完成对服务器 64.4.12.174 的连接后, 如果本机客户端要向另一个客户端发送即时消息, 就会向服务器 64.4.12.174 发送一个消息, 消息中附带要发送的内容, 其格式如下:

```
msg 3 a 157
mime-version: 1.0
content-type: text/plain; charset=utf-8
x-mms-im-format: fn=microsoft%20sans%20serif; ef=i; co=000000; cs=0; pf=22
```

```
hello! how are you?
```

注: 最后一行为发送的内容。

收即时消息

如果有一个客户端向你本机发送即时消息，则通过服务器 64.4.12.174，即连接本机客户端 tchboard 服务器发送一个消息，其消息格式如下：

```
msg [email]example@passport.com[/email] mike 157
mime-version: 1.0
content-type: text/plain; charset=utf-8
x-mms-im-format: fn=microsoft%20sans%20serif; ef=i; co=000000; cs=0; pf=22
```

hello! how are you?

注：第一行 msg 命令后的参数为对方客户端的电子邮件地址，最后一行为发送内容。

七、启动白板

通信原理

(1) 其通信原理与语音对话通信原理前几步骤基本一致，但在第 8 步的时候有所不同，本机向服务器 (64.4.12.160: 1863) 发送的消息中并不包括本机的端口号，只返回了一个 ip 地址，其结构所示如下：

```
msg 4 a 237
mime-version: 1.0
content-type: text/x-mmsmsgsinvite; charset=utf-8
invitation-command: accept
```

.....

ip-address:本机的 ip 地址

(2) 随后服务器 64.4.12.160 回复 ack。

(3) 本机客户端向 notification 服务器 (64.4.13.195:1863) 发送一个 qry 请求，确定是否保持和服务器 64.4.13.195 的连接。

(4) 服务器 64.4.13.195 返回一个无参数的命令 qry，表示确定。

(5) 然后由服务器 64.4.12.160 发送一个消息，其结构如下所示：

```
msg [email]name_123@hotmail.com[/email] name_123
mime-version: 1.0
content-type: text/x-mmsmsgsinvite; charset=utf-8
invitation-command: context
context-data:10.1.1.211:13374
```

.....

注：在 context-data 选项中包含了被请求方的 ip 地址和端口号

(6) 本机向被请求方进行 tcp 连接，随后进行数据交换。

八、启用应用程序共享

其通信过程与启用白板相同。

九、发现问题

由于通过上网查找的有关 msn 文档和资料相对有点早，与现在所用的版本有差别，所以在一些细节方面还是有所不同的，比如在实践中用 sniff 跟踪数据包发现先行版本的 msn 在启动的时候并没有进行初始化连接 dispatch 服务器，而是直接与 notification 服务器进行连接。notification 服务器的 ip 地址和端口也不是由 dispatch 服务器返回的，而是已经实际指定了

(64.4.13.195:1863)。

十、解决方法

获取 ip 地址和端口

(1) 首先根据 notification 服务器 (64.4.13.195:1863)，本机客户端与服务器 64.4.13.195:1863 之间会建立一条连接。

(2) 用户新开一个窗口进行操作时，本机客户端会向服务器 64.4.13.195:1863 请 tchboard 服务器的 ip 地址和端口，然后根据返回 tchboard 服务器的 ip 地址和端口建立一条连接。我们可以根据服务器 64.4.13.195:1863 返回的数据包获取请求 tchboard 服务器的 ip 地址和端口。服

务器以命令形式返回，其格式如下：xfr 10 sb 64.4.12.193:1863 cki 16925950.1016955577.17693。我们可以根据命令行关键字 xfr 来截取这个包，然后解 tchboard 服务器的 ip 地址和端口。

(3) tchboard 服务器与客户端连接成功后，在操作之前本机客户端会 tchboard 服务器发送一个消息，邀请另一方加入。如果对方接受你的邀请，则会 tchboard 服务器向你发送一个消息，接受你的邀请。在消息中会有一个关键字 ip-address，其可能会包含着对方的 ip 地址和端口或只有 ip 地址。如果包含端口号则用“:”区分。如果在选项 ip-address 中没有包含端口，则需查找另一个关键字 port，其可能存在端口号（发送文件返回的消息中存在这个关键字，其余的操作中好象没有发现）。

(4) 当本机客户端收 tchboard 服务器的回复后，再次回复一个消息。其中同样可能包含着本机的 ip 地址和端口，因此需要向上述步骤 3 一样进行分析，获取本机的 ip 地址和端口。启用白板功能例外，本机客户端返回的 ip 地址和端口存放在消息的 context-data 选项中。最后由本机和被邀请方之间建立一条连接。在 msn 通信过程中所有的消息都是由 msg 开头的。截获数据包后先分析第一行是否以 msg 开头，然后再从消息中查找关键字 ip-address、port 和 context-data，从中获取要建立连接的 ip 地址和端口。

区分建立的连接是 tcp 还是 udp

在 msn 中只有语音对话和视频对话的连接是 udp 的。首先客户端会向服务器发送一个操作请求数据包，包中存在一项数据指明了请求的操作类型，其关键字以 application-name 开头，随后为相关数据以 ascii 码表示。语音对话和视频对话 ascii 码相同，都为 3a 20 e8 af ad e9 9f b3 e5 af b9 e8 af 9d 0d 0a，其中 3a 20 为冒号和空格，0d 0a 为回车和换行。只要截取这段数据就能获取请求的操作类型。application-name 只存在一次，只有第一次发请求操作时包含这项，以后就这项数据就不存在了。服务器返回 ip 地址和端口时为了便于区分到底是哪个操作请求需要建立的连接，因此在数据包中还存在一项为 invitation-cookie，其内容为一串随机生成的数字。在服务器返回的数据包中都把这项数据返回给客户端，因此客户端可以根据返回的 id 区分具体是哪个操作请求的 ip 地址和端口。根据 application-name 和 invitation-cookie，我们就可以区分随后建立的连接是哪个操作请求的以及建立的连接是 tcp 还是 udp。

注：此外其它操作类型对应的 ascii 码分别为：

文件传输：3a 20 e6 96 87 e4 bc a0 e8 be 93 0d 0a

启用白板：3a 20 e7 99 bd e6 9d bf 0d 0a

寻求远程协助：3a 20 e8 bf 9c e7 a8 8b e5 8d 8f e5 8a a9 0d 0a

启用应用程序共享：3a 20 e5 ba 94 e7 94 a8 e7 a8 8b e5 ba 8f e5 85 b1 e4 ba ab 0d 0a

此外发送消息没有该项数据内容。

更多信息请登录 <http://www.byzoro.com>