# RAZORBACK™

VRT

SOURCEfire®

## Overview

What is Razorback?

## Razorback Is…

- An Open Source framework (GPLv2) to enable advanced processing of data and detection of events
- Able to get data as it traverses the network
- Able to get data after it’s  received by a server
- Able to perform advanced event correlation

- …Our answer to an evolving threat landscape

## The Challenge is Different

- Attacks have switched from server attacks to client attacks
- Common attack vectors are easily obfuscated
  - Scripting languages are infinitely variable
  - Compression obscures attack signatures
  - And more!
- File formats are made by insane people
- Back-channel systems are increasingly difficult to detect

## The Problem With Real-Time

- Inline systems must emulate the processing of thousands of desktops
- Detection of many backchannels is most successful with statistical evaluation of network traffic
- Deep file inspection requires too much time to process!

## Fill the Gap

- A system is needed that can handle varied detection needs
- A system is needed that extensible, open and scalable
- A system is needed that facilitates incident response, not just triggers it

# Architecture

What makes it tick?

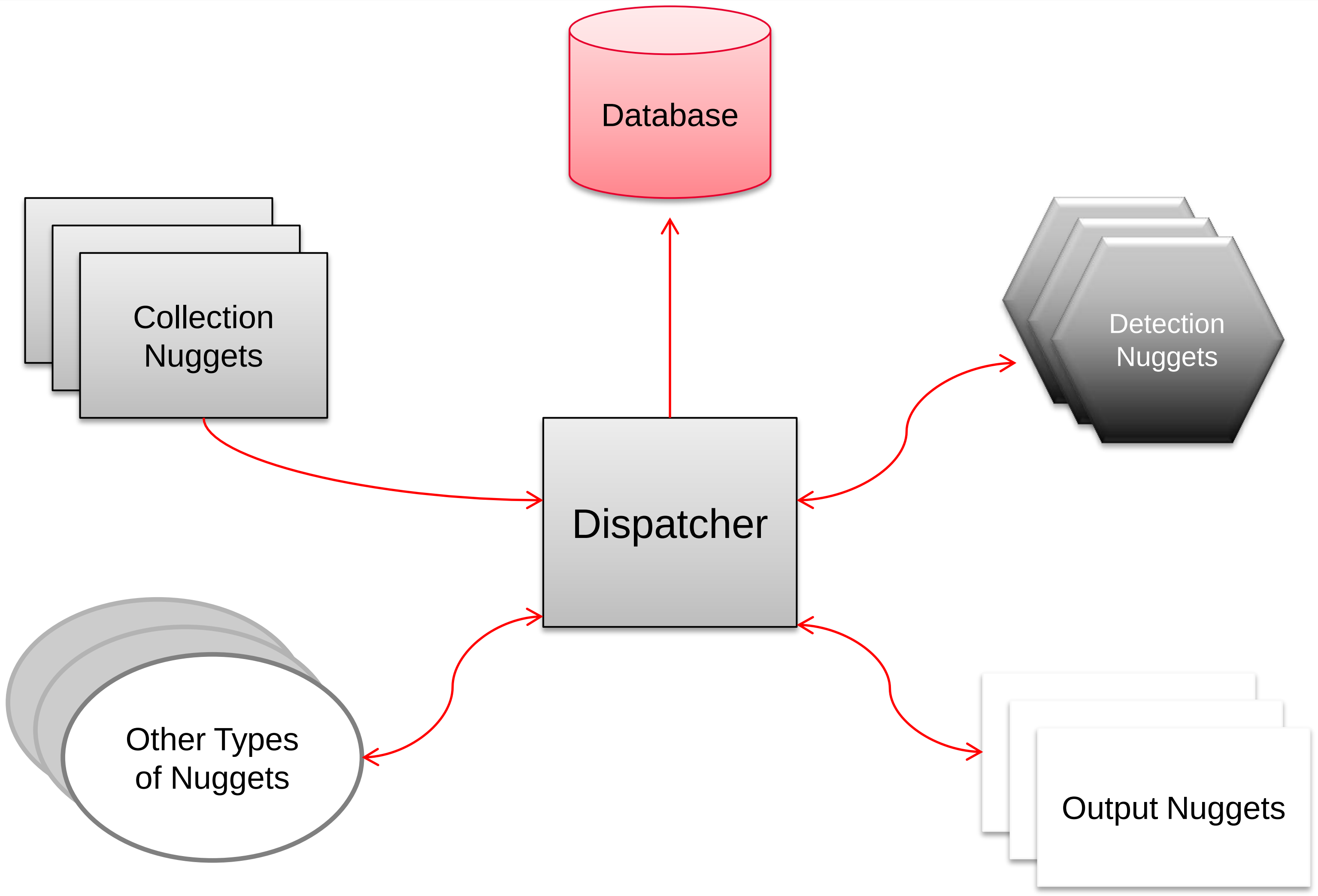
## Framework Goals

- Provide entry for any arbitrary data type
- Provide routing of input data to any number of relevant data processors
- Provide alerting to any framework-capable system
- Provide verbose, detailed logging
- Make intelligent use of all data

## Razorback is comprised of…

- A collection of elements working together
- Each element performs a discrete task
- Elements are tied together via the Dispatcher
- Nugget types:
  - Data Collection
  - Data Detection/Analysis
  - Output
  - Intelligence
  - Correlation
  - Defense Update
  - Workstation

# The System


Database
Collection Nuggets
Detection Nuggets
Dispatcher
Other Types of Nuggets
Output Nuggets

## The Dispatcher

- Handles all communication between nuggets
- Handles database interactions
- Database driven
