Postfix attachement filtering & PIKT

Keeping unwanted attachements out of your user's reach

What ?

Once upon a time, when we didn't have virus scanning installed on our MTA, we decided to filter out Win32 executable attachments, so clueless users would be somehow protected again the villains out there, and against other clueless users. We also didn't want our staff or users t spread viruses to the rest of the world, destroying our sysadmin reputation.
As good BOFHs, we imposed a "no executables in mails" policy on our users, for inbound and outbound mail through our MTA. After all, very few people really need to send out executable attachments in mail, and if they really need to, they can still zip it.

How ?

As PIKT diehards, we of course wanted to use it for this task. We also wanted PIKT to report mail abuse attemps (executable attachements mostly, but also relaying attempts and other usu errors reported by the MTA). So we used PIKT to create the Postfix 'body_checks' file, which is parsed by postfix at startup. When postfix receives a mail (from MUA or MTA), it tries to match every single line of mail against the regular expressions in body_checks.

Preliminary setup

In this document, we consider that postfix is set up to use body_checks (check the postfix doc) We also assume that postfix servers are members of the 'postfix_servers' pikt group.

Defining unwanted extensions in attachments

Unwanted extensions are listed in an object file, RejectExtensions. All mail attached files havi this extension will be rejected by postfix. You can customize this at your convenience, and accroding to you local policy, to add for instance .doc, .xls, etc... to protect against macro-virus.

```
#if postfix_servers

RejectExtensions
                    au
                    bat
                    chm
                    cla
                    cmd
                    com
                    css
                    dll
                    dot
                    exe
                    hlp
                    hta
                    jse
                    lnk
                    ocx
                    pak
                    pif
                    pps
                    scr
                    sct
                    shs
                    src
                    vbe
                    vbs
```

```
                    vxd
                    wsh

#endif //postfix_server
```

RejectExtensions.cfg

Generating /etc/postfix/body_checks from RejectExtensions

We then needed an alarm, run withing an alert run, to create the body_checks file. Note that i the alarm below, the body_checks file is hardcoded. change it at will, or customize with a macro.

```
#if postfix_servers

GenBodyChecks
   init
      status active
      level urgent
      task "Generates postfix body_checks rules"
      input file "=objdir/RejectExtensions.obj"

      dat $ext 1

   begin
      if #fopen(BC, "/etc/postfix/body_checks", "w") == #err()
         output mail "Unable to open /etc/postfix/body_checks"
         die "oops!"
      endif

      do #write(BC, "\#Pass-all rule")
      // this rule protects agains PIKT Mail warning rejects
      do #write(BC, "/=bypass_tag/ OK")

   rule
      do #write(BC, "/^begin(-base64)? [0-9]+.*(\\.|\=2E)".
                $inline."(\\?\=)?(\\.)?/ REJECT")
      do #write(BC, "/^[^<]*(body|filename|name\=).*(\\.|\=2E)".
                $inline."(\\?\=)?(\\.)?/ REJECT")

   end

   do #fclose(BC)

#endif // postfix_servers
```

GenBodyChecks

This alarm will take each line of RejectExtensions object, and generate two lines body_checks. The first line handled uuencoded (eventually base64 ones) inlined attachments, and the secon line the more frequent mime type attachments.

```
/^begin(-base64)? [0-9]+.*(\.|=2E)exe(\?=)?(\.)?/ REJECT
```

UUEncoded attachments match regexp (dot exe files)

```
/^[^<]*(body|filename|name=).*(\.|=2E)exe(\?=)?(\.)?/ REJECT
```

MIME attachments match regexp (dot exe files)

The alarm also add one special line at the top of body_check. This line makes the matching process jump out and accept the current mail message line (the 'OK' at the end of the rule instead of the usual 'REJECT'). This is a pure hack done because PIKT reporting mails (described below) should match those body_checks regexps. Since PIKT is reporting executable attachement attemps and giving a lot of details, it's reporting messages were matched by postfix and bounced back. Rather painful, and loop prone. To circumvent this just define a 'bypass_tag' macro in your macro.cfg file. PIKT will thus generate an 'OK' match against that macro and send postfix attachment denial reports with that tag on the line. You sh make that tag rather secure, since the tag knowledge would allow an attacker to craft mail messages in such ways that postfix wouldn't match them. It's appearance in mail messages sh also be very unlikely, especially in mail messages with attachments you want to filter ! So avoi 'exe' for instance. Also, that tag shouldn't be titanesque, since it will appear at the front of eve line reporting an invalid attachment attempt.
The value used here is just for the purpose.

```
/PASS/ OK
```

Pass all rule (choose carefully)

At that stage, postfix has all it needs to reject 'bad' mail.
Now on to reporting stuff.

