BennuProposal BennuFaq

Bennu Frequently Asked Questions

Author: Daniel S. Haischt <<MailTo(dsh AT apache DOT org)>>

Date: December 2007

Online version: http://wiki.apache.org/incubator/BennuProposal/BennuFaq

***Status:* .**

${renderedContent}

FAQ Item List

We would like to encourage everyone to contribute new FAQ items to this list. Especially if you think you got a question that is quiet common and may be raised by several people independently.

General Items

Q: *The provided list of Bennu envisioned or proposed features is already quiet extensive. Don't you think this is way too much for the time being?*

A: No. We are seeking for a long term relationship with the ASF and thus we are proposing Bennu to the Apache incubator. Frankly speaking, maintaining the current feature set of m0n0wall or FreeNAS isn't much of an effort (Note: FreeNAS was completely ported to pfSense within about three month, having a K-LOC of 185.687). We think providing a rich set of ideas may assist in attracting a diverse community of developers towards Apache and the Bennu project. Thus the comprehensive amount of proposed features.

Operating System Related Items

Q: *Will Bennu re use a particular type of operating system?*

A: We are trying to provide an operating system agnostic management layer for software service that can be found on particular operating systems. Tho initially we will focus on a FreeBSD compliant management layer.

Q: *Why have you chosen to initially focus on FreeBSD?*

A: Bennu will be re using the m0n0wall code base as an initial starting point. m0n0wall is FreeBSD based and thus the Bennu management layer will be initially FreeBSD compliant as well.

Q: *Isn't it a real overhead to try to accomplish an OS agnostic management layer?*

A: To a certain degree Yes. On the other hand with the help of appropriate software design (like service abstraction and a polymorphic service behavior), POSIX complaint coding guidelines and the appropriate tooling (autotools or cmake etc.) we do believe that such an OS agnostic software service management layer can be accomplished or at least porting the layer to a different OS will be easier.

Q: *So is it correct that Bennu will be composed of an operating system (e.g FreeBSD) plus a management layer to manage software services of an operating system?*

A: No. The Bennu system (as it will be maintained in a SCM system) will only consist of the aforementioned management layer plus an appropriate build system that is assembling a custom operating system, suitable to be run either on an embedded device or a server appliance. The outcome will be a Bennu image that contains anything that is required to run the Bennu system on an embedded device or a server appliance.

Hardware Related Items

Software Related Items

Q: *Which programming language will be used to implement the Bennu software service management layer?*

A: C/C++

Q: *Which interfaces will be exposed by the management layer to enable other software systems to interact with the Bennu system?*

A: Some particular implementation of a web service protocol (either SOAP based or REST or even both).

Q: *In the hardware section above you've mentioned that there will be some kind of a build system that is coming along with the Bennu (source code) distribution. What's the particular purpose of such a build system?*

A: Basically it's more than impractical to require the end user to assemble a ready to run Bennu operating system image on his own. Thus a Bennu build system will be put into place that is responsible for *a)* compiling the Bennu software service management layer *b)* creating a directory tree structure that will compose the actual Bennu runtime system *c)* configuring the Bennu boot loader etc. and *d)* creating an installable software image that contains the Bennu binaries plus an accompanying operating system. The installable software image will be geared towards a specific hardware platform (e.g. embedded or server).

Q: *You've mentioned a so called software service management layer that is being provided by Bennu several times. What's the exact purpose of this management layer?*

A: *a)* Management of software service that may be hosted by an operating system such as the operating system's firewall rule processing engine or 3rd party software service such as a HTTP proxy implementation and *b)* providing an interface to management function that may be re used by a GUI to administer software services.

Q: *Hummm ... Sounds like Bennu is trying to accomplish what others like Webmin are already providing.*

A: No, not exactly. Bennu is providing an abstraction layer that allows to administer concrete software services. The abstraction will be fine grained to allow to replace a software service that implements for example a HTTP proxy by another software service implementation having the same purpose. Such a software interchange scenario may be required if a customer increases the QoS criteria for a HTTP proxy software service implementation where software service *A* does not fulfill these kind of criteria any more and thus must be exchanged by software service implementation *B*. Furthermore we are envision to provide software services which are proxying the actual service implementation. That way it may be possible to off-load the actual service implementation to another hardware device. Finally Bennu does not provide a mandatory web based GUI as it is implied by Webmin. The web based GUI may be deployed as an add-on to the Bennu runtime environment. Although it would be also possible to administer several Bennu runtime environments using a centralized management facility and thus the web based management GUI on each Bennu runtime environment becomes obsolete.

Q: *How will the software service management layer persist configuration settings of particular software services?*

A: The configuration of the complete Bennu system will be stored to a DBMS, flat files such as XML or INI files, a directory service such as LDAP or even to the native format of a software service implementation. Choosing the later will be a valid choice for example if you want to edit the settings of a software service on the command line manually using an editor. Reading and storing the configuration settings of certain software services will be provided by pluggable config stores.

Q: *Will it be possible to migrate configuration settings between different config store implementations?*

A: Yes.

Q: *Will it be possible to import configuration settings from a m0n0wall or pfSense systems?*

A: Yes. Basically migrating from m0n0wall or pfSense to Bennu will be supported. If Bennu doesn't support a certain function which was available on m0n0wall or pfSense it wouldn't be possible to migrate the corresponding configuration settings for obvious reasons.

Q: *It sounds like the software service management layer is just providing the building blocks for managing software services which may be found on an operating systems (e.g. UNIX daemons etc.). Does this mean this management layer may be used to assemble a custom NAS or PBX system based on Bennu?*

A: Yes. This has been already done using m0n0wall as a foundation (e.g. FreeNAS and Askozia are both m0n0wall based) and thus, because Bennu shares the same idea of providing re usable components, it would be as well possible to create a PBX or NAS system by using the Bennu software artifacts. Tho we currently put our focus on firewalling/routing, Wi-Fi and unified threat management.

Q: *I am used to managing my router devices using a command line shell or a character based GUI (e.g. curses). Will Bennu provide such a shell or even a character based GUI?*

A: This has been envisioned as a possible field of investigation but at the time there are no concrete plans on implementing such administration facilities.

Q: *IDS and IPS systems do require a certain amount of computing power. How are you plans to support this particular kind of software on embedded devices?*

A: Each installable software service that may be installed on a Bennu runtime environment will provide a certain degree of meta information which includes data about for which system the service was originally designed (e.g. may *NOT* be run on embedded devices). Thus the user will get a certain degree of decision guidiance about whether a particular service is suitable for an embedded device or not. At the end of the day it's upon the user to make a decision to install an IDS/IPS on an embedded device or not.