- APIs are available for easy nugget development

## Database

- Configuration information
- Event information
- Contextual information
- Metadata
- Provides a wealth of information for correlating events and activities

# General Nugget Functionality

- Dispatcher Registration
  - Types of data handled
  - Types of output generated
- UUIDs
  - Identifier of nuggets
  - Type of nugget
  - Types of data handled and/or provided
  - Allows for easy addition and removal of elements

# Collection Nugget

- Capture data
  - From the network
  - From a network device directly
  - From log files
- Contact dispatcher for handling
  - Has this data been evaluated before?
  - Send the data to the Dispatcher

## Detection Nugget

- Handles incoming data from Collection Nuggets
- Splits incoming data into logical sub-blocks
  - Requests additional processing of sub-blocks
- Provides alerting feedback to the Dispatcher

## Output Nugget

- Receives alert notification from Dispatcher
- If alert is of a handled type, additional information is requested:
  - Short Data
  - Long Data
  - Complete Data Block
  - Normalized Data Block
- Sends output data to relevant system

## Intelligence Nugget

- Does not generate “alerts” per se
- Generates data that could potentially be used later for trending or event correlation

## Correlation Nugget

- Interacts with the database directly
- Provides ability to:
  - Detect trending data
  - Identify “hosts of interest”
  - Track intrusions through the network
  - Initiate defense updates

## Defense Update Nugget

- Receives update instructions from dispatcher
- Performs dynamic updates of network device(s)
- Update multiple devices
- Update multiple devices of different types!
- Notifies dispatcher of defense update actions

## Workstation Nugget

- Authenticates on a per-analyst basis
- Provides analyst with ability to:
  - Manage nugget components
  - Manage alerts and events
    - Consolidate events
    - Add custom notes
    - Set review flags
    - Delete events
  - Review system logs

