Beyond Fast Flux: Parasitic Command And Control Networks in the Near Future

www.immunityinc.com

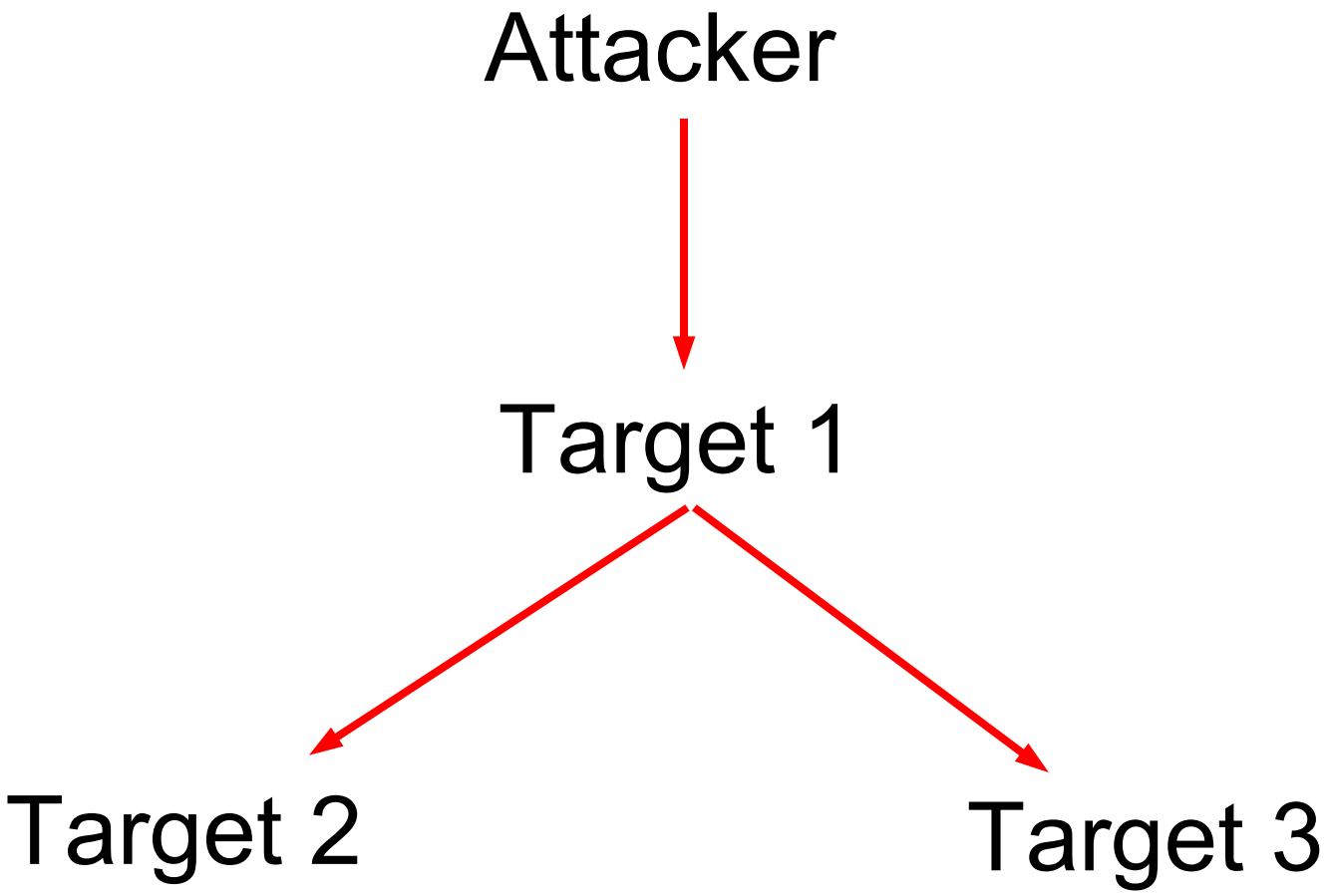
Agenda

- Problems of scale when hacking
  - Client-sides
- Immunity's PINK Framework
- Trojaning hard targets
  - Immunity Debugger Parasitic Infection

Targets are ephemeral

- Time
  - Your workstation turns on and off as you come to work
- Location
  - Your laptop travels across network security boundaries
- Configuration
  - Your server is upgraded, reconfigured, network infrastructure changes around it

Command and Control in most hacking platforms is a tree



IMMUNITY

Networks are not trees

- A fully connected graph is what we want
  - Self routing with some human input
- This is a hugely expensive solution
  - Management costs
  - Development costs
    - Need to emulate TCP over thousands of protocols
    - Those who don't use TCP are doomed to re-implement it...

