Presenting Mongoose
A New Approach to Traffic Capture

(patent pending)

presented by

Ron McLeod and Ashraf Abu Sharekh

January 2013

Outline

- Genesis - why we built it, where and when did the idea begin
- Issues – requirements
- What we built and how it works (mostly)
- Recent and current challenges
- Our biggest challenges and ongoing Work

Genesis - Why?

- Network administrators require situational awareness to detect:
  - Scanning, Intrusion, Exfiltration, Policy Violations and System Performance Issues
  - Most organizations that we have encountered in our security practice are not monitoring or logging their network activity beyond bandwidth usage.
  - The exceptions in our experience being large government entities and Universities with significant IT staff.
  - So we started asking Why not? …More on this later.

Genesis - When

- It all came together during a “Walk in the Desert” and the statement “We would like to monitor the activity of the network from outside the network. And (maybe) without the users knowing that it is happening.”
- For security reasons I can’t say what desert or who made the statement or who I was walking with.
- There were other pre-existing sparks but this was a watershed moment.

Issues - Privacy

Privacy was of significant importance.

- Network owners did not want external monitors to know or leak information about network structure.
- IP addresses may be interpreted as personal private information in some jurisdictions.
- External monitors must not be able to tie traffic to a machine or a user. Only internal Network admins should be able to do this.
- Users (and administrators) are nervous about payload capture, until there is a problem.
- Communication to and from the network must be secure.

Taken together these issues meant that we would have to be able to modify captured traffic.

Issues - Network Architecture

Independence

- Must work regardless of physical layer components (wireless, wireline)
- Must not require knowledge of sub-netting and NAT'ting within the network.
- Must not rely on the presence of firewalls or services such as active directory.
- Must be able to continue to monitor and control a device that moves throughout the network and beyond.
- Must not be blinded by the use of VPN's

Issues - Visibility

- Must not be noticeable to the end user through:
  - performance (CPU, Memory, bandwidth)
  - or as a running application.

Issues - Speed to Deployment

- **To understand this issue, we must first describe our typical incident response experience.**
  - An organization suspects a data breach or is performing an audit.
  - Q & A with the network administrator:
    - **Can you draw me a diagram of your network structure so I can decide where to put the taps?**
      - **No. I didn't build it.**
    - **Can you tell me which of your routers are capable of producing flow or which switches have port mirroring?**
      - **What's flow?**
    - **When do you need this?**
      - **Today.**
- **Alternatively : in covert deployment speed may be of the essence.**

Issues - Control

- Must be capable of remote interdiction and modification.
  - if a machine is doing a bad thing I need to be able to stop it immediately regardless of my network infrastructure, while maintaining the operating state for forensic analysis.
- Interdiction should not obviously be an interdiction unless I want it to be.
  - i.e. if someone is stealing data from a machine I need to stop the theft but I don't want them to run away before the authorities get there.

Simple – Right?

Two Years later….

Mongoose is a host based traffic collection system that:

- installs in a few minutes as a downloadable kernel patch and service.
- captures inbound and outbound traffic at the host.
- Builds a proprietary representation of each packet and places it in a “dump file”.
- dump files containing (initially) 20,000 packets (1.5 meg) are shipped approximately every 2 minutes to a cloud server farm via a secure SSL connection.

At the server farm:

- Dump files are processed to produce a proprietary flow representation and stored in a client database.
- Alert and classification systems constantly scan the flow data (ongoing development)

Through a web interface:

- Network administrators can log in from anywhere and get a near real time picture of their network activity.

