

# Passive Capture Engine



network  
resonance

Kevin Dick  
[kevin@networkresonance.com](mailto:kevin@networkresonance.com)

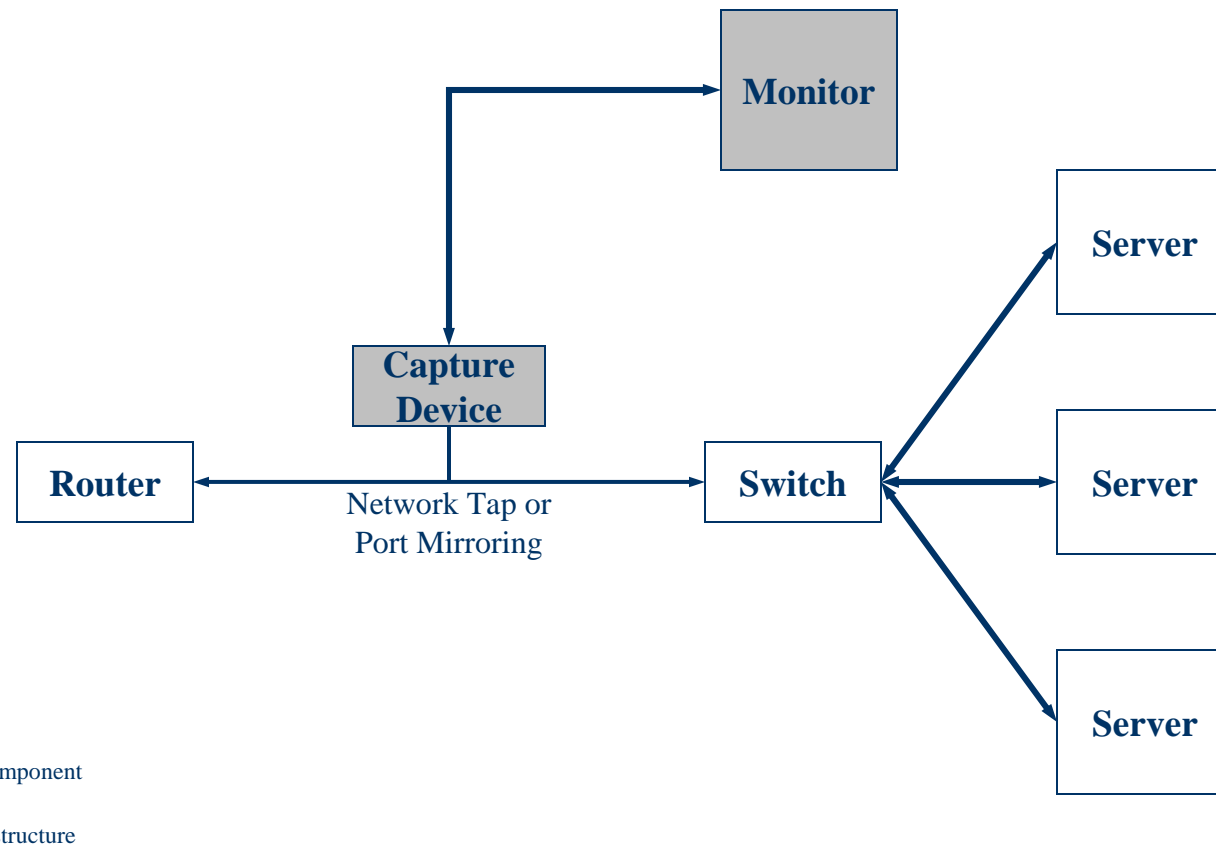
# Enterprises Hate Server-Side Agents

- Installation on potentially hundreds of servers
- Must patch and upgrade every agent on every server
- *Concerned about **performance degrading***
- *Concerned about **reliability decreasing***
- *Concerned about **security risk***

# Inline Devices Aren't Much Better

- Does reduce software maintenance hassle
- But requires significant network topology reconfiguration
- *Still degrades performance*
- *Still decreases reliability*
- *Still introduces security risk*

# Passive Capture Provides Solution



# The Good News for Solution Vendors

- Greatly simplifies management and upgrading
  - *Much lower TCO for customer*
- 
- Very fast to install and configure
  - *Much easier to do production-grade pilots*
- 
- Zero impact on performance, reliability, or security
  - *Minimizes objections from IT*

# The Bad News for Solution Vendors

- Probably a whole new technical discipline
- Probably not a unique competitive advantage
- *Hard to achieve production **performance***
- *Hard to achieve real-time decryption of **SSL***
- *Hard to achieve application-level **fidelity***

