Maintaining and Securing Kerberos

Tom Yu

MIT Kerberos Consortium

March 30, 2009

Overview

Completed krb5-1.7 goals

Rethinking the release cycle

Guiding principles

Completed krb5-1.7 Goals

Enhanced GSS-API error messages

Cross-platform CCAPI (Mac and Windows)

Kerberos Identity Management (KIM) API

Master key rollover

Incremental propagation

Enctype negotiation

Lots of Microsoft protocol extensions

FAST (in progress)

Rethinking the Release Cycle

Originally: 18-month cycle

Proposed:

- 6–12 months
- New features available sooner

Consensus of core developers

Precedent: Ubuntu, GNOME, et al.

Advantages of Shorter Cycle

Less scrambling to include features in release

Release features only when ready

Improved quality

Disadvantages of Shorter Cycle

- Longer maintenance lifetimes
  - Security patch implications
- Possibly fewer features per release
- User/vendor reluctance to track releases

Topic Branches

Develop new features or “topics” on branches

Keep branch synchronized with trunk

Integrate branch into trunk when ready

Feature development can span releases

Lightweight branch capability is essential

New Version Control System?

- Subversion (existing)
  - Branches are heavyweight
  - Weak merging support
- Git
  - Branches are lightweight
  - Better merging support
- Migration is costly
  - Data model mismatches
- Hybrid approach: git-svn