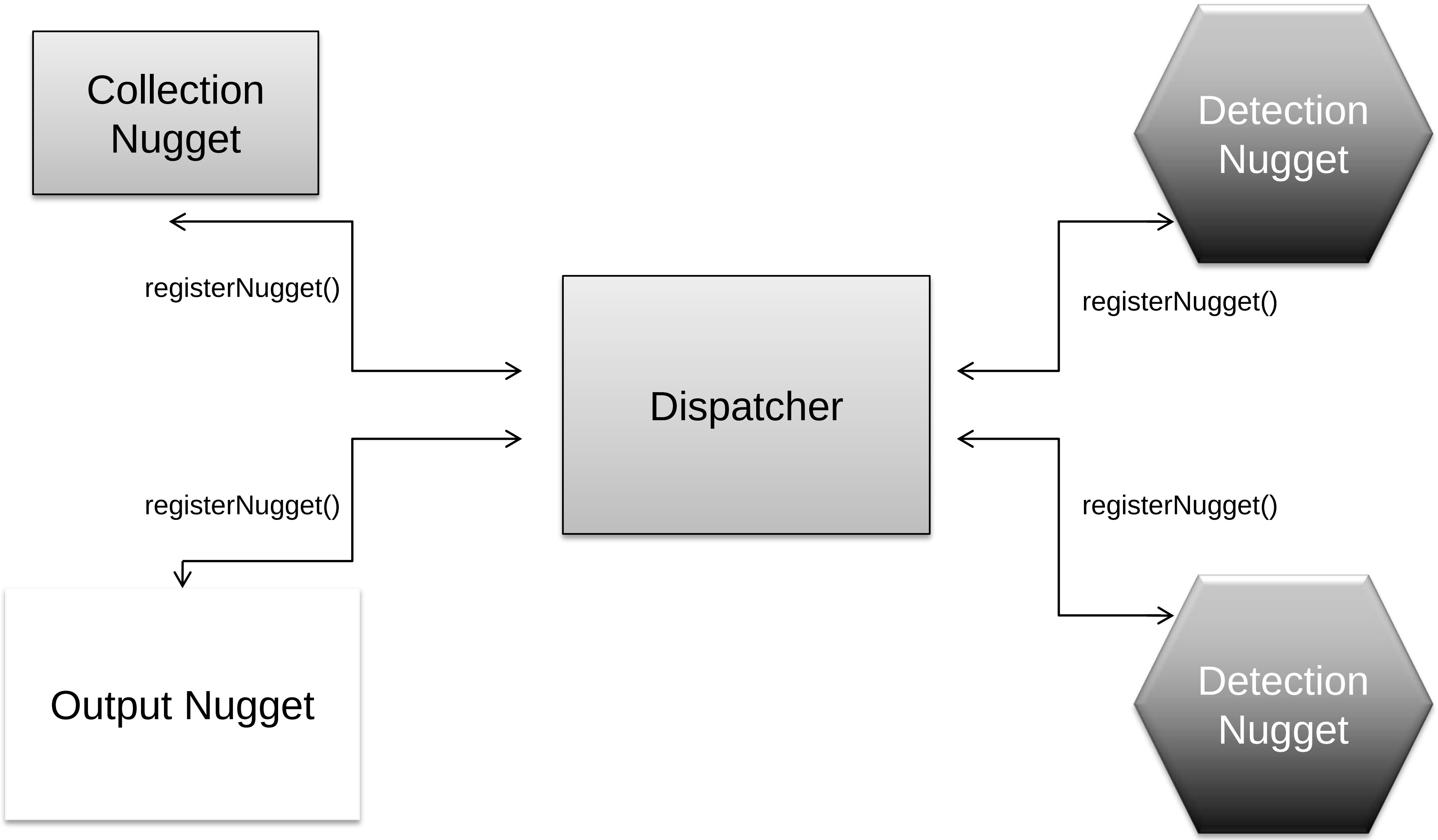
## Concept of Operations

How do they work together?

