研究論文

パスワードクラッキングに対抗するためのセキュリティ対策

山守　一徳
三重大学 教育学部

サーバーを運営していると、ログインを試してきていると思われるログを見ると落ち着いておられない。多くのユーザーを抱えているサーバーは特に、各ユーザーのパスワードが強固な文字列になっているのか不安に駆られる。実際に、海外からのパスワードクラッキング攻撃が続き、パスワードを破られたと思われる事象も発生したことがある。そこで、パスワードに失敗した回数が上限を超えたら、その発信元 IP アドレスからの接続要求を遮断するという設定を施してみた。その方法を紹介すると伴に、対策後にどうなったかについて、パスワードクラッキングの現状を紹介する。

１．はじめに

対策は CentOS6.10 のサーバー上に施した。メールサーバー等のサービスが稼働しているため、POP3、IMAP の 110 番、143 番ポートを使うポートが開いていた。そのため、海外からの恰好の餌食となり、パスワードを辞書式攻撃で試すというパスワードクラッキング攻撃が行われていた。ファイアウォールの不正侵入検知機能に引っかからないように、ゆっくりとした速度で延々と試し続ける攻撃である。メールアドレスが、論文や発表物で公開されていて他から入手可能であるので、パスワードを試す時のアカウント名については、不一致失敗は少数である。他から入手したと思われる色々なアカウント名に対してパスワードが試され、パスワード不整合によるログイン失敗のログが多く残されている状況にあった。

その中で、１０００件以上のメールを一気に発信する事象が発生し、ファイアウォールの不正侵入検知機能で警告が挙がったため、そのメールの発信者のパスワードが破られたことがわかった。急ぎアカウントを停止し、パスワードを変更し、被害状況を調べてもらったが、パスワードの文字列が安易だったことがわかった。

多くのユーザーを抱えるサーバーであるため、全ユーザーに事象を説明し、パスワードを強固な物に置き換えるように依頼を掛けたが、全ユーザーが強固なパスワードになっているか確かめる術もない。そこで、ソフトウェア設定によって対策を施すことにした。用いたソフトは、swatch である。

２．swatch ツール

swatch は Linux で動く、リアルタイムにログを監視するツールである。正規表現で一致する文字列が出た時に、特定のコマンドを実行させることができる。今回は、authentication failure の文字列を監視し、IP アドレスごとにその回数をカウントし、上限を超えたら、遮断することにした。遮断させるツー

ルには、iptables を用いる。iptables は、サーバーごとのファイアウォールのようなソフトであり、遮断ルールを追加したり削除したりすることができる。CentOS6 では、iptables がほぼ標準でインストールされており、利用は簡単である。

２．１　swatch のインストール

CentOS6 で swatch は、yum install swatch というコマンドだけではインストールできず、EPEL リポジトリをインストールする必要がある。yum install epel-release のコマンドを叩いてから、yum install swatch を実行する。さらに、今回はファイル操作を行うため、yum install perl-File-Tail を実行する。さらに、IP アドレスの発信元情報を探るため、yum install jwhois も実行する。

