MIT Kerberos Releases: new and upcoming

Tom Yu

MIT Kerberos Consortium

October 21, 2009

Overview

- Timeline
- Guiding principles
- Rethinking the release cycle
- Release 1.7
- Release 1.8
- Future directions

Timeline

- Target 9-month cycle
- krb5-1.7
  - Released Jun. 2009
- krb5-1.8
  - Jan. 2010 – feature freeze and release branch
  - Mar. 2010 – final release
- krb5-1.9
  - Sep. 2010 – feature freeze and release branch
  - Dec. 2010 – final release

Guiding Principles

- Code quality
- Modularity
- End-user experience
- Administrator experience
- Performance
- Protocol evolution

Rethinking the Release Cycle

- Originally: 18-month cycle
- Proposed:
  - Approximately 9-month cycle
  - New features available sooner
- Consensus of core developers
- Precedent: Ubuntu, GNOME, et al.

Advantages of Shorter Cycle

- Less scramble to cram new 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
- No repository change for now

Implementing Shorter Cycles

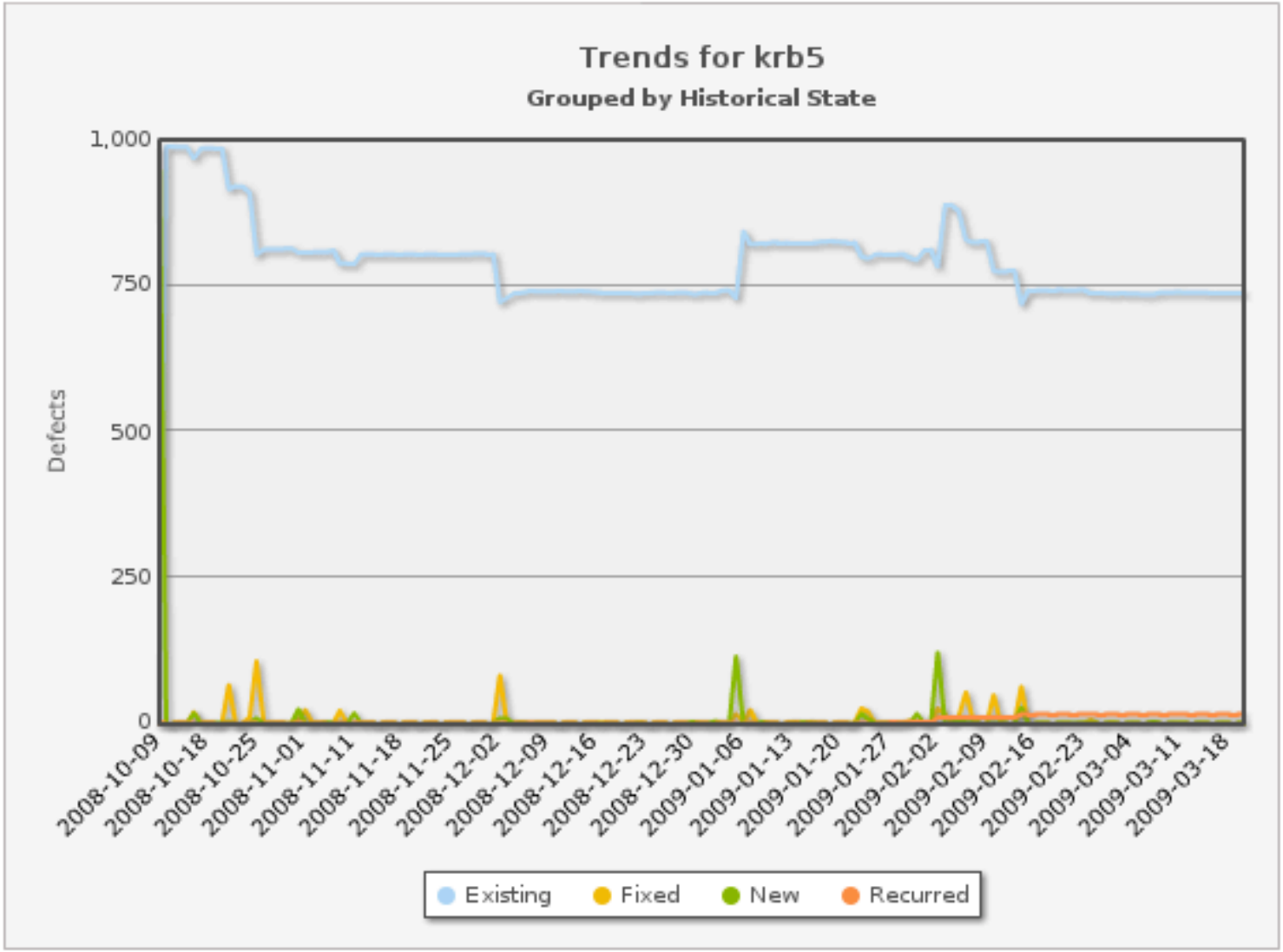
- Target dates, not release numbers
- Feature readiness determines release dates