Building and storing routing tables is a hard problem

- Harder for us due to covertness
- We don't want any node to have a larger picture of all the other owned nodes than it absolutely has to
- Automatic solutions are possible, but for now, manual operation of routing is easiest

Scalability problems

- Management of one hundred ants is easy
  - Picture of thirty million ants
- A good client-side vulnerability can be used to own a quarter million boxes a day
- Future work involves self-directed worms

Asymmetric attack means we need to not have a rack of machines

- Portable C&C
- Scalable C&C
- Covert C&C
- Immunity's PINK infrastructure solves these problems

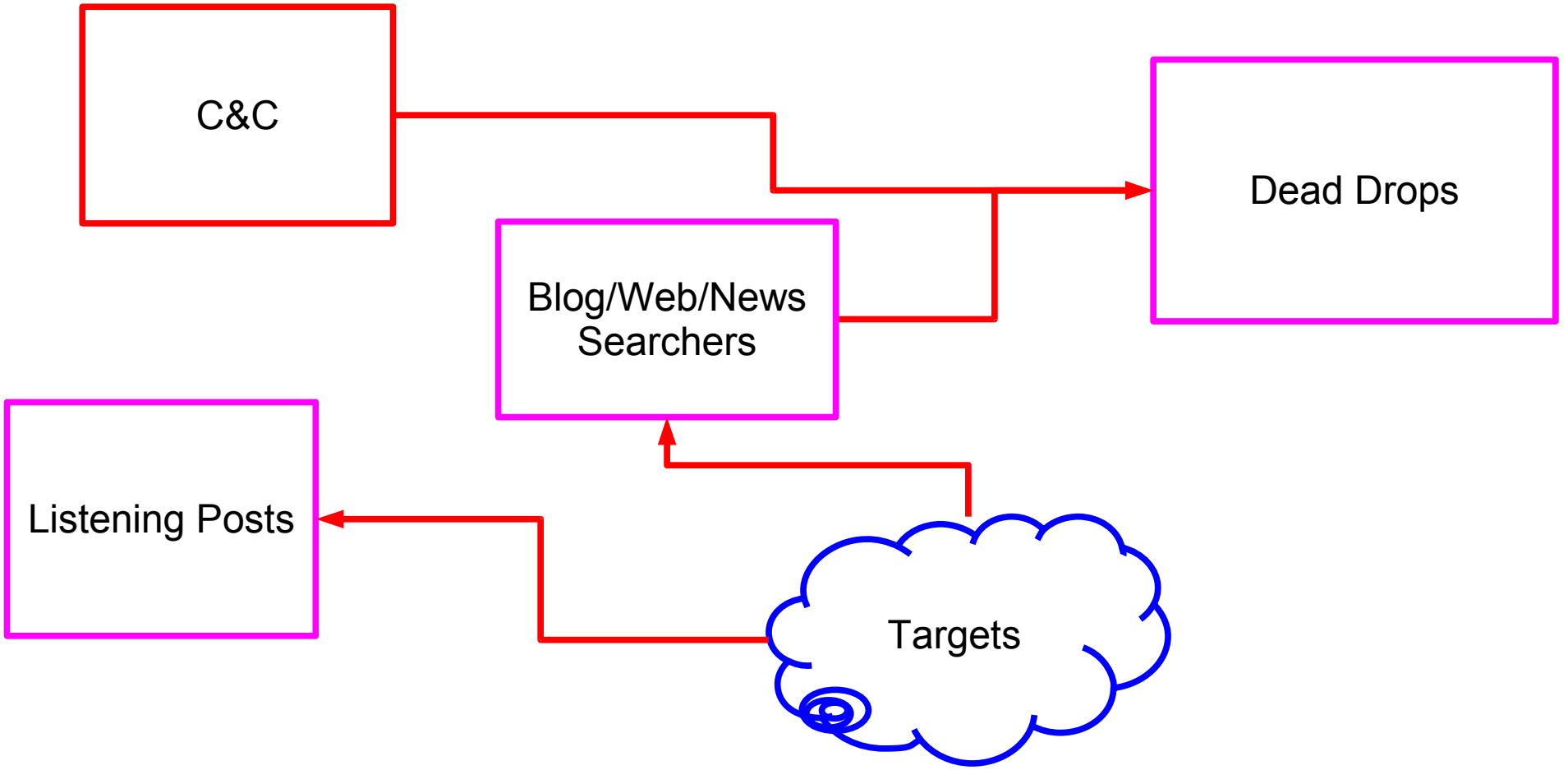
Current Botnet C&C technology

- IRC
- HTTP to single server
- Fast-Flux of DNS Servers
- Storm P2P protocols

Covertness or Reliability?