２．２　swatch の設定

(1) /usr/local/bin/swatch-action.sh の作成

/usr/local/bin/swatch-action.sh のファイルを作成する。root ユーザーだけが実行できれば良いので、ファイルの所有者は root にし rwx------の権限を与える。ファイルの中身は図１の通りである。4 行目の mailaddr= root で、root ユーザーへメールが飛ぶ。5 行目の read LOG でログファイルから読み込むことになるが、図２の１行目の ＃logfile の右に書かれたファイルが読み込むログファイルになる。この swatch-action.sh は、図２の 4 行目の 14 という引数を伴って呼び出されるので、14 番目の項目を IPADDR の変数に最初代入することになる。14 番目の項目の先頭から IP アドレスが書かれている場合とそうでない場合を場合分けして IP アドレスを切り出している。主に海外からの認証失敗アクセスを遮断したかったため、自分のサーバーからのアクセスと 133 で始まる IP アドレスからの認証失敗の検出は除外した。

```
#!/bin/bash
PATH=/bin:/sbin:/usr/bin
LANG=C
mailaddr=root
read LOG
# ログから IP アドレスを抽出
IPADDR=`echo $LOG | cut -d " " -f $1`
echo "$IPADDR" | grep "^[0-9]*¥." > /dev/null 2>&1
if [ $? -eq 0 ]; then
      IPADDR=`echo "$IPADDR" | sed -e 's/¥([0-9]*¥.[0-
      9]*¥.[0-9]*¥.[0-9]*¥).*/¥1/p' -e d`
else
      IPADDR=`echo "$IPADDR" | sed -e 's/.*[^0-9]¥([0-
      9]*¥.[0-9]*¥.[0-9]*¥.[0-9]*¥).*/¥1/p' -e d`
fi
# IP アドレスが 127.0.0.1 の場合は終了
if [ "$IPADDR" = "127.0.0.1" ]; then
   exit
fi
# IP アドレスが 133 で始まる場合は終了
addr1=`echo $IPADDR | cut -d . -f 1`
if [ "$addr1" = "133" ]; then
   exit
fi
if [ -n "$IPADDR" ]; then
   echo $LOG >> /var/log/swatch/$IPADDR
   cnt=`cat /var/log/swatch/$IPADDR | wc -l`
else
  cnt=0
fi
if [ $cnt -ge 5 ] || [ $# -eq 2 -a    "$2" = "lock" ]; then
     iptables -I INPUT -s $IPADDR -j DROP
     echo "iptables -D INPUT -s $IPADDR -j DROP >
/dev/null 2>&1" | ¥
     at now+1hour > /dev/null 2>&1
     [ "$mailaddr" != "" ] && (cat /var/log/swatch/$IPADDR ; ¥
     echo ; whois $IPADDR) | ¥
     mail -s "$IPADDR $cnt lock!" $mailaddr
     echo "`date` $IPADDR $cnt lock!"
else
     echo "`date` $IPADDR $cnt"
fi
```

図１　swatch-action.sh

```
# logfile /var/log/secure
watchfor /dovecot:auth.*authentication failure/
  pipe "/usr/local/bin/swatch_action.sh 14"
  throttle=00:00:10
```

図２　secure.conf

```
/var/log/swatch/swatch.log {
  missingok
  notifempty
  sharedscripts
  postrotate
     /etc/rc.d/init.d/swatch restart > /dev/null || true
  endscript
}
```

図３　/etc/logrotate.d/swatch

/var/log/swatch/$IPADDR で示される IP アドレス毎のファイルがログに残り、”$cnt -ge 5”で 5 回以上失敗すると、iptables コマンドで遮断を行う。DROP の指定をしているので、攻撃者には返答を返すことなく要求を捨てている。”at now+1hour”によって、1 時間後には、iptables に書き込んだ遮断ルールも消されるようになっ

ている。これが無いと、iptables にルールが大量に残ることになり応答性能上問題となる。5 回以降の失敗の時に root へ飛ぶメールの中には、whois $IPADDR　で実行された結果もメール本文の中に現れるため、その IP アドレスがどこの国からの攻撃なのかを知ることができる。

なお、このソースは、参考文献に挙げたサイトに書かれているソースを参考にしているため、ほとんど同じであるが、$IPADDR の値が、抽出できない場合の対策が施してある。監視を行う dovecot のログは、”Aug 24 15:07:05 minerva auth: pam_unix(dovecot:auth): authentication failure; logname= uid=0 euid=0 tty=dovecot ruser=アカウント名 rhost=40.97.142.○○ user=アカウント名”の形で書かれ、14 番目の項目に、”rhost=40.97.142. ○○”という IP アドレスが表れてくるのが通常である。しかし、”rhost=::1”という形で現れる場合もあり、”rhost=::1”の記述では IP アドレスが抽出できない。その場合、$IPADDR が空になるため、echo $LOG >> /var/log/swatch/$IPADDR の実行文で、ディレクトリへ書き込みをしようとしてしまいエラーが発生する。そこで、条件分岐を入れエラー対処している。

