Infra-estrutura (FreeBSD) Unix no (Mac) OS X

Autoria de P. Tracanelli (FreeBSD Brasil)
06/03/2012
Última Atualização 06/03/2012

No dia 01/03 o Renato me convidou pra escrever um pouco da relaÃ§Ã£o Unix-BSD-OSX, pra sÃ©rie de artigos de seguranÃ§a sendo divulgados pela IDS Tecnologia na MacMagazine. Escrevi um artigo um tanto extenso, que foi condensado propriamente ao ser publicado na MacMagazine (clique pra ver), e partes dele serÃ£o reutilizados ao longo dos demais artigos. No entanto em particular tive pedidos pelo artigo na Ã-ntegra, entÃ£o segue ele postado aqui na FUG tambÃ©m, espero que gostem :-)Â

Pessoalmente gosto muito desse trecho da histÃ³ria dos BSD em geral e acaba ilustrando como a guerra jurÃ-dica que o CSRG/Berkeley sofreu por parte da USL/AT&T quando a segunda processou Berkeley por conta dos 6 arquivos AT&T restantes no BSD Unix. LÃ³gico que pro mundo BSD foi uma passagem terrÃ-vel, retardou a adoÃ§Ã£o de sistemas BSD e liberdade Open Source do cÃ³digo BSD. Mas teve seus lados positivos, como Torvalds e seu kernel baseado no Minix quando ele ficou inseguro ao usar o 386BSD, e nessa passagem outro ponto positivo, a criaÃ§Ã£o do Mach pela universidade de Carnegie Mellon como uma alternativa ao BSD sob base BSD, posteriormente aproveitados no NeXT Step.

Segue entÃ£o o conteÃºdo, na Ã-ntegra, abaixo.
Infra-estrutura Unix no (Mac) OS X

Em sua documentaÃ§Ã£o oficial, a Apple aponta que o kernel do OS X (coraÃ§Ã£o do sistema operacional) Ã© construÃ-do com base nos sistemas BSD Unix e Mach.

Entre outras coisas, a parte BSD da alma do OS X implementa os recursos primÃ¡rios de sistemas de arquivos, serviÃ§os de rede e gerenciamento / identificaÃ§Ã£o de usuÃ¡rios e grupos do sistema. A seguranÃ§a Ã© tambÃ©m BSD, o subsistema BSD Unix garante que as polÃ-ticas de seguranÃ§a sejam imperativamente respeitadas em todos os aspectos do sistema, sistemas de arquivos, processos, rede e memÃ³ria. Ã‰ onde comeÃ§a a integraÃ§Ã£o do kernel BSD com o Mach, uma vez que o lado Mach na alma do OS X gerencia memÃ³ria, controle de threads, abstraÃ§Ã£o e gerenciamneto de hardware e comunicaÃ§Ã£o entre processos. O lado Mach tambÃ©m gerencia tarefas, supervisionado pelo subsistema BSD.

Parecem duas â€œalmasâ€• trabalhando em conjunto, e Ã© como o coraÃ§Ã£o do OS X realmente trabalha. Ã‰ um kernel hÃ e isso Ã© fundamental para a seguranÃ§a do sistema e nossa compreensÃ£o sobre essa seguranÃ§a. Vamos entÃ£o olhar isso mais de perto.

Quando Steve Jobs saiu da Apple, em Maio de 1985, oficialmente os argumentos de Scully para convencer a diretoria da empresa eram problemas financeiros e fracassos em projetos recentes, como de vendas quase insignificativas do Apple Lisa alinhados a queda na aceitaÃ§Ã£o de sistemas Mac. No entanto os desentendimentos precediam os fatores de negÃ³cio, e iniciavam-se na parte tÃ©cnica. Jobs estava descontente com o estado atual dos Mac e seu sistema, e insistia que os Mac deveriam assumir uma personalidade mais â€œUnixâ€•. Essa ideia era muito bem aceita por parte da equipe de desenvolvimento da Apple e mÃ¡ vista por outra parte e alguns gestores, como Scully. Foi sÃ³ recentemente que uma passagem da biografia autorizada de Steve Jobs indicou consistentemente que a ideia de tornar o Mac mais â€œUnixâ€• foi um dos grandes estopins para desentendimento de Jobs com Scully e a diretoria.

O resultado disso, sabemos, a saÃ-da de Jobs da Apple, o que abriu margem para novos empreendimentos de Jobs. AlÃ©m de sua relaÃ§Ã£o com Hollywood por meio da Pixar, Jobs criou a NeXT Computers, empresa que passaria a criar hardware e sistema operacional de grande poder de processamento matemÃ¡tico, com foco inicialmente no mercado acadÃªmico que sempre teve grandes demandas de poder computacional para simular estudos e cÃ¡lculos complexos. Jobs levou para a NeXT uma parte considerÃ¡vel dos engenheiros de software da Apple, os que queria trabalhar em um Unix. O que aumentou ainda mais o desconforto da diretoria da Apple Ã Ã©poca com seu fundador.

Os computadores da Next mostraram-se depois, adequados a processamento grÃ¡fico de grande desempenho, como efeitos grÃ¡ficos e animaÃ§Ãµes em filmes e comerciais de TV. Na NeXT, Jobs colocou em prÃ¡tica seus planos e junto aos computadores NeXT, incluindo o famoso Next Cube, foi criando um sistema operacional, o NEXTSTEP.