## Nevertheless, There Is a Trend

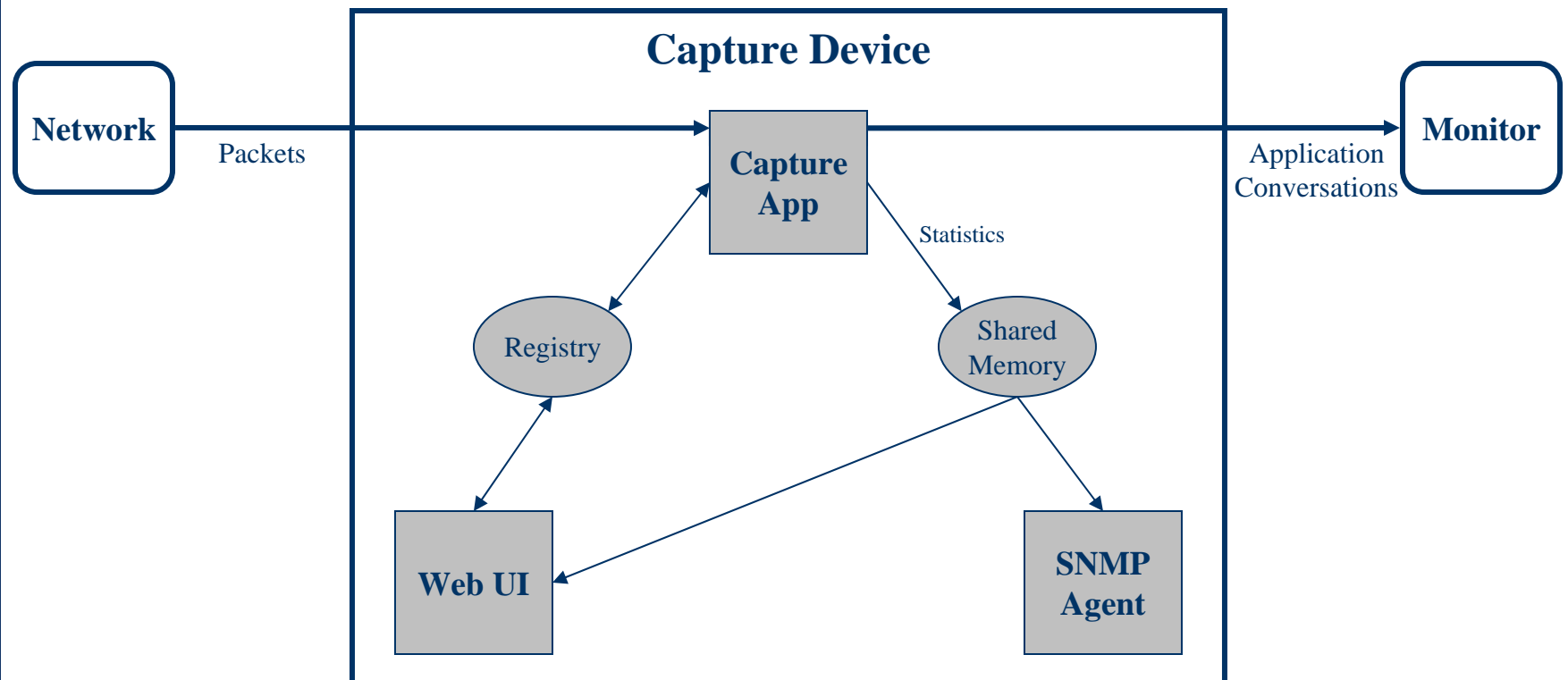
- CompuWare Vantage (via Adlex acquisition)
- CoRadiant TrueSight
- Entrust (via Business Signatures acquisition)
- Mercury Interactive (via BeatBox acquisition)
- Quest Foglight Experience Monitor/Viewer (via Xaffire acquisition)
- TeaLeaf RealITea
- Computer Associates (Via Wily Technology acquisition)

# The Passive Capture Engine

*Real-time access to application messages without integration*

- Full Reassembly      extracts application level messages
- Real Time              extracts them as applications receive them
- SSL-capable          uses keys to decode encrypted traffic
- Extensible             can add new protocol decoders
- Easy-to-integrate     delivers extracted messages via a socket

# How it Works



# Captured Data for HTTP Conversations

- HTTP Requests and Responses
  - Lengths
  - Headers
  - Payloads
  - Any MIME Type
- TCP and SSL Metadata
  - TCP Client Host-Port
  - TCP Server Host-Port
  - TCP Retransmits
  - SSL Session ID
  - SSL Cryptosuite
  - SSL Certificate Chain
- Performance Data
  - Total Server Time = Last Packet of Response - First Packet of Request
  - Server Think Time = First Packet of Response - Last Packet of Request
  - Estimated Round-trip Time (RTT)
  - End-to-end Time = Total Server Time + RTT