(2) /etc/swatch/secure.conf の作成

/etc/swatch/のディレクトリを作成し、/etc/swatch/secure.conf のファイルを作成する。ファイルの中身は図 2 の通りである。1 行目の # logfile /var/log/secure は、この記述を図 4 の中の WATCHLOG=` grep "^# logfile" $conf |… で検出しているため、必須の行であり、/var/log/secure のログファイルが監視対象になることを示している。

```
#!/bin/bash
# chkconfig: 2345 90 35
# description: swatch start/stop script
# Source function library.
. /etc/rc.d/init.d/functions

PATH=/sbin:/usr/local/bin:/bin:/usr/bin
mkdir -p /var/log/swatch
start() {
 # Start daemons.
 ls /var/run/swatch_*.pid > /dev/null 2>&1
 if [ $? -ne 0 ]; then
 echo -n "Starting swatch"
 pno=0
 for conf in /etc/swatch/*.conf
 do
 pno=`expr $pno + 1`
 WATCHLOG=`grep "^# logfile" $conf | awk '{ print $3 }'`
 swatch --config-file $conf --tail-file $WATCHLOG ¥
 --script-dir=/tmp --awk-field-syntax --use-cpan-file-tail --daemon ¥
 --pid-file /var/run/swatch_$pno.pid ¥
 >> /var/log/swatch/swatch.log 2>&1
 RETVAL=$?
 [ $RETVAL != 0 ] && return $RETVAL
 done
 echo
 [ $RETVAL = 0 ] && touch /var/lock/subsys/swatch
 return $RETVAL
 else
  echo "swatch is already started"
 fi
}
stop() {
 # Stop daemons.
 ls /var/run/swatch_*.pid > /dev/null 2>&1
 if [ $? -eq 0 ]; then
 echo -n "Shutting down swatch"
 for pid in /var/run/swatch_*.pid
 do
 kill $(cat $pid)
 rm -f $pid
 done
 echo
 rm -f /var/lock/subsys/swatch /tmp/.swatch_script.*
 else
 echo "swatch is not running"
 fi
}
status() {
 ls /var/run/swatch_*.pid > /dev/null 2>&1
 if [ $? -eq 0 ]; then
 echo -n "swatch (pid"
 for pid in /var/run/swatch_*.pid
 do
 echo -n " `cat $pid`"
 done
 echo ") is running..."
 else
 echo "swatch is stopped"
 fi
}
```

図 4 (a)　/etc/rc.d/init.d/swatch（前半）

watchfor /dovecot:auth.*authentication failure/ によって、dovecot:auth.*authentication failure に

一致する文字列が見つかると、
/usr/local/bin/swatch_action.sh 14　へ文字列を送り込んでいるが、throttle=00:00:10　によって 10 秒は間隔を空けている。

(3) /etc/logrotate.d/swatch の作成

/etc/logrotate.d/swatch のファイルを作成する。ファイルの中身は図 3 の通りである。

２行目の missingok は、ログファイルがない場合でもエラー出さないことを指示し、notifempty は、ログファイルが空の場合は rotate しないことを指示し、sharedscripts は、postrotate または prerotate で指定したコマンドを実行することを指示している。postrotate の中で rotate 時に swatch を再起動掛けている。

```
case "$1" in
  start)
  start
  ;;
  stop)
  stop
  ;;
  restart)
  stop
  start
  ;;
  status)
  status
  ;;

*)
  echo "Usage: swatch {start | stop | restart | status}"
  exit 1
esac

exit $RETVAL
```

図 4 (b)　/etc/rc.d/init.d/swatch（後半）

/etc/logrotate.d/は、１週間毎に読み込まれるので、swatch のログも１週間毎に更新されていく。

(4) /etc/rc.d/init.d/swatch の作成

/etc/rc.d/init.d/swatch のファイルを作成する。ファイルの所有者は root にし rwxr-xr-x の権限を与える。ファイルの中身は図 4 の通りである。# chkconfig: 2345 90 35　と　# description: swatch start/stop script　の行は、chkconfig を動かす時に必要な記述である。

