



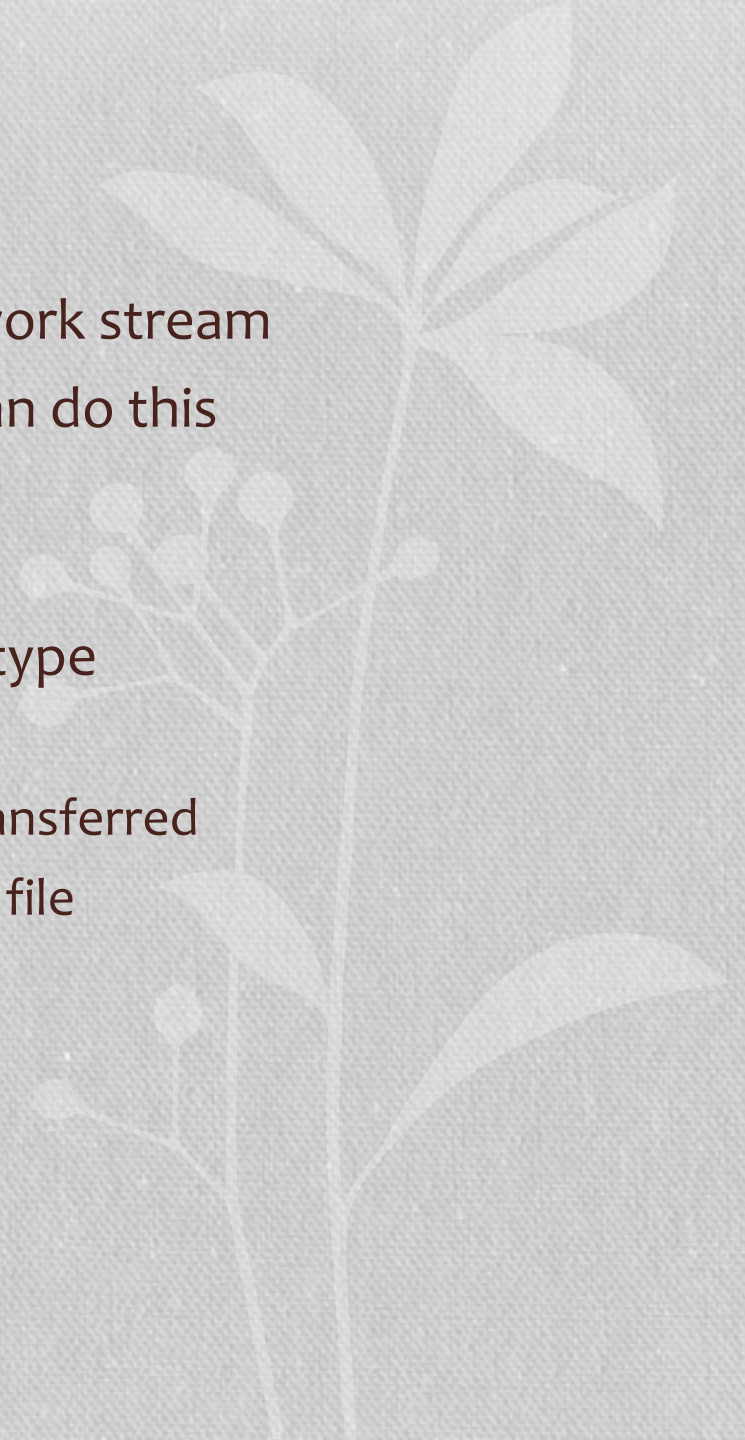
BRO FILE CARVING

Using scripts to scarf files from
pcaps and the wire

by Bill Stackpole

File carving

- Common desire to extract files from network stream
- Tools (tcpextract / network miner / etc) can do this
 - Usually w/out protocol knowledge
- Often work by finding START of some filetype
 - Collect next “n” bytes
 - Provide an approximation of file that was transferred
 - Can require additional carving to extract the file



File carving

- Bro is protocol aware
- Can use this to its advantage
 - Identifying protocols
 - Identify mime types
 - Trigger on mime type detection