Implementing Shorter Cycles

Target dates, not release numbers

Feature readiness determines release dates

Guiding Principles

Code quality

Modularity

Credential management

End-user experience

Administrator experience

Performance

Protocol evolution

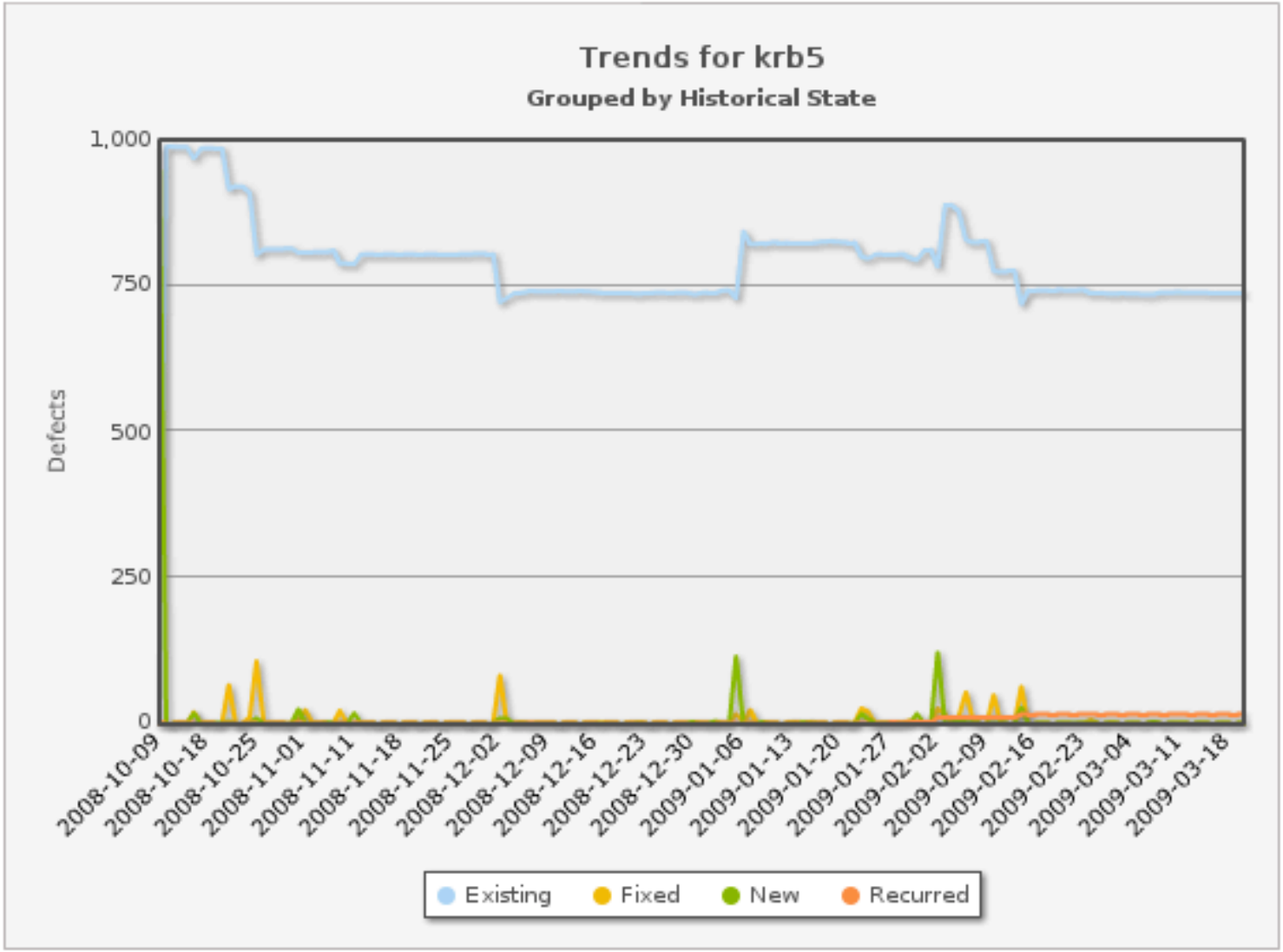
Code Quality

- Remove krb4 (done)
- Use safer library functions (ongoing)
  - Avoid false positives
  - Avoid need to validate “unsafe” calls
  - Stop using strcpy, strcat, sprintf, etc.
    - Mostly done
    - New internal APIs for complex operations
- Reduce commitment to “difficult” platforms (done)

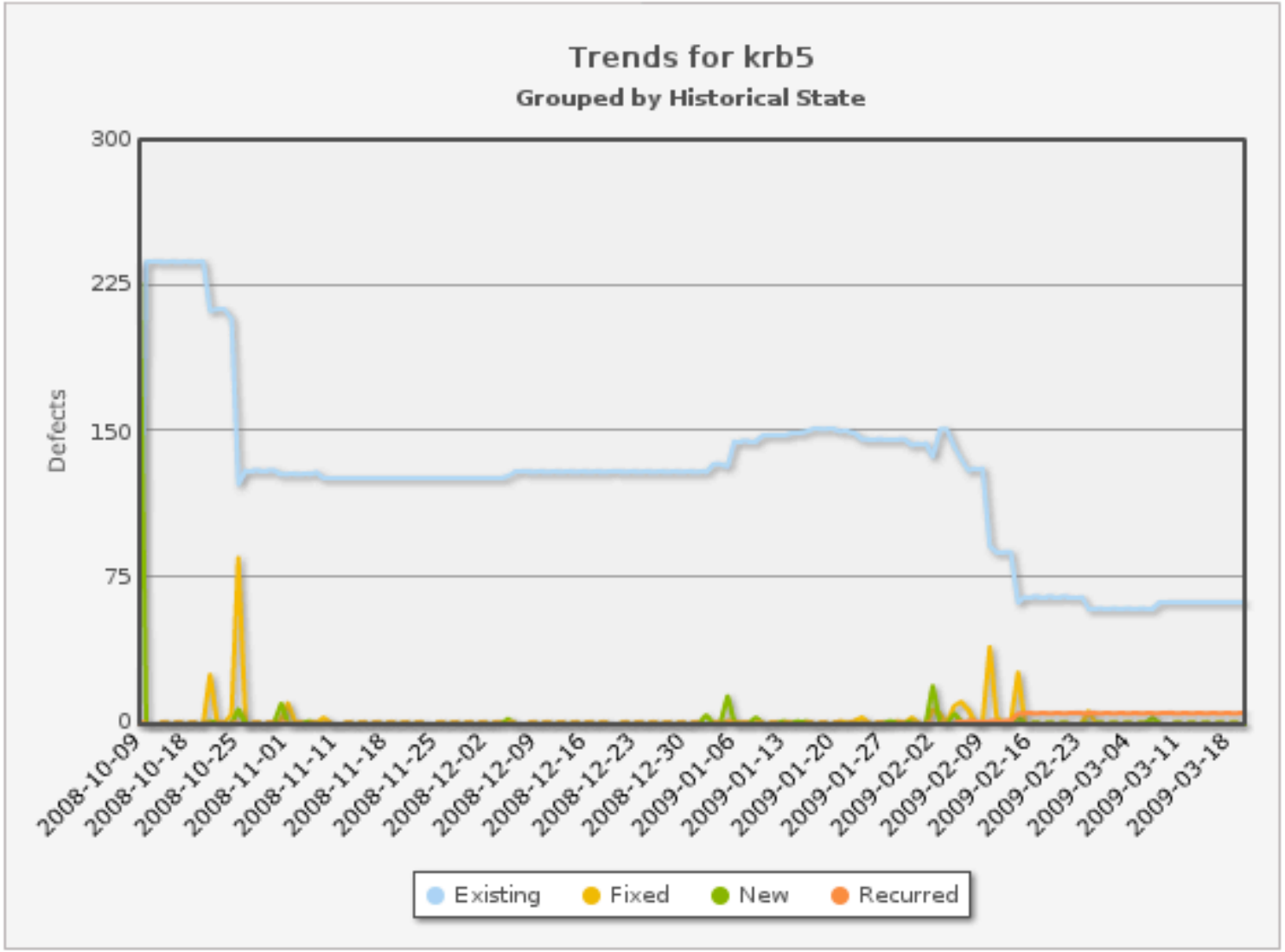
Code Quality (cont'd)