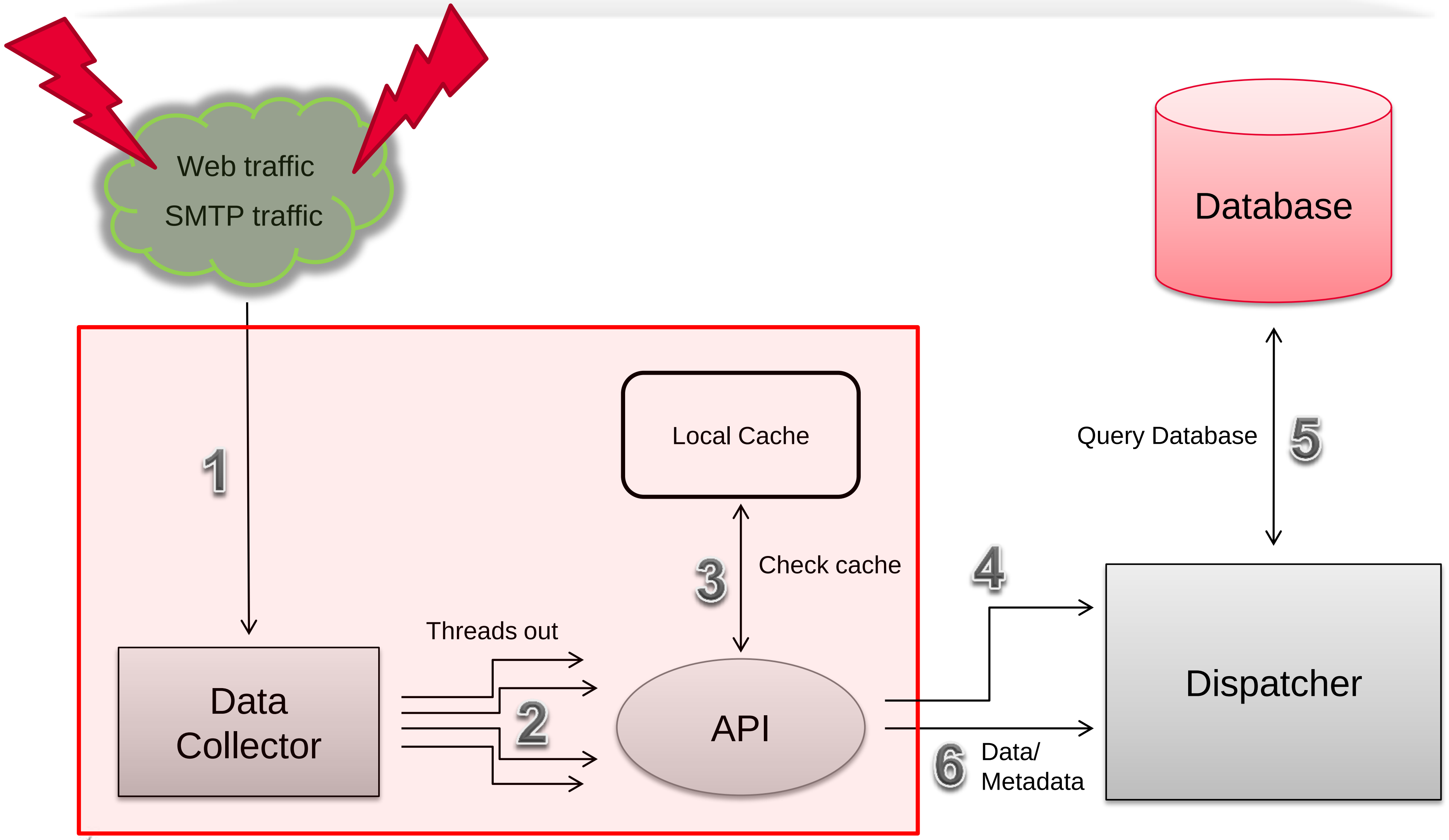
## Registration Phase

- Nuggets are brought online
- Nuggets register with the dispatcher:
  - Their existence
  - The data types they handle
  - How many threads they can run at once
- Dispatcher tracks via routing table
- Dispatcher hands back a unique “nugget id”

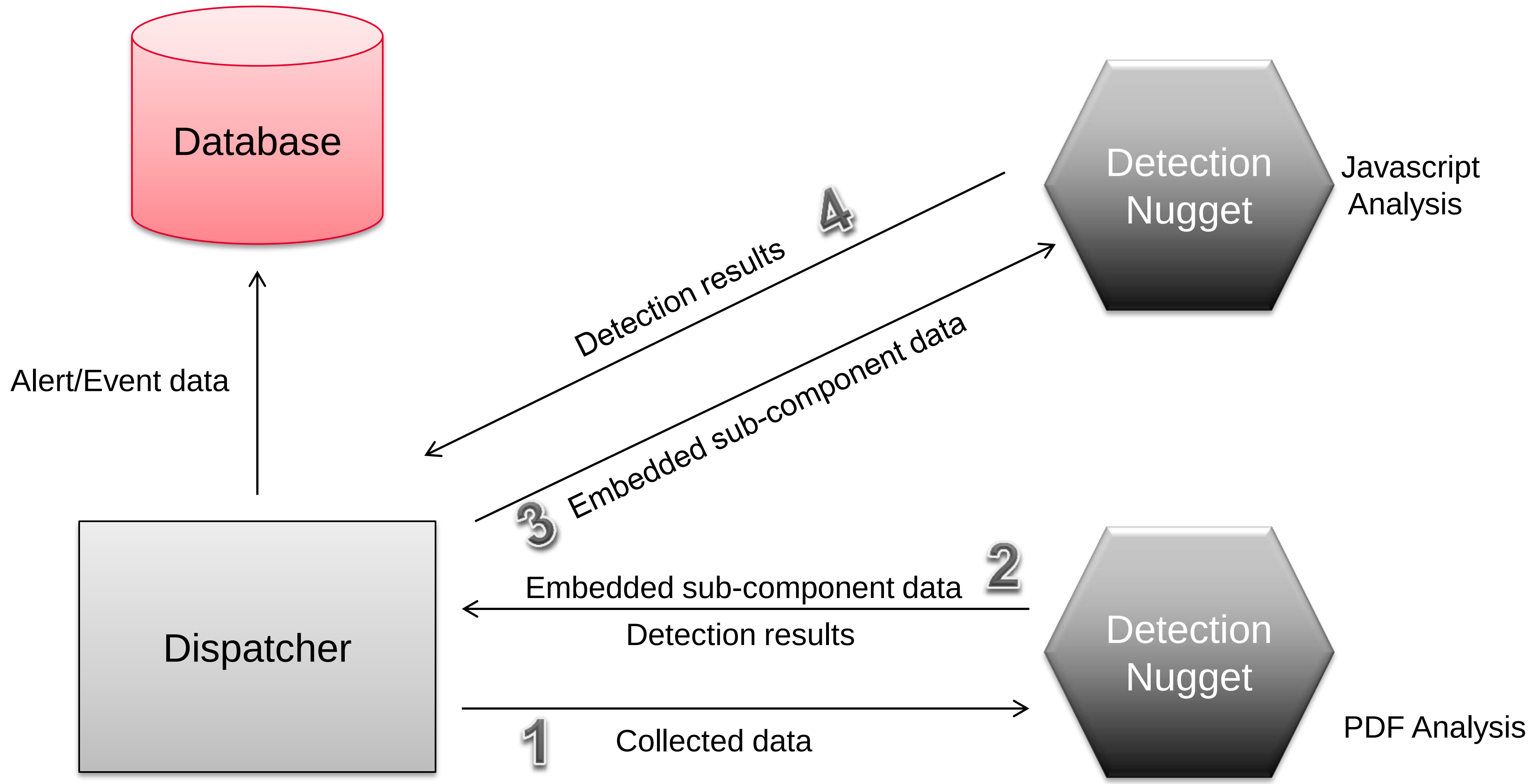
# Hi! I exist!


Collection Nugget
Detection Nugget
registerNugget()
registerNugget()
Dispatcher
registerNugget()
registerNugget()
Output Nugget
Detection Nugget

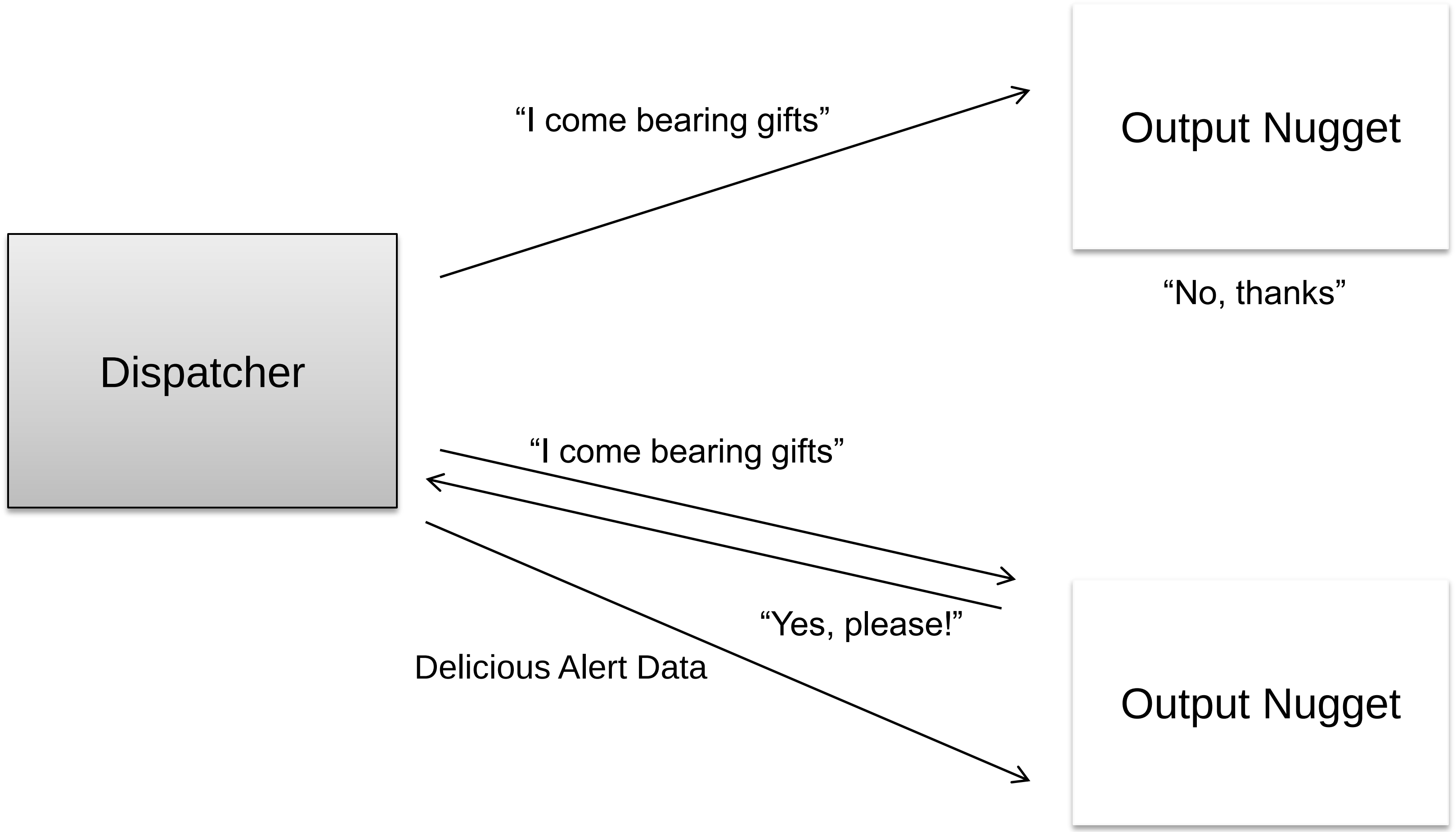
# Traffic comes in…



# Dispatcher farms out detection…



# Output nuggets are informed…



## Cache

- We want to avoid reprocessing files and sub-components we've already looked at
- MD5 and size are stored for files and subcomponents both bad and good
- But, after an update to any detection nugget, all known-good entries are thereby declared "tainted"

## Why Taint known good?

- Why taint known good?
  - Previously analyzed files may be found to be bad
- Why not just remove those entries?
  - We don’t want to rescan all files
  - If we see an alert for a previously scanned file matching the same MD5 and size, we can alert retroactively

## Case Study: SMTP

What happens when an email is received?

## Handling SMTP Traffic

- A PDF with a malicious embedded EXE is attached to an email
- How does the system work to tell us about this malicious attachment?
- Components in use
- Track the data

## Current Capabilities

Nuggets that are currently available.  Many more to come, and you can help!

## Collection Nuggets

- Snort-as-a-Collector (SaaC)
  - SMTP mail stream capture
  - Web capture
  - DNS capture
- Custom post-mortem debugger
  - Traps applications as they crash
  - Sends the file that triggered the crash to Dispatcher
  - Sends the metadata of the crash to the Dispatcher

## Detection Nuggets

- Zynamics PDF Dissector
  - Deobfuscation and normalization of objects
  - Target known JavaScript attacks
- JavaScript Analyzer (w/ Zynamics)
  - Search for shellcode in unescaped blocks
  - Look for heap spray
  - Look for obvious obfuscation possibilities

www.zynamics.com/dissector.html

## Detection Nuggets (cont'd…)

- Shellcode Analyzer (w/ libemu)
  - Detection and execution of shellcode
  - Look for code blocks that unwrap shellcode
  - Win32 api hooking
    - Determine the function call
    - Capture the arguments
  - Provide alerts that include shellcode action

libemu.carnivore.it

## Detection Nuggets (cont'd…)

- Office Cat Nugget
  - Full Office file parsing
  - Vuln-centric detection against known threats
- SWF Nugget
  - Decompresses and analyzes flash
  - Detects known flash threats

## Detection Nuggets (cont'd…)

- ClamAV Nugget
  - Analyze any format
  - Signature- and pattern-based detection
  - Updatable signature DB
  - Can further serve as a collector
  - Can issue defense updates

## Output Nuggets

- Deep Alerting System
  - Provide full logging output of all alerts
  - Write out each component block
  - Include normalized view of documents as well

- Maltego Interface
  - Provide data transformations targeting the Razorback database

www.paterva.com

## Workstation Nuggets

- CLI functionality to query:
  - Alerts, events, and incidents
  - Nugget status
  - Display metadata
  - Run standardized report set

## Programming Interfaces

How are nuggets created?

## Custom API

- API provided for easy creation of nuggets
- The API provides functionality for:
  - Registering a new nugget
  - Sending and receiving data
  - Cache and database interaction
- Threading is handled automagically!

## General Functions

- registerNugget()
  - Type of nugget
  - Type(s) of data handled
  - Connection information
- registerHandler()
  - Specifies handler function
  - Type(s) of data handled for that function
  - Can register multiple handlers per nugget

## Collection and Detection Nuggets

- sendData()
  - Sends captured data to the dispatcher
- sendMetaData()
  - Adds any additional information about the collected or parsed data
- sendAlert()
  - Specific alert data to be sent to Output Nuggets

## Intelligence Nuggets

- Functions provide access to modify database
- Types of Intelligence Nuggets supported:
  - Email
  - Web
  - DNS
- Easy to add new protocols
  - Create database schema
  - Provide function for accessing that schema

## What if I don’t like C?

- Nuggets can be written in any language
- Wrappers providing interfaces to the API functions are provided
  - Ruby
  - Python
  - Perl
  - If you can wrap C, you can create an API

## Conclusion

Let’s wrap this up!

## Razorback Framework…

- Extensible. Open. Modular.
- All functions are separated and distributed
- Core is written in C, APIs available for other languages as well
- Limitless possibilities!

## This is great!  How can I help?

- See a need for a nugget?  Write one and send it in!
- Full source code available on Sourceforge
  - http://sourceforge.net/projects/razorbacktm
  - http://sourceforge.net/projects/nuggetfarm
- Bug tracking via Sourceforge Trac

# Questions??

- Patrick Mullen
  - pmullen@sourcefire.com
  - phoogazi on Twitter
- Ryan Pentney
  - rpentney@sourcefire.com
- Sourcefire VRT
  - labs.snort.org
  - vrt-sourcefire.blogspot.com
  - VRT_Sourcefire on Twitter

Razorback Team:

Alex Kambis
Alex Kirk
Alain Zidouemba
Christopher McBee
Kevin Miklavcic
Lurene Grenier
Matt Olney
Matt Watchinski
Nigel Houghton
Patrick Mullen
Ryan Pentney
Sojeong Hong