

Bro Usage at The Ohio State University

2007 Bro Workshop

Seth Hall
The Ohio State University
July 25, 2007

Primary Usage: Bro as Knowledge Base

- Operationally useful information.
- As new attack methodologies are learned during incident detection and response, Bro can be programmed to notice similar events in the future. Even complex multi-step events.

Secondary Usage: Bro as Centralized Logging Tool

- Web logs for entire organization.
 - No cooperation needed from the owners of the web sites.
 - This is currently used in reporting hosts which are infected with Beagle/Bagle.
- DNS logs
 - DNS servers will tend to log a lot of information that we don't necessarily care about. We can log only what we care about with Bro.

Scripts and Experiences

SSH from “bad” countries

- Concept is simple, we never see valid, successful SSH sessions from certain countries.
- GeoIP support in 1.3 enables the ability to get a country code based on an IP address.
- SSH is encrypted, so we watch for the quantity of data returned from the server. Once a byte threshold is crossed, the client is deemed to have authenticated successfully.

Example

```
const bad_countries: set[string] = {"RO"};
event protocol_confirmation(c: connection, atype: count, aid: count)
{
    host = c$id$resp_h;
    if( atype == ANALYZER_SSH &&
        lookup_location(host)$country_code in bad_countries )
    {
        # do a notice
    }
}
```

This code is not what we're using and doesn't work quite right. It doesn't understand when someone successfully authenticates, so it does the notice on every protocol confirmation from the bad countries.

SQL injection detection

- We can match attacks with a signature.
- Usually, attacks that are used for stealing data require a large number of request.
- Even if an attacker does IDS evasion tricks, it's unlikely they will evade detection completely if they try to siphon data from the application.
 - More difficult to catch are early generic probes.
 - Early probes should be detectable by watching for signatures in http reply traffic.

SQL injection IPS?

- It would be nice to do something about the attacks.
- If Bro determines that an attack is occurring against one of our sites, we would like to block the attackers.
 - It is painful to have data stolen.
- More detail on next slide.

SQL injection Protection from Dedicated Attackers

- In order to protect ourselves from attackers with many hosts, we dramatically lower the threshold of SQL injection requests before block.
 - If Bro has determined that an attack is ongoing, we can have it block any host that sends 1 or 2 requests to the attacked site that appear to be SQL injections instead of waiting for the original threshold to be crossed again.
 - Attackers with many source addresses still have a very hard time grabbing data.

SMTP Rejects counter

- Central idea is that spammers have bad address lists.
- If an SMTP server receives mail for an address it doesn't accept, it will typically return a status code in the 5xx range.
- We can watch the percentage of addresses rejected out of the total number of destination addresses for a time period.
- In the end, we usually catch hosts spamming hours before SpamCop sends reports to us.

SMTP Rejects continued

- Future extensions..
 - If a host seems to be spamming, sample a few subject lines from the host before sending the notice.
 - See if host is accepting connections on mail ports. It should stop us from blocking new mail servers that we did not add to the list of known mail servers.
 - In case the host is just forwarding spam mail.

SMTP Rejects Example

```
const smtp_reject_codes: set[count] = {
    550, # Requested action not taken: mailbox unavailable
    551, # User not local; please try <forward-path>
    553, # Requested action not taken: mailbox name not allowed
    554, # Transaction failed
};

event smtp_reply(c: connection, is_orig: bool, code: count, cmd: string,
                msg: string, cont_resp: bool)
{
    local foo = smtp_reject_hosts[c$id$orig_h];
    if ( code in smtp_reject_codes && c$id !in foo$rej_conns )
    {
        ++foo$rejects;
    }
    ++foo$total;
}
```

Google's Safe Browsing API

- Google publishes **huge** hash list of sites that attempt to exploit browsers.
- Testing indicates many false positives.
- False positives can be remedied by watching for downloads of executable files within a minute or two by the client after the questionable URL.
- Probably best suited for a cluster because of the high processing load.

SMTP Originators

- Show which hosts are actually originating mail
- Haven't found how exactly this is useful yet, but it's nice to know if some internal host is forwarding a lot of mail through campus mail servers.
- If anyone can think of how this would be useful, let me know!

SMTP Originators example log

- 1185281343.54232 128.146.4.138 :: 128.146.216.134 -> 64.18.6.14
- 1185281347.18726 128.146.4.138 :: 128.146.216.134 -> 64.18.6.14
- 1185281352.2651 128.146.4.138 :: 128.146.216.81 -> 64.18.6.14
- 1185281353.95947 128.146.4.138 :: 128.146.216.134 -> 64.18.6.14
- 1185281354.95924 128.146.8.219 :: 128.146.216.81 -> 66.196.97.250
- 1185281355.28517 164.107.86.239 :: 128.146.216.134 -> 160.36.0.83
- 1185281355.62097 128.146.4.138 :: 128.146.216.134 -> 64.18.6.14
- 1185281355.80536 128.146.230.111 :: 140.254.47.13 -> 205.145.128.110
- 1185281356.32031 128.146.40.180 :: 128.146.216.133 -> 208.47.184.2
- 1185281356.58765 128.146.4.138 :: 128.146.216.81 -> 64.18.6.14

Future work

- Monitor and record all SSL certificates for entire network.
- Extend SSH logins from “bad” countries to encompass other authenticated protocols such as RDP.