Release 1.7

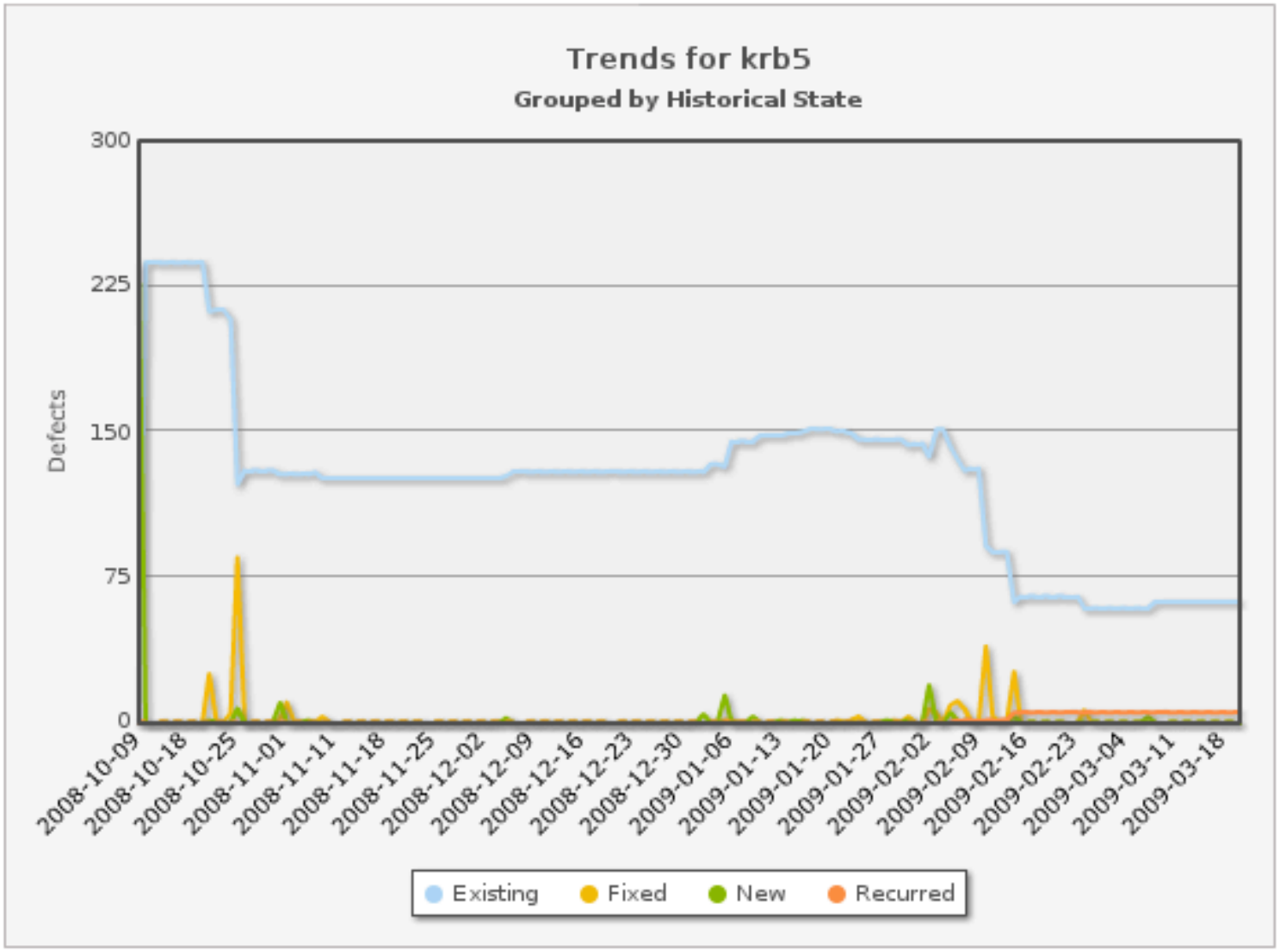
1.7: Code Quality

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

Coverity Defects



Coverity Defects (libkrb5)



1.7: End-user Experience

- Enhanced error messages for GSS-API
- Credential management
  - KIM API
  - Cross-platform CCAPI
    - Done for Mac & Windows
- Service principal referrals

1.7: Administrator Experience

- Incremental propagation
- Master key rollover
- Audit support
  - Log all ticket requests (done)

1.7: Performance

- Replay cache (“rcache”)
  - Collision avoidance

Protocol Evolution

- Encryption algorithm negotiation
- Microsoft Kerberos extensions
- FAST

Release 1.8

1.8: Code Quality

- Move toward test-driven development
  - Staff: scripts for testing new code
  - Contributors: manual testing procedures
- Increase conformance to coding style
  - Whitespace cleanup
  - Reindenting
  - Selective refactoring

1.8: Modularity

- Crypto modularity
  - FIPS 140 modules
  - Native (OS vendor, etc.) crypto
    - OpenSSL (in progress)
    - PKCS#11 / NSS (time permitting)
  - Hardware acceleration
- Improved API for authorization data

1.8: Performance

- Encryption performance
  - Cache derived keys
  - Later: enable using non-extractable keys
    - Hardware tokens
    - FIPS 140 modules
    - etc.
- Investigate other reported performance issues

1.8: End-user Experience