File analysis in bro

- Built-in file analysis mechanism – “File Analysis Framework”
- One analysis: write a detected file to disk
 - Need to ask it to do so for each file of interest



Prototype

#Create a new event handler “file_new”

#When Bro finds a file being transferred (via any protocol it knows about),

write a basic message to stdout and then tell Bro to save the file to disk.

```
event file_new( f: fa_file)
{
  local fuid = f$id;
  local fsource = f$source;
  local ftype = f$mime_type;
  local fname = fmt(“ extract-%s-%s”, fsource, fuid);
  print fmt(“*** Found %s in %s. Saved as %s. File ID is %s”, ftype,
fsource, fname, fuid);
  Files:: add_analyzer(f, Files:: ANALYZER_EXTRACT,
[$ extract_filename = fname]);
}
```

Run the script against a pcap file

- `Cd /home/bill/brostuff/temp`
- `bro -C -r ../pcaps/net-<tab> ../scripts/extract<tab>`



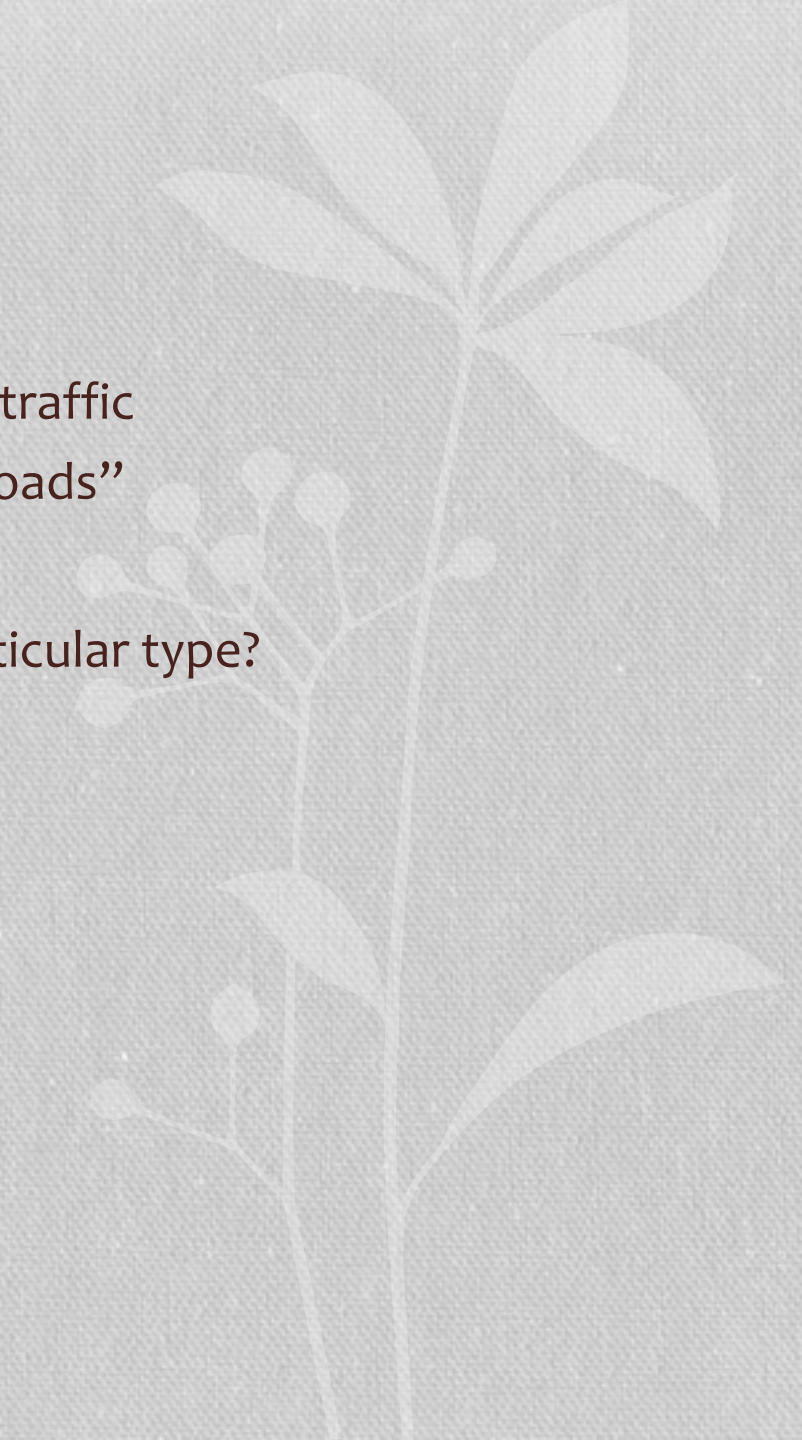
Review content

- STDOUT: 1 line per file detected / found / carved
- Directory contains
 - log files (as before)
 - An additional directory
 - extract_files
 - Contents: one file per file detected / carved
 - Keyfield: bro id tag