(5) /var/log/swatch/のディレクトリの作成

/var/log/swatch/のディレクトリを作成する。このディレクトリの下に、攻撃してきた IP アドレス毎のファイルと swatch.log のログファイルが作られる。swatch.log は、swatch.log-日付のファイルになって古いログが保存されていく。日付は１週間おきの日付であり、swatch.log には１週間分のログが溜まる。IP アドレス毎のファイルは、中身が攻撃してきた履歴が蓄積して残され、ファイルの更新日は最新の攻撃された日時になる。

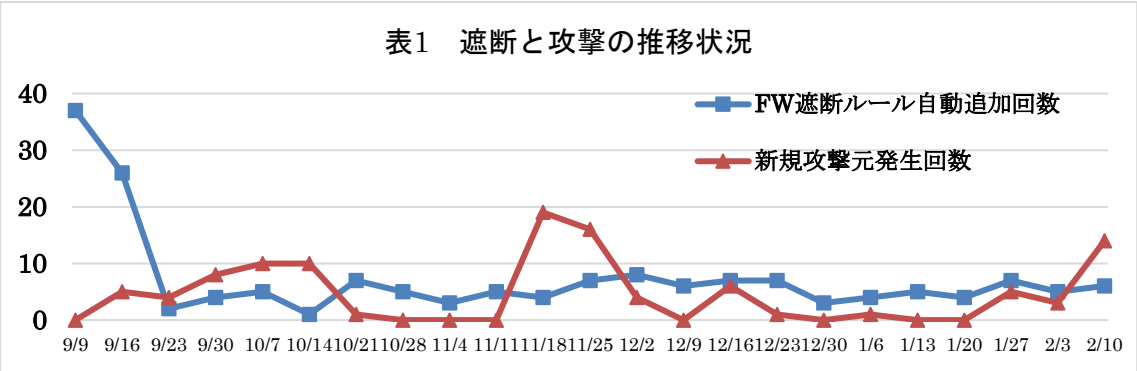
２．３　swatch の起動

/sbin/service swatch start で起動し、/sbin/service atd start も起動しておく。OS を再起動した時に自動で起動するように、/sbin/chkconfig –add swatch を実行し、/sbin/chkconfig swatch on とすることで、通常立ち上げ時にも自動起動する。/sbin/chkconfig atd on も実行しておく。

３．動作状況

設定を行った後に、FireWall で遮断されるルールが自動で追加されていくことになり、FireWall の動作が Drop にしてあるため、攻撃者から見るとパスワードを試しても応答がないという状況に見えるはずである。その結果、徐々に攻撃が減ると想定されるが、実際の推移状況はどうなったかを表 1 に示す。縦軸は、1 週間の間に起きた事象の数を示しており、左端の 9 月 9 日の欄は、9 月 2 日深夜 2 時～9 月 9 日深夜 2 時の 1 週間の間に、FireWall の遮断ルール追加が、自動的に 37 回起きたことを示している。

遮断ルールは、1 時間経過すると削除されるため、同じ攻撃元からの攻撃も含めての 37 回である。同じ攻撃元から 5 回以上の authentication failure というパスワードが間違ってますのエラーが起きると遮断ルールが追加され、1 時間の遮断中には、攻撃されても Drop 状態であるため、その間に何回攻撃されたかはログから分からない。1 時間の経過後に遮断ルールが消えると、攻撃が続いていた場合、6 回目の攻撃であるとカウントされて、再び FireWall に遮断ルールが追加される。左端の 9 月 9 日の欄の 9 月 2 日～9 月 9 日の 1 週間の間に、新規攻撃元発生回数が 0 回であるのは、その前までに多数の攻撃元から攻撃されていたため、1 回目の新規の攻撃元が現れなかっただけである。37 回は、複数の攻撃元から何回も攻撃されている状況にあった。