# Proven Platform

- First OEM customer deployed over two years ago
- Second OEM deployed over a year ago
- Signed Indicative Software as third OEM in July
- We estimate deployment at over 50 large end-customers
- Finance, insurance and e-commerce are primary verticals

# Extensible Capture Platform

- Customizable formatting of captured data
- Customizable delivery of captured data
- Modular architecture accommodates additional protocols
  - VoIP
  - Instant Messaging
  - Message Queuing
- Full device management through Web interface
- Device statistics also accessible through any SNMP console

# Next Generation Architecture Released

- Dynamically assembled decoder chains
  - Enables simultaneous multi-protocol decoding
  - Enables decoding the same stream at multiple levels
  - Makes it easy to add higher level services
- Extensible statistics and logging subsystems
  - Makes it easy for OEM customers to customize the solution
- Central in-memory registry for configuration data
  - Makes it easy to change configuration dynamically

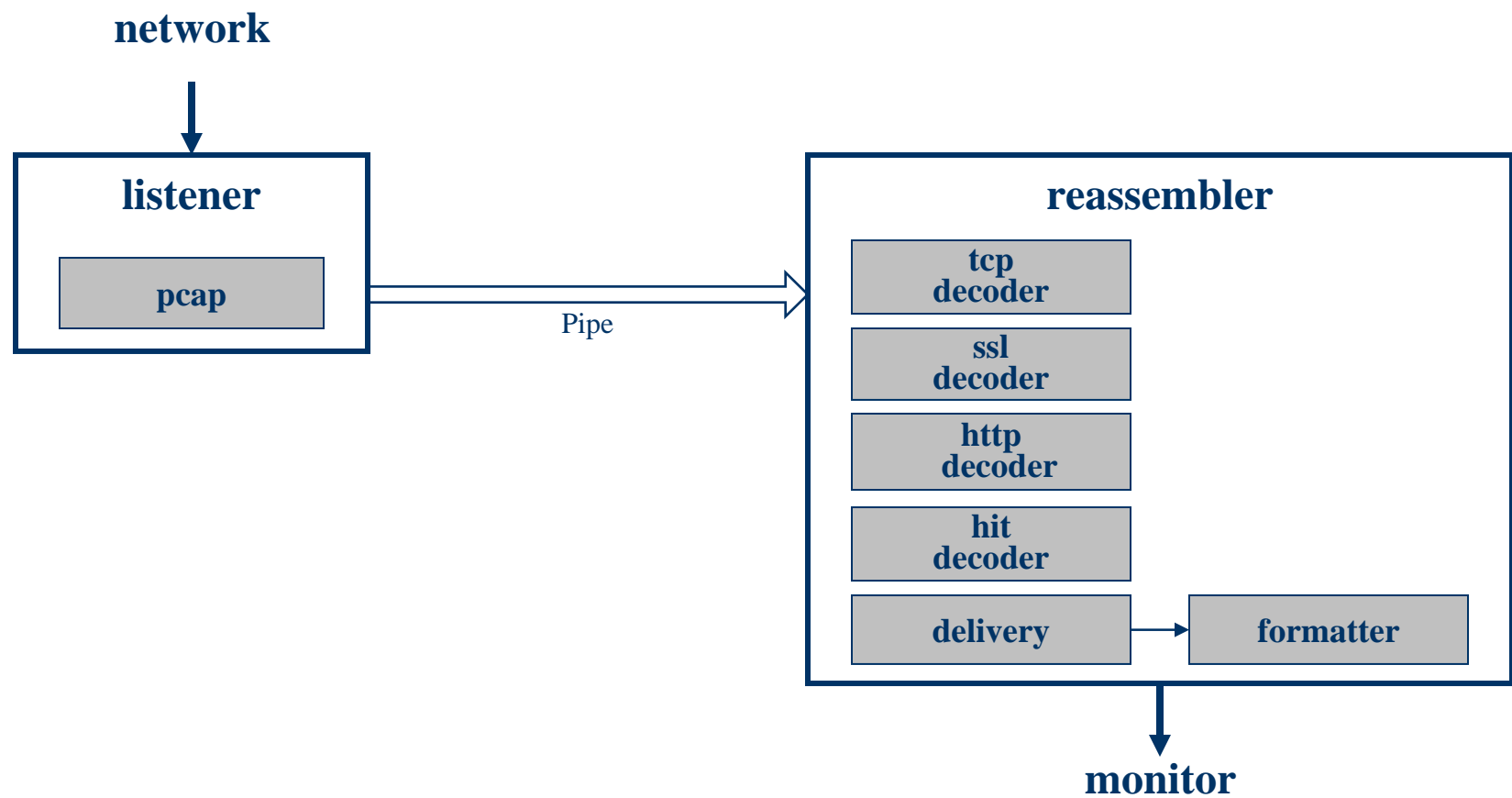
# Advantages

- *Time-to-market:* beta in as little as 1 month
- *Cost:* one time fee roughly equal to a burdened engineer-year
- *Typical Performance:*
  - 200Mbps sustained rate
  - 400Mbps burst rate
  - 2000 hits/sec of HTTP/HTTPS
  - 150 new (non-resumed) SSL handshakes/sec
- *Support:* experts on call to help you serve customers

# Technical Detail



# Decoder Modules



# Decoder API

- All protocol handling uses pluggable decoder modules
  - Decoder chains are dynamically configurable via registry
  - New decoders can be loaded from shared libraries
- Decoders can have multiple “next” decoders
  - For switch-hitting
  - Or data forking
- Decoders can have multiple flows
  - IP decoder has one flow (all the packets)
  - TCP decoder has many flows (multiple connections)

# Decoding a chunk

- Data comes down from the previous layer
  - E.g., a block of contiguous TCP data
  - Each chunk has a history
    - Which other decoders touched it (and their meta-data)
    - What flows (TCP connection, SSL connection, etc.) it's part of
- Options
  - Handle this data
    - Buffer
    - Modify
    - Inspect
  - Pass data to next decoder(s) (call `NR_decoder_propagate_chunk()`)
  - Drop this data (return 0)
  - Don't bother me for this flow again (return `R_REJECTED`)

# Configuration API

- All data stored in centralized registry
  - Each parameter is a keyword-value pair
    - E.g., `listend.primary.interface=/dev/eth0`
- Programs can access registry via
  - API
    - `NR_reg_get_string()` `NR_reg_get_uint4()`, etc.
  - HTTPS
    - Simple PUT/GET/DELETE interface
- Change management
  - Clients are notified via callbacks

