An Automated Incident Response System Using BIND Query Logs

John Kristoff
UltraDNS
jtk@ultradns.net

June 2, 2006

BIND Query Logs

- Easy to enable and collect, some disk space required
- Easy to parse, summarize and search
- Amenable to usage and behavior analysis
- Lacks all answer data and some query detail

Bot Detection Through DNS Query Monitoring

- Most bots do a lookup using a call to the local resolver
- Most bots use a domain name dedicated for the botnet
- Almost nothing but a bot would query for these names

How to Get This Deployed at Northwestern

- Make sure this offloads work to local admins
- Tune the false positive rate to be, in practice, practically zero
- Do not wait for someone to give you the OK before building it
- Pretend to be an expert by giving a NANOG botnets presentation

Northwestern User Status Agent (NUSA)

- Delegate network control and responsibility to local admins
- e.g. switch port config, zone maintenance, usage reports
- Web-based portal - A bunch of Perl scripts and MySQL
- A security event results in port deactivation and/or alert

NUSA DNS Bot Detection Overview

- Take a list of known bad domain names
- Tail your BIND query logs looking for said names
- If you detect badness, insert/update detail into database
- Generate alert to local admin with link to summary details

A tail, a regex and a database injection

- do: tail -f logfile | querywatch -c badnames.txt | query2db

1. tail logfile in real time, restart process when log is rotated
2. init a *bad* names list and do regex fu on the STDIN
3. send details of a matching query to a database for processing

Control scripts

- One script runs in crontab immediately after log rotation
- Kills old tail, regex, db injection children and parents
- Loops until new log starts
- The crontab script starts a second script
- The second just restarts the tail/regex/dbinject processes

querywatch

- Two modes, exact query match (express) and substring match
- Express for speed
- Substring match for search zone appends
- Substring match with sampling reduces false positives
- However, substring match with sampling misses stable bots
- http://aharp.ittns.northwestern.edu/software/querywatch

query2db

- Parse timestamp, query source, query name, type, class
- Ignore netblocks we do not administer
- Ignore well known caching/forwarding DNS servers
- Ignore any other whitelisted hosts
- Insert bad query detail into a remote database
- http://aharp.ittns.northwestern.edu/software/query2db

DNS Query Log Monitor Database

- Simple, but imperfect MySQL database schema
- Separate fields for db insert time and log timestamp
- Each query from querywatch likely to be a new db record
- NUSA periodically scanned this table for new incidents

Lessons Learned

- No false positives as far as I could tell
- Probably could have used express mode
- Mostly neutered bots, fixing them not a priority
- Alerts were ignored, could have used an escalation process

Appendix A - Other BIND Query Log Related Stuff

- http://aharp.ittns.northwestern.edu/software/
  - named-report
- http://www.nanog.org/mtg-0410/kristoff.html
- http://aharp.ittns.northwestern.edu/talks/bots-dns.pdf
- http://www.internet2.edu/presentations/jtsaltlake/
  - 20050214-Botnets-Moody.pdf
- www.cc.kumamoto-u.ac.jp/˜musashi/

Appendix A - crontab script

```
#!/bin/sh
# note: fully qualified paths removed to conserve space
TODAY="‘/bin/date +%Y-%m-%d‘"
LOGS=/path/to/bind/query/logs
parentPID=‘cat /path/to/dnsbotmon/process.pid‘
ps -afe | awk ’$3==’$parentPID’ print $2’ | xargs kill
kill $parentPID > /dev/null 2>&1
while :
do
    if ls $LOGS/server1-$TODAY.log $LOGS/server2-$TODAY > /dev/null 2>&1
    then
        dnsbotmon.sh &
        echo $! > /var/run/dnsbotmon.pid
        break
    fi
done
```

Appendix A - dnsbotmon script

```
#!/bin/sh
LOGS=/path/to/bind/query/logs
BADNAMES=/path/to/bad/names/list
TODAY="`date +%Y-%m-%d`"

tail -f $LOGS/*-$TODAY.log | querywatch -c $BADNAMES | query2db
```