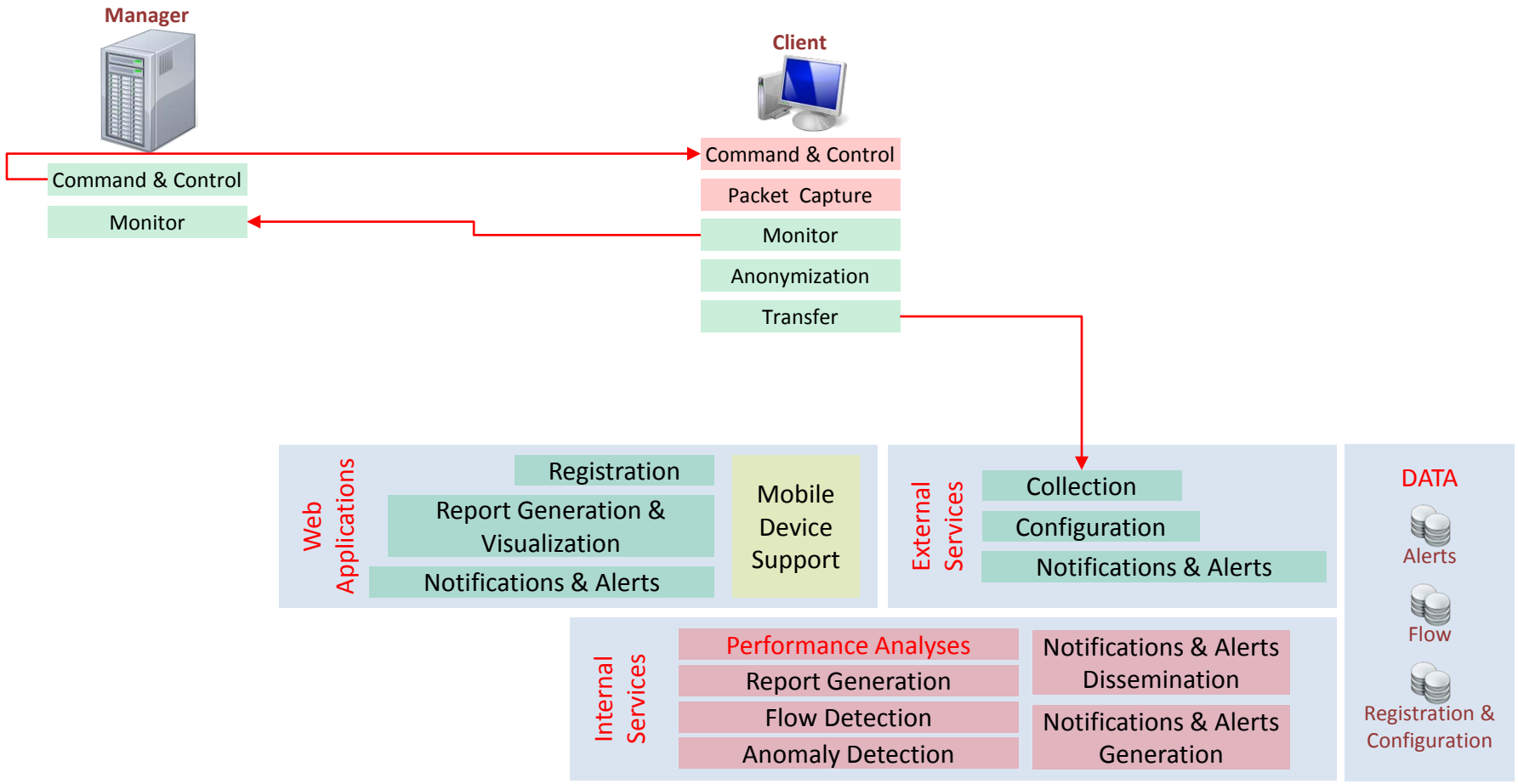
Through a software “Manager” Console:

- Network Administrators can exercise remote control over certain aspects of the client machine.

Manager Functionality
(the parts we can talk about)

- Remotely:
  - retrieve machine information (cpu, memory, processor, operating system etc) for diagnostics or inventory.
  - **start and stop network access**.
  - adjust size of payload capture up and down in real time by any number of bytes to full payload. (**Hybrid Capture**)
  - adjust the number of packets per dump file.
  - start and stop various components in the Mongoose system.
- Other functionality under development – since we are in the kernel you can let your imagination go wild.