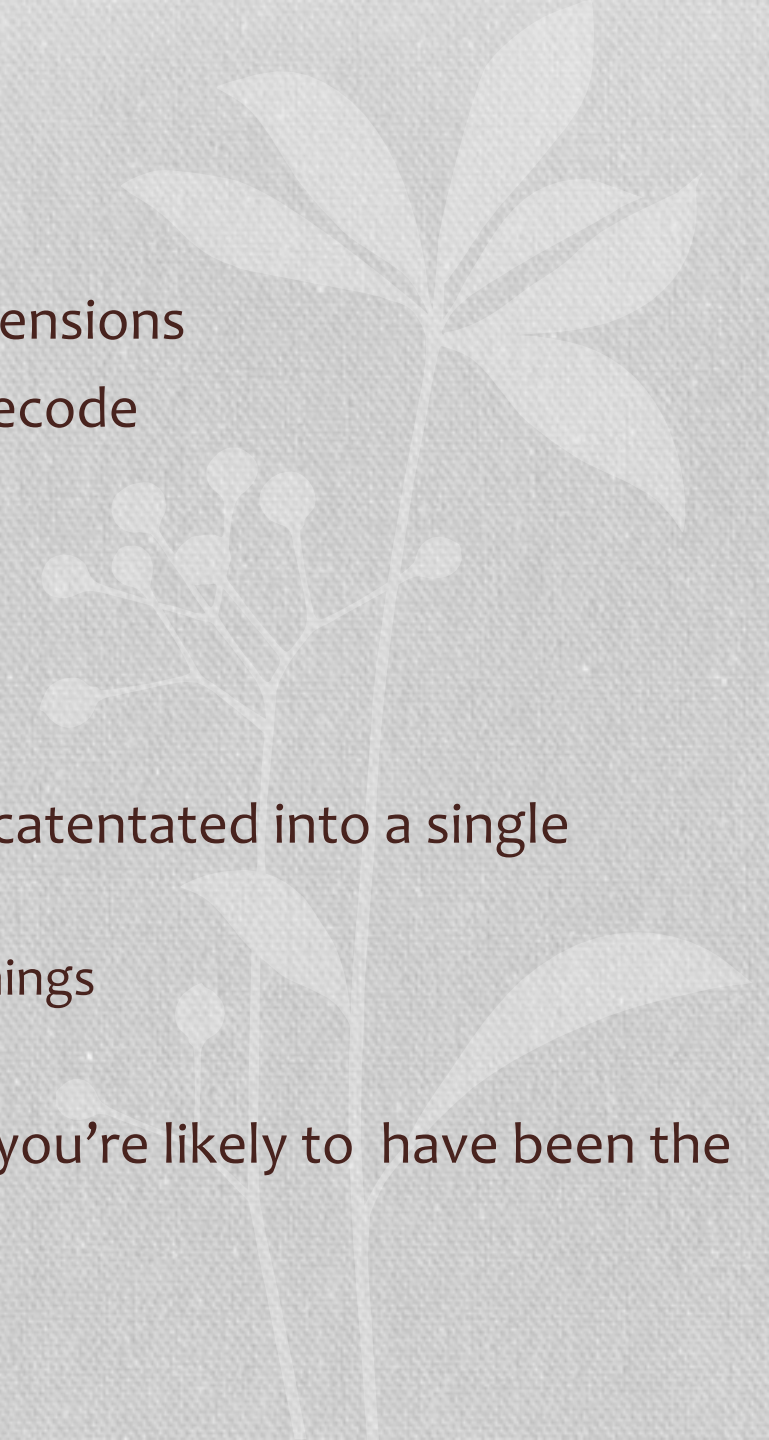


LOTS of content

- And much
 - is not what most would expect from web traffic
 - Not what most think about as “file downloads”
- What if
 - You ONLY wanted to extract files of a particular type?
 - No problem!
- Bro knows MIME type
- Figures it out by examining the file
 - Does NOT just trust what the protocol says!



MIME?

- Multipurpose/multipart Internet Mail Extensions
 - Originally used to replace uuencode/uudecode
 - Unix-to-unix encode / decode
 - Generates inline blocks of plain text
 - Unmolested by the internet
 - Each representing another “file”
 - Allows multiple message parts to be concatenated into a single message
 - For message xfer efficiency among other things
 - If you’ve ever received file attachments, you’re likely to have been the recipient of an inline MIME!
- 

MIME

- Popular use is to describe what KIND of content each message part contains
 - MIME types even used by many non-MIME applications to describe their own data types!
- HTTP uses MIME formatting
 - For transactions that send multiple pieces of data at once
 - i.e. form submissions w/POST