- Reduce DNS dependence
  - Aux. data in ccache: track whether KDC supports service principal referrals
  - Requires additional protocol support

1.8: Administrator Experience

- Disable single-DES by default
  - DES is not very secure anymore
  - Future releases may remove DES entirely
- Improved enctype configuration
  - Explicitly configured enctype lists caused problems
- Lockout for repeated login failures
  - Satisfy regulatory requirements
- Trace logging for easier troubleshooting

1.8: Protocol Evolution

- FAST enhancements
  - FAST negotiation for ease of migration
- Anonymous PKINIT
  - Allows for easier host key establishment
- Services4User (S4U) enhancements in GSSAPI
  - Application server API for performing S4U operations

Future Directions

Future Directions (a sampling)

- Code quality
  - More integrated automated testing
- Modularity
  - GSS-API user/kernel split
- Performance
  - Concurrency enhancements
- Protocol evolution
  - Newer algorithms
  - Use modern cipher modes (CCM, GCM)

Questions?

Supported Platforms

- Will be influenced by sponsor input
- Test OS families on limited platforms
  - …but work with others for related platforms
- Mac OS X (“Darwin” command-line build)
- GNU/Linux (OS family)
  - Debian, Ubuntu, or Red Hat (x86_64 and x86)
- Solaris (SPARC, x86_64/x86)
- BSD (OS family)
  - NetBSD (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