- Move toward test-driven development
  - Existing test suite is fragile
  - Wholesale rewrite is expensive
  - Require test strategy for new code
  - Incrementally introduce stricter testing
- Coding style
  - Existing inconsistency
  - "Fixing broken windows"

Coverity Defects



Coverity Defects (libkrb5)



Modularity

- New crypto API
  - Native (accelerated) crypto API support
  - Performance optimizations (caching, etc.)
- Support readily building subsets
  - “Lite” client – mobile devices
  - “Lite” server
  - GSS-API: context estab. vs msg. protection
    - e.g. Solaris user/kernel space split

Modularity (cont'd)

- GSS-API mechanism glue
  - At least rough form for NTLM support (done)
  - Possible refinements later
- KDC Database (long-term)
  - Track IETF data model work
  - New API around 1.8
  - New implementation around 1.9
- Secure co-processor ("would be nice")

Modularity: Crypto API

- Vendor interest in FIPS 140-2 compliance
- Working with NIST on clarifications
  - FIPS 140-3
  - SP 800-57 part 3
  - SP 800-63 rev. 1
- OS vendor crypto APIs
- PKCS#11, OpenSSL?
  - Explore using a common abstraction

End-user Experience

- Enhanced error messages for GSS-API (done)
- Credential management
  - KIM API (done)
  - Cross-platform CCAPI
    - Done for Mac & Windows
    - UNIX implementation (1.7+)
- Referrals (in progress)
  - DNS independence via referrals
- Localization of static error strings (1.7+)

Administrator Experience

- Incremental propagation (done)
- Improve key rollover
  - Master key (done)
  - Application service keys (1.8)
- Audit support
  - Log all ticket requests (done)
  - BSM etc.
- Disable DES by default (in progress)

Performance

- Decrease DNS traffic (1.7+)
  - Stop trying to crawl up to the root
- Replay cache (“rcache”)
  - Disable on KDC (1.7+)
    - Avoid known false-positive issues
  - Collision avoidance (done)
  - Improve implementation (1.7+)
  - Disable by service type name (1.7+)
- New crypto API (1.8) facilitates optimizations

Protocol Evolution

Encryption algorithm negotiation (done)

Microsoft Kerberos extensions (done)

Improved PKINIT support (1.7+)

Anonymous PKINIT (1.8)

FAST (in progress)

International strings in protocol (1.8+; IETF)

Timestamp-independence (1.8, 1.9)

Replay-proofing protocols (1.8, 1.9)

Questions?

Supported Platforms

- Mac OS X
  - “Darwin” command-line build
- GNU/Linux (OS family)
  - Currently Debian, Ubuntu, or Red Hat on x86_64 and x86
- Solaris (SPARC or x86_64/x86)
- BSD (OS family)
  - Currently NetBSD on x86_64 and x86

Process Changes

- Streamline project proposal process
- Community resources
  - Wiki for developers – k5wiki.kerberos.org
  - Source browsers – OpenGrok, FishEye
  - White papers, tutorials, best practices
- Incrementally adopt style, review guidelines
- Improve testing infrastructure
- Analysis tools
  - Coverity, compiler warnings (static)
  - Valgrind, Purify (runtime)

Interface Change Strategy

- Crypto, KDB, etc.
- Incremental, staged approach
  - Design new interface
  - Upper layer on new interface
    - Implement new interface on top of old
  - New lower layer
  - Compatibility interface on top of new interface
    - If needed