Mongoose Architecture



Beta Testing and Experimentation

- Approximately eight months of Beta testing in up to five live production sites operating under confidentiality agreements (geographically distributed).
- 20 - 50 client machines per site reporting to a single collection and processing site.
- Real implementations now have limited shared access to a single collection server and multiple processing nodes, one per customer.

Beta Testing and Experimentation

Excluded traffic

- initially captured everything.
- 90 – 98% of all traffic was local broadcast and link layer traffic for address resolution, name services etc.
- much of this was never meant to exit the local link, but we sent data on all of it out of the edge router….and quickly impacted the bandwidth.
- We currently exclude much of this traffic but may give the network admin the ability to sample it for brief periods.

Beta Testing and Experimentation Environment

- Testing and Development Environment
  - Multiple Servers (VM's) located in both Quebec and Alberta.
  - All beta clients located in Nova Scotia.
- Commercial environment
  - Multiple Servers (VM's) located in Nova Scotia.

Current and Recent Challenges

- Choosing a platform
  - Windows 7 family (Vista, Win 7, Win 8, Server 2008 and so on..)
    - Will not work with XP, server 2003 etc..
  - Android development in the near future

Current and Recent Challenges

- first challenge: capture and modify traffic.
  - We do it in the kernel. We don't use pcap. The rest is secret sauce.
- second challenge: process and ship the packets in a way that does not affect computer performance and is not easily visible to the user.
  - processing dump file on the client causes a cpu spike of < 1 sec/file.
  - Shipping files causes a much smaller cpu spike that does not exceed the normal operating range of other running applications.
    - i.e. if CPU is operating at 30% – 50% then shipping spike is within this range.
- Experiments involved changing the processing algorithm, dump file size and shipping frequency until an acceptable performance level was achieved.
- Dump file size is configurable through the Manager

Current and Recent Challenges

- Secure communication
  - Each Mongoose client contains a unique certificate for use in SSL communication with the collection servers.

Current and Recent Challenges

- Constructing Flows
  - Originally less than 100 lines of C code.
  - 20,000 packet representations are processed in less than 1 sec.
  - Experiments with map/reduce and Hadoop clusters have not yet proved beneficial over our current implementation given the current number of clients (dump files) collected
  - This is due largely to the overhead associated with the Hadoop approach.

Current and Recent Challenges
Some Hadoop Results

- Hadoop with one name node and two data nodes vs existing processing.

Existing: < 1 sec per dump file

Files	Hadoop processing time in secs/file
20	2.45
40	1.28
53	0.96

Using six data nodes we processed 750 dump files at a rate of 0.11 secs/file (best result).

Current and Recent Challenges
Alerts

- Currently four alert categories
  - blacklist of external ips
  - sensitive ports
  - exception reporting on specific machines
  - behavioral classification (neural classifiers)
- Near real time Alert conditions remains our biggest challenge.
  - currently experimenting with algorithmic and system modifications to improve alert performance.

Current and Recent Challenges
Behavioral Alert Classification

Interesting results from neural classifiers for user/machine pairing

- training with 72 hours of real flow data from the population of a beta client
- using flow data statistics similar to that described in my presentation at FloCon 2006.
- multilayer feed forward network with back propagation of error.
- neural network maintains 100% discrimination accuracy for a small sample set of (3) machines for one month without re-training. Not tested beyond this point.
- challenges include the incorporation of the neural classifier into the alert processor and scaling of test population. One is limiting the other. Would like to have the ability to dynamically expand and contract the number of machines we are classifying to test the scalability.

Interesting areas of experimentation and development

- User signatures - isolate an individual based on network traffic. For use in insider masquerade attacks and for covert surveillance.
- Device signatures – isolate a device based on traffic signature. For use in authentication and surveillance.
- Application signatures – classify an application.

Some Unresolved Questions for our Beta Clients

- How long do you want to maintain your flow database? 30 days?
- How long do you need full payload capture to be running? 1 minute per sample?

Our biggest challenges and ongoing work

- performance on the client.
- secure remote communication.
- server infrastructure that is sustainable in the business model.
- provisioning and decommissioning customers and clients.
- Near real time alerts and classification.

Thank You!

Questions?