O NEXTSTEP juntou em um Ãºnico sistema, as caracterÃ-sticas Unix outrora planejadas e outros recursos. O sistema passou a ser fundamentado em um kernel de arquitetura Microkernel, o Mach. O Mach por sua vez Ã© um kernel criado na Universidade de Carnegie Mellon, cujo objetivo original era formar um sistema Unix completo, livre da licenÃ§a comercial da AT&T, para isso o conceito original do Mach Ã© um kernel rodando sob uma base (userland, aplicaÃ§Ãµes, bibliotecas, etc) BSD, jÃ¡ que o BSD Unix em versÃµes antes do 4.4-BSD Lite2 dependia da licenÃ§a do AT&T Unix (comercial) para ser utilizado. EntÃ£o o Mach, em uma base BSD, incorporando do kernel BSD as tecnologias

relevantes, criadas pela Universidade de Berkeley (onde o BSD foi criado), passou a dar origem ao NEXSTEP dentro da NeXT.

E para garantir resultados em seu projeto, Jobs foi beber na fonte, e contratou como Engenheiro Chefe de Software, Avie Tevanian, um dos criadores do Mach na Carnegie Mellon. Avie posteriormente seguiu com cargo similar (CSTO) dentro da prÃ³pria Apple atÃ© 2006.

Alguns recursos no entanto, como pilha de rede, sinalizaÃ§Ã£o de processos e sockets de rede, criados em Berkeley no BSD, Jobs queria ver em seu NEXTSTEP. EntÃ£o o NEXTSTEP comeÃ§ou a tomar uma forma Ãºnica, mesclando uma arquitetura de kernel monolÃ-tico (BSD) com microkernel (Mach), um dos poucos Ãºnicos sistemas com kernel hÃ-brido funcionais. Essa arquitetura permanece atÃ© hoje.

Portanto a inspiraÃ§Ã£o original do Mach foi incrementada no NEXSTEP que agora passava a ser um sistema de base BSD (userland), kernel Mach mas tambÃ©m kernel BSD. Devido a natureza hÃ-brida do novo sistema e monolÃ-tica do kernel BSD, o(s) microkernel Mach passaram a poder se comunicar independentemente (como originalmente) mas registrar-se no kernel monolÃ-tico BSD, de forma similar a mÃ³dulos e extensÃµes de kernel (chamados kext no OS X, kernel extensions), o que faz com que o microkernel Mach cumpra as exigÃªncias de seguranÃ§a do BSD.

Mais tarde isso seria melhorado, com a adoÃ§Ã£o dos Kernel Entry Points BSD no Mach (projeto TrustedBSD) permitindo componentes imperativos de controle de acesso (MAC â€“ Mandatory Access Control) do BSD impor controles tambÃ©m no Mach.

Junto a essa arquitetura Unix o NEXTSTEP trouxe display em PostScript e um subsistema de janelas grÃ¡ficas, junto com uma base de desenvolvimento fundamentada em kits de orientaÃ§Ã£o a objeto e uma nova linguagem: Objective-C. Essa plataforma computacional tambÃ©m foi a base para a criaÃ§Ã£o da Ãºltima dentre as principais tecnologias Internet, a Web. Ã€ Ã©poca jÃ¡ tÃ-nhamos correio eletrÃ´nico, DNS (criados em Berkeley, BSD), FTP, Gopher, etc. Mas foi sÃ³ em 1990 que Tim Berners-Lee em um NeXT Cube com NEXTSTEP criou o primeiro servidor HTTP e o primeiro navegador, batizado de WorldWideWeb (dando origem a sigla). A alma da web, o HTML, foi baseado no Textclass da NeXT.

Em 1997 a Apple adquiriu a NeXT, o interesse inicial era no sistema operacional. Agora o BSD jÃ¡ era livre (Open Source) e rodava em plataformas abertas. EntÃ£o a ideia era trazer pra Apple algo parecido ao que a NeXT tinha no NEXTSTEP, com base no NEXSTEP mas novo e melhor. Os componentes Mach foram reaproveitados do NEXTSTEP mas a base BSD deu lugar Ã base FreeBSD, um BSD mais moderno, livre, com uma licenÃ§a mais amigÃ¡vel e que tinha desenvolvedores originais do BSD Unix.

A Apple e seu Mac comeÃ§ou a retomar forma apÃ³s lanÃ§amento do iMac com um novo sistema, Mac OS 9. O Mac OS 9 foi uma tentativa, algumas caracterÃ-sticas Unix passaram a ser incorporadas mas sem muito sucesso: O Mac OS 9 era arcaico demais para incorporar aquela tecnologia. Jobs resolveu mater o Mac OS como era conhecido, e decretou o fim do Mac OS 9.

Paralelamente a isso, deu-se inÃ-cio a criaÃ§Ã£o de um novo sistema operacional dentro da Apple, um sistema naturalmente â€œevolucionÃ¡rioâ€•, batizado inspiradamente, Darwin. O Darwin era um Fork do FreeBSD, com caracterÃ-stic do kernel Mach herdadas do NEXTSTEP, com um kernel monolÃ-tico integrado agora FreeBSD, e toda a base (userland, bibliotecas, etc) FreeBSD e algumas vezes, NetBSD. Trechos do NetBSD foram inicialmente utilizados em aplicaÃ§Ãµes da base para a arquitetura PowerPC, tÃ-pica Ã Ã©poca dos iMac.

O Darwin passaria a ser um sistema baseado no FreeBSD, com heranÃ§as Mach do NEXTSTEP, e a Apple nessa base estÃ¡vel, sÃ³lida e comprovada, focaria seu desenvolvimento no â€œdiferencialâ€• Apple: o kernel adotou o â€œCore Graphic â€œCore Audioâ