- P2P is reliable, not covert
  - Requires chatty communications on the network
  - Difficult to pass through strict proxies
  - Easily fingerprintable

PINK C&C Framework



IMMUNITY

Blogsearch

- Blog searching is the current best parasitic host protocol for PINK
  - Almost instantaneous responses
  - Easy to find hosts for our blogs
  - Lots of signal to hide in
- Any search engine will do though

IMMUNITY 

PINK DEAD DROPS

- <Cover Text>
- <TRIGGER>
- <base 64><RSA Encrypted/Signed Command></base64>
- <END TRIGGER>
- <More Cover Text>

Each Target is looking for multiple triggers

- Goal is to divide our targets into manageable sets
  - Per Country
  - Per Company
  - Per Domain
  - Per Time-of-exploit
  - etc
- “All hosts from immunityinc.com” please contact listeningpost.my.com using HTTP MOSDEF on port 443
- All target.com's please deliver any .xls with “Payroll” string to email address bob@example.com

Signed and Encrypted payloads prevent replay attacks with removal kits

- Triggers need to be signed with time-based key as well
- Making triggers strings of random words makes it hard for search engines to filter our requests

Client-side conclusions

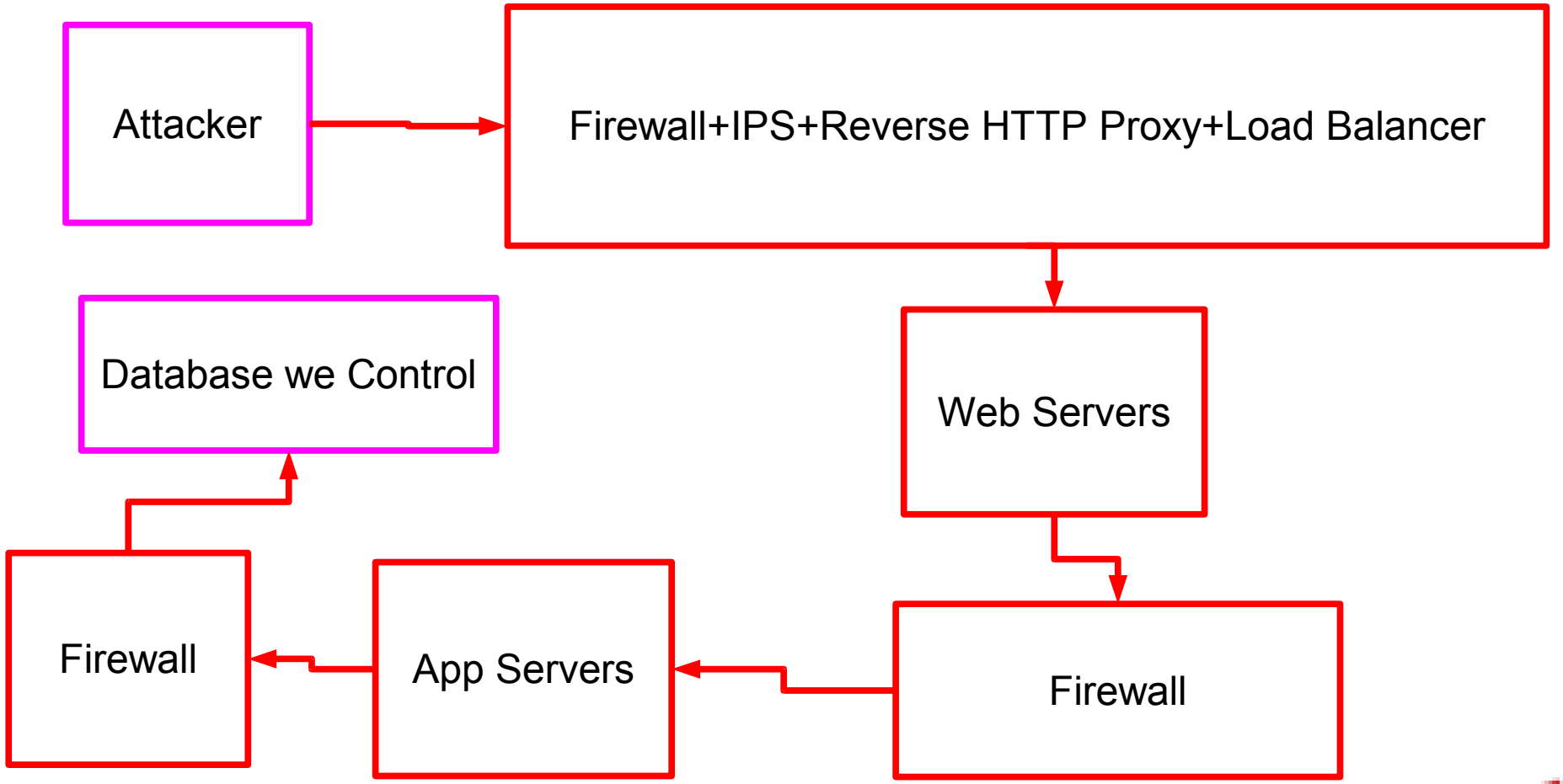
- Currently in Beta-testing state – pushing out to CANVAS shortly
- Parasitic C&C is:
  - Nearly impossible to detect and monitor
  - Easily re-targetable to any search engine or search option on a web page
  - Does not require expensive infrastructure to maintain