9 月 1 6 日の欄の 9 月 9 日～9 月 16 日の 1 週間においては、FireWall の遮断ルール追加が、26 回起きており、9 月 23 日の欄の 9 月 16 日～9 月 23 日の 1 週間においては、FireWall の遮断ルール追加が、2 回になっている。この急激に回数が減っていることが、攻撃元に攻撃の効果が無いことを伝え、諦めさせることができたように思える。それ以降のこの FireWall の遮断ルール追加の回数が 0 にならないのには理由があり、真に辞書式攻撃で試すというパスワードクラッキング攻撃でなく、authentication failure というパスワードが間違ってますのエラーを発生させている場合があるため、0 回にはならない。　具体的には、自分のノートパソコンを新規使い始めた時に Microsoft アカウントを設定し、その後、別パソコンでパスワード変更を行い、ノートパソコンの Microsoft アカウントのパスワードを間違った状態のままに使い続けると、Microsoft から認証確認が行われた時に、authentication failure というログがサーバーに残ることになる。実際に、40.97.142.157 から 262 回の認証エラーを起こし、40.97.156.53 からに切り替わり、その後、40.97.156.61 からに切り替わって 2 月 10 日時点で 91 回目の認証エラーが引き続いて起きている。これらの IP アドレスは、NetRange: 40.74.0.0 - 40.125.127.255、CIDR: 40.76.0.0/14、40.80.0.0/12、40.120.0.0/14、40.125.0.0/17、40.124.0.0/16、40.96.0.0/12、40.74.0.0/15、40.112.0.0/13 の範囲のアドレスであり、Microsoft が管理している IP アドレスの範囲の中にある。どのアカウントに対して認証確認をしてきているかは、ログから分かるが、利用者に自覚がないため、現象を知らせても直すことができずにいる。

11 月 18 日の欄の 11 月 11 日～11 月 18 日の 1 週間に、新規の攻撃元が 19 個現れている。11 月 25 日の欄の 11 月 18 日～11 月 25 日の 1 週間には、16 個現れ、その後、個数が減っている。2 月 10 日の欄

の 2 月 3 日～2 月 10 日の 1 週間には、14 個現れ、このような新規の攻撃が時期を置いてやってくることがわかる。FireWall の遮断ルール追加回数を見ると、新規攻撃元の増加に従って遮断ルールが増えている様子がない。これは、新規攻撃元は 5 回以上試すことはせず、遮断ルールの追加条件に至っていないことを示している。この新規の攻撃元は、Microsoft からのアクセスと違って、認証エラー回数が数回で終わっており攻撃を諦めてくれるように思える。

４．まとめ

導入した方法は、その後の動作状況から、非常に効果が高いことが言える。5 回以上パスワード認証に失敗すると 1 時間認証成功しなくなるという条件は、正当な利用者に伝達した後に適用しなければならないが、導入時には利用者から苦情もなく、適用を開始できた。

Microsoft が認証チェックを続けてくるのは止めてもらいたいが、利用者側の放置行為が原因でもあるので、利用者モラルを上げたいところである。

パスワードクラッキング攻撃を受けるポートサービスを利用可にしていること自体が問題でもあり、さらなるセキュリティ強化を行っているが、それについてここで紹介することは避ける。似たような問題で悩んでいる管理者に向けて、導入した方法の対策があることを紹介した次第である。パスワードクラッキング攻撃はずっと続いており、パスワード自身を強固な文字列にするように教育することも必要である。さまざまな世の中のサービスに対して、パスワードを使い回ししている利用者がまだ存在していると思われ、ユーザー教育を含めてのパスワード漏洩対策も必要である。

参考文献

(1) Dovecot 認証アタックを「swatch」で回避の巻：
https://blog.trippyboy.com/2011/centos/dovecot 認証アタックを「swatch」で回避の巻 serversmanvps/
(2019 年 2 月現在)

(2) ログ監視ツール導入（swatch）：https://centossrv.com/swatch.shtml （2019 年 2 月現在）