Reporting mail abnormal events

The alarm below will check the postfix 'maillog' for abnormal events. Actually, four events are checked for : forbidden attachment attempt, header_check match, relaying attempt and unknc recipients. There is also a fifth 'unclassified error' category, which encompass errors we couldn't categorize above. Note the macros defined below.

```
PostfixRejects
  init
    status active
    level urgent
    task "Check if postfix rejected some mail"
    input log "=maillog"

  begin
    // to ensure input source
    exec wait "=touch =maillog"

    set #rej_count["Attachements"] = 0
    set #rej_count["Recipient"] = 0
    set #rej_count["Relay"] = 0
    set #rej_count["Header"] = 0
    set #rej_count["Unclassified"] = 0
    set $rej["Attachements"] = ""
    set $rej["Recipient"] = ""
    set $rej["Relay"] = ""
    set $rej["Header"] = ""
    set $rej["Unclassified"] = ""

  rule
```

```
  /// Check for attachements
  =postfix_checkfor("Attachements",
                ".*reject: body .*name=(.*); from=<(.*)> to=<(.*)>",
                "=bypass_tag $2 tried to send a suspicious attachement ($1) to $3")
  /// Check for unknown recipients
  =postfix_checkfor("Recipient",
                ".*reject: RCPT from (.*): 550 <(.*)>: User unknown; from=<(.*)> to=<(.*)>",
                "=bypass_tag $3 tried to mail non-existent $4/$2 from $1")
  /// Check for relay attempts
  =postfix_checkfor("Relay",
                ".*reject: RCPT from (.*): 554 <(.*)>:.*from=<(.*)> to=<(.*)>",
                "=bypass_tag $3 tried to relay thru =pikthostname to $4/$2 from $1")
  /// Check for mail rejected by header rules (/etc/postfix/header_checks
  =postfix_checkfor("Header",
                ".*: reject: header (.*)",
                "=bypass_tag header_checks on =pikthostname rejected this mail: $1")
  /// check for mail rejectected for some other reason
  =postfix_checkfor("Unclassified",
                ".*: reject: (.*)",
                "=bypass_tag =pikthostname rejected the following unclassified mail: $1")

 end

 // We report problems
 =postfix_reportfor("Attachements",
                "/!\\".$fixed(#rej_count["Attachements"]).
                "Suspicious attachements rejected")
 =postfix_reportfor("Recipient",
                "/@\\ ".$fixed(#rej_count["Recipient"]).
                "Non-existent recipient in enveloppes")
 =postfix_reportfor("Relay",
                "/!\\ ".$fixed(#rej_count["Relay"]).
                "Relay attempts")
 =postfix_reportfor("Header",
                "/!\\ ".$fixed(#rej_count["Header"]).
                "Rejected by headers")
 =postfix_reportfor("Unclassified",
                "/?\\".$fixed(#rej_count["Unclassified"]).
                "Unclassified problems")

#endif // postfix_servers
```

PostfixRejects : reports postfix rejected mail

```
#if postfix_servers


bypass_tag PASS

postfix_checkfor(T, R, M) // (T) is type,(R) is regexp,(M) is message
  if $inlin =~ (R)
    set $rej[(T)] .= (M).$newline()
    =incr(#rej_count[(T)])
    next
  endif

postfix_reportfor(T, M) // (T) is type, (M) is message
  if $rej[(T)] ne ""
    output mail (M).$newline()
    output mail $rej[(T)]
  endif
```

Scheduling the stuff

Here are the two alerts used to schedule body_checks generation and maillog file scruting. Note that GenerateBodyChecks has '=piktnever' timing, because it is intended to be run manually.

```
#if postfix_servers


   GenerateBodyChecks
      timing    =piktnever
      alarms    GenBodyChecks


#endif //postfix_servers
```

GenerateBodyChecks : executing GenBodyChecks alarm

```
#if postfix_servers
  MailAbuse
    timing 20% * * * *
    mailcmd "=mailx -s 'PIKT Alert: MailAbuse on =pikthostname (PIKT)'
            mblanc@erasme.org dlacroix@erasme.org"

    alarms
       PostfixRejects

#endif
```

MailAbuse : executing PostfixRejects alarm

That's it. Those scripts should solve most of your unwanted attachments problems. They can b improved a lot, and if you do so, please send us what you've done (mblanc at erasme dot org).

What you get

Here is the kind of mail you can expect from pikt parsing the log file.

```
Subject: /!\ MailAbuse on smtp.site.org (PIKT)
Date: Fri, 8 Feb 2002 15:01:08 +0100 (CET)
From: root@smtp.site.org (root)
To: sysadmin@site.org


                                PIKT ALERT
                            Fri Feb 8 15:01:08 2002
                                 mail1

          URGENT:
            PostfixRejects
              Check if postfix rejected some mail

          /!\ Suspicious attachements rejected

                    PASS badguy@somewhere.com tried to
                    send a suspicious attachement(test.exe)
                    to sysadmin@site.org
```

/@\ Non-existent recipient in envelopes

001d99c@kylmedia.fi tried to mail non-existent werner@site.org/werner@site.org from unknown[192.192.192.192]
usager32@site.org tried to mail non-existent usager@site.org/usager@site.org from poste32.al.lan[192.168.101.32]

/!\ Rejected by headers

header_checks on smtp.site.org rejected this mail: header Received: from somemta.there.fr (somemta.there.fr [192.192.192.192])??by smtp.site.org (Postfix) with ESMTP id 6A8B71700D0??for ; Fri, 8 Feb 2002 14:16:04 +0100 (CET); from= to=

PIKT report about rejected mail

Caveats

These scripts may not encompass all your needs, nor even fit any of them. Read them carefully before using them, especially the body_checks regexps part. Those regexps might (read : will) match and bounce legitimate mails, or let in supposed-to-be bounced ones.
It happened here; those regexps are the result of a trial-error loop process. That process is certainly not complete. Those scripts are not meant to replace a good anti-virus product, but since viruses are here to make a living for AV vendors, you might not feel like buying one. (no, didn't say AV vendors are themselves creating the viruses they fight :).

Notes

In the code above, lines may be broken for readability and convenience.

References

- **PIKT**, Robert Osterlund, http://pikt.org
- **MIME (Multipurpose Internet Mail Extensions) Part One**, Borenstein, Freed, http://www.faqs.org/rfcs/rfc1521.html
- **MIME (Multipurpose Internet Mail Extensions) Part Two**, Moore, http://www.faqs.org/rfcs/rfc1522.html
- **Postfix**, Wietse Venema, http://www.postfix.org

Michel Blanc <mblanc at erasme dot org>
$Revision: 1.4 $ - $Date: 2002/02/19 15:28:24 $