 - Most everything HTTP is tagged with a MIME type
 - Text/html
 - Text/plain
 - Image/jpg
 - Image/gif
 - etc

Lets carve executables!

- Same script as before – but we’re going to change a few things:

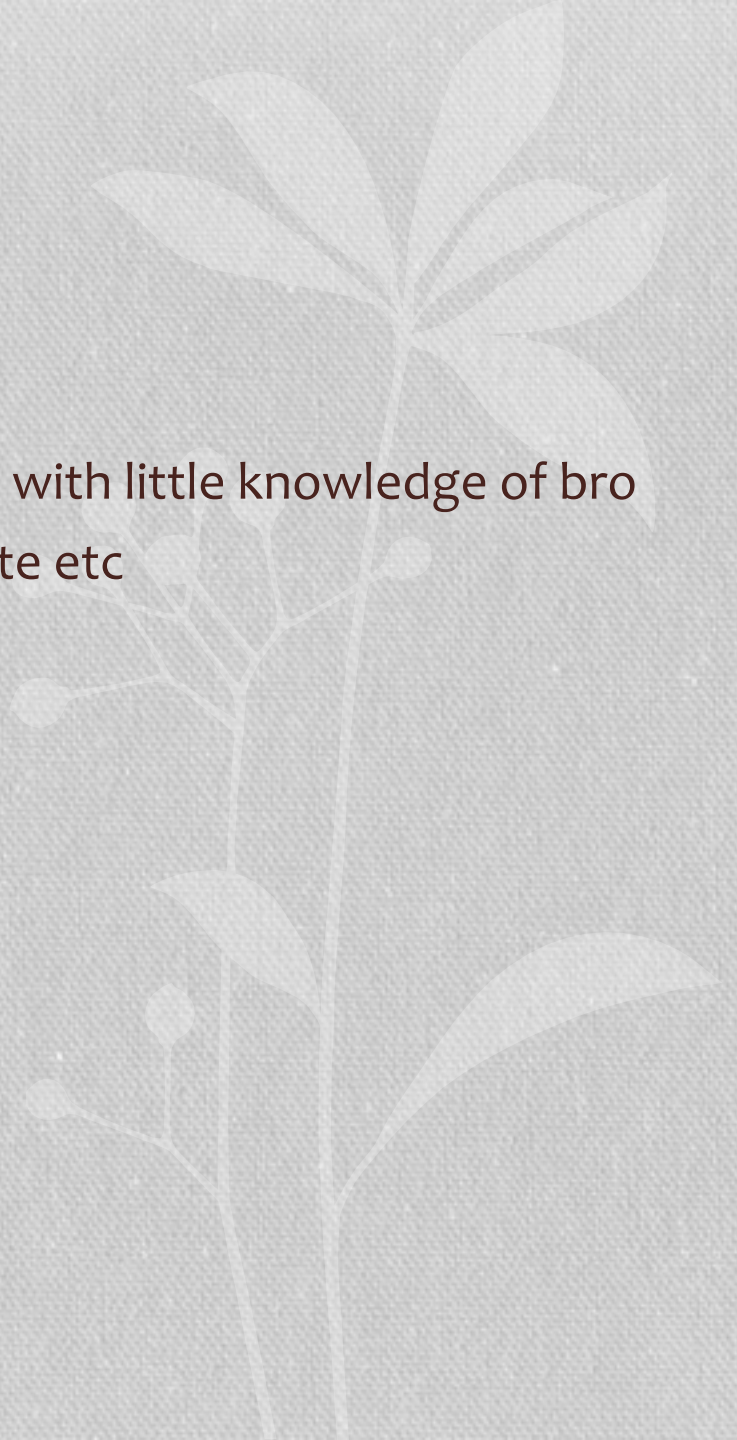
```
#!/usr/bin/env bro
```

```
# When Bro finds an executable file being transferred (via any  
# protocol it knows about), write a basic message to stdout  
# and then tell bro to save the file to disk
```

```
event file_new( f: fa_file) {  
    # Check that we have a MIME type value in this record  
    if (f? $mime_type) {  
        # See if the type is one we care about  
        if( f$mime_type == "application/ x-dosexec" || f$mime_type ==  
            "application/ x-executable") {  
            local ftype = f$mime_type;  
            local fuid = f$id;  
            local fsource = f$source;  
            local fname = fmt("extract-%s-%s", fsource, fuid);  
            print fmt("*** Found %s in %s. Saved as %s. File ID is %s", ftype,  
                fsource, fname, fuid);  
            Files:: add_analyzer( f, Files:: ANALYZER_EXTRACT  
                [$extract_filename = fname]);  
        }  
    }  
}
```