IMMUNITY 

Servers and hard targets

- Servers may not be able to contact us via HTTP
- Need way to connect to stationary targets behind firewalls and application proxies covertly
- Each target is different!
- Example target: MS SQL Server 2005 in strict DMZ tier

Every web application is a unique snowflake



IMMUNITY

Custom automatic backdoors

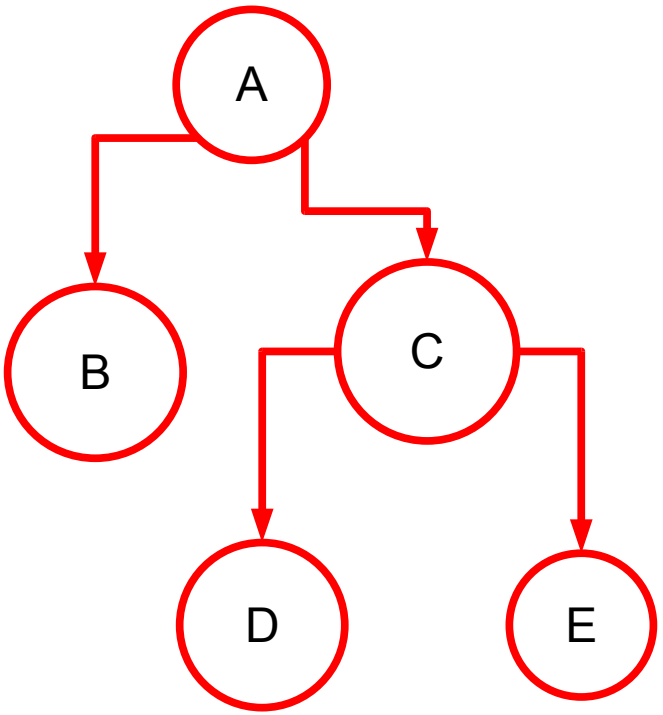
- Use Immunity Debugger to analyze target .exe/.dll
- Send traffic to it and trace where our triggers are seen
- Create custom patch to PINKize target .dll and write this to disk and memory
- Box is now trojaned in a way that does not require direct connectivity!

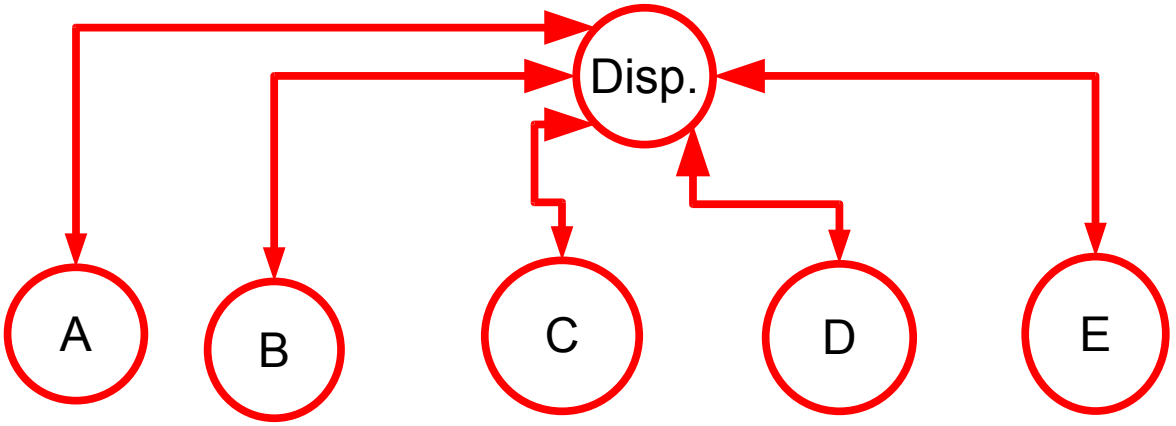
Why Immunity Debugger?

- Includes built in analysis engine
- Full Python scripting API can do both dynamic and static analysis
- Send data to the server and then see what API it triggers
- Mutate our parasite to look statistically like the target program
- Trojan in memory or on disk or both

Avoiding Structural BinDiff

- Change all CALL opcodes to point to our dispatcher
- Have dispatcher send hooked API's to our code instead





IMMUNITY

Overall Conclusions

- Botnets and trojans will be extremely difficult to find and analyze in the near future.
- Nascent market shift to automated incident response as part of vulnerability analysis faces ongoing challenges as attackers build one-time custom-use trojans

IMMUNITY