# Statistics API

- Common statistics system
  - Statistics stored in shared memory
- Dynamically register new statistics types
  - Struct with the names of the fields
  - A vtbl to implement printing, clearing, etc.
- Each statistics element is named
  - `NR_stats_get("my_statistics_name",  
my_statistics_vtbl, NR_STATS_CREATE, &stats_ptr);`
- Generic utilities to manage stats
  - `nrstatstcl`
  - Automatically loads new stats libraries and does the right thing

# Logging API

- Looks like syslog
  - `r_log(LOG_FACILITY, LOG_ERR, "Error value is %d",r)...`
- Dynamically register new logging modules
  - `r_log("my_module_name", &MY_LOG_FACILITY);`
- Logging levels are individually controllable via registry
  - `logging.facility.my_module_name=5`
- Or via CLI/GUI
  - `Debug err my_module_name`

# Command Line Interface

- IOS-style interface
  - Accessible via SSH, console port, etc.
- All device functions can be managed this way
  - Configuration data is written into registry
  - Commands (reboot, ifconfig, etc.) handled directly
- Extensibility
  - New commands can be dynamically loaded
  - Just point it at the new shared library that implements the commands

# Integrating via Proxy

- You use our default hit delivery format
- We deliver hits to a proxy on your monitoring server
- The proxy converts our default format to your desired format

- 
- Pro: You can use our completely standard distribution
  - Pro: Easy to bridge different protocol styles
  - Con: Requires an extra server component

# Integrating via Custom Formatter

- The delivery module passes a raw hit buffer to the formatter
- The formatter passes back a formatted hit buffer
- You can customize the formatter to produce your desired format

- 
- Pro: You can make the format look exactly like what you use today
  - Pro: Minimum amount of change from standard distribution
  - Con: Cannot bridge protocol styles

# Integrating via Custom Delivery

- The delivery module receives hits from the last out module
  - Changes to the delivery module do not affect the modules above it
  - You write a custom delivery module that uses your internal protocol
- 
- Pro: You can use the same protocol style throughout your product
  - Pro: You still use the crucial parts of the standard distribution
  - Con: More programming

# Security of Private Keys

- To decode SSL, you need to give the device all servers' private keys
- These keys need to be available for automated restart
- Compromising the device therefore yields access to all the keys

- 
- Exactly the same situation as if you use an SSL-decoding IDS
  - All Web and application servers face the same challenge

- 
- Certified Data Trail provides alternative that protects private keys