Note changes

- Shebang line (`#!/user/bin/env/bro`)
 - Can run this script as a standalone
 - Can give the script to someone else (analyst) with little knowledge of bro
 - Must have appropriate permissions to execute etc
 - `Chmod 755 <script_name>`



Lets carve gif images!

- Modify our existing script to change the MIME type



Extracting files from live network traffic

- Integrate this capability into TSO bro instance
 - Useful to extract **every** occurrence of a certain file type in near-real time
 - As they pass across your network segment
 - Rather than carving from saved pcap files



How to get bro to run all the time?

- Config file: local.bro
 - In TSO should be in /opt/bro/share/bro/site/local.bro
 - Not **really** a config file
 - Simply a script that bro loads and runs at startup
- Default version w/TSO
 - Loads other bro scripts
 - Perform useful functions
 - Detecting scans
 - Logging applications in use on the network
 - Adding geoIP lookups to certain protocol logs
 - etc



Local.bro

- A local bro config file
- Where you will add your customizations
- Can simply paste our code to the bottom of local.bro
- Can reference script(s) stored in other locations



If you change local.bro

- Need to use the “bro control” program to check, install, and restart bro.
- Three steps:
 - *broctl check* – performs syntax/sanity check of config
 - To see if you’ve broken anything
 - *broctl install* – make config changes “active” (thru “installing” changes)
 - *broctl restart* – restart the bro service to force it to read and move into memory your modifications

Lets push our bro script into local.bro

- Cd /opt/bro/share/bro/site/
- Cat *our_file.bro* >> bro.local
- Less bro.local
 - Ck that it worked the way we thought it should
- Call and run broctl
 - *broctl check*
 - *broctl install*
 - *broctl restart*



Ck to be sure it's working

- After running for some time
 - Check `/nsm/bro/logs/current`
 - Review `files.log` (ck for files extracted)
 - If yes, ck “analyzers” field to see which files had EXTRACT analyzer applied to them
 - i.e. `cat files.log | bro-cut -u ts fuid analyzers conn_uids extracted | grep EXTRACT`
 - Should get 1 line per extracted file
 - Things you don't recognize? Run `*strings*` against the file in question