Razorback Project Charter

Razorback 0.2 Project Charter

*0.2*

Final

3/3/2011

Table of Contents

[System Summary 3](#_Toc282685345)

[System Management 3](#_Toc282685346)

[Kool-Aid Target Impact 3](#_Toc282685347)

[Customer is not stupid 3](#_Toc282685348)

[Be Nice to the Customer 3](#_Toc282685349)

[Speed and Scalability 3](#_Toc282685350)

[Component Detailed Design 4](#_Toc282685351)

[Dispatcher 4](#_Toc282685352)

[Database 6](#_Toc282685353)

[API 8](#_Toc282685354)

[Design Deficiencies 8](#_Toc282685355)

[Long Term Strategy 9](#_Toc282685356)

Table of Figures

[Figure 1: Component Chain for Malicious PDF 5](#_Toc282685315)

[Figure 2: Alert Database Structure 7](#_Toc282685316)

# Project Mission/Purpose

Razorback 0.2 should lay the ground work for all future releases in 2011. The main goals for this release are fixing configuration management, locking the SVN source code directory structure, fixing the internal automake configuration, selecting a queuing mechanism, developing version 1 of the queuing support, and updating the communication architecture between components to support the queuing mechanism.

# Supported Platforms

## Primary Open Source

* Ubuntu 10.4 (32/64)
* FreeBSD 8.1 (32/64)
* Debian 5.0 (32/64)
* RHEL 6.0 (32/64)

## Primary

Primary platforms are the main development and testing platforms for Razorback 0.2. All Razorback components should build and pass all unit tests on these platforms. Additionally, these are the core platforms that all QA testing and functionality verification should be conduct on.

## Secondary

Currently there are no secondary semi-support platforms. In the future there may be platforms where Razorback is known to compile, but its unit test and core functionality isn’t tested or verified by Sourcefire.

## Tertiary

Currently there are no known Tertiary platforms that Razorback has been externally ported to and are maintained by parties external to Sourcefire. In the future this section should list those platforms and if their user counters increase should be considered for Secondary or Primary status.

# Supported Compilers

There are many compilers and many versions of those compilers in use across numerous platforms. This section outlines the compilers that are used to build Razorback internally, which ones are known supported, which ones aren’t known supported, and which ones aren’t supported at all.

## Primary

Razorback needs to compile with these, because they are needed for the Commercial and Primary Open Source platforms.

* gcc version 4.2 and newer
* gcc version 4.1.2 20071124 (Red Hat 4.1.2-42)

## Secondary

These compilers are common and Razorback should compile with them. We will not attempt to work around bugs in them. If a certain version is found to be seriously bugged, it'll be blacklisted.

* gcc 3.4.6

## Tertiary

Any compiler that provides an IOS C/POSIX environment and that is not listed above or blacklisted is considered a tertiary compiler.

## Blacklisted

If a compiler is known to produce invalid code for Razorback or its components it should be listed here.

* None Known

# Requirements

## Packaging

1. Layout new SVN directory hierarchy
2. Move Razorback 0.1 code into new directories
3. Fix autoconf for building
4. Add top level autoconf for packaging
5. Layout install directories
6. Update install for new directory layout

For additional information, see the Razorback Developer Guide.

## Real-Time Dispatcher Services

1. Provide an admin interface to the system
2. Provide a registration interface to authenticate and uniquely track nuggets
3. Provide a dedicated caching service to support high-latency blocking and to minimize duplicate submissions
4. Provide an alerting listener to handle returns from detection nuggets

## Security

1. Encryption, as an option, for all network traffic that can reveal data block information, interpretation or forensic data
2. Authentication of nuggets and unique identification of nuggets
3. Authorization checks at the earliest point possible to provide protection from information disclosure or submission bombing

## Networking

1. Support IPv6 on all network services
2. Encryption, as an option, for all network traffic that can reveal data block information or interpretation or that will reveal forensic data
3. Authorization at the earliest point possible to provide protection from disclosure or submission bombing

## Administration

1. Support queue visibility
2. Provide methods to update authentication information for nuggets

## Message Queue initial implementation

1. Support for horizontally scaling message queues
2. Replacement of routing table with sorted queue tree
3. Dynamic creation of queues as appropriate to support revised routing
4. Update all components to pull from and push to the queues
5. Pool processing (one of many)
6. One-to-many messaging
7. Queue to disk on failure support
8. Ensure proper load balancing where appropriate

## Database improvements

1. Store data blocks to a searchable non-SQL solution
2. Modify schema to support internationalization (UTF-8)
3. Move to a Nugget UUID – timestamp duple for unique event identification
4. Update tables supporting the intel activity to minimize duplicate data, particularly in user agent, hostname and URL fields.
5. Add triggers to manage Known Good/Known Bad notifications
6. Modify schema to support long-term pending requests

## Nugget registration

1. Move to a master Nugget for each nugget type
2. Rewrite nugget registration process
3. Modify identification of nuggets to persistent UUIDs

## API Modification

1. Have the API always send a “Known Good” message at the end of detection, unless it is known that a “render verdict” packet has already been sent
2. Provide API call to set detection to “long term pending”
3. Provide API call for nugget to retrieve list of outstanding pending requests

## Quality Assurance

1. Create initial regression testing test cases.

## Open Source Beta

1. Notification of 0.2rc availability should be given to the Razorback mailing list.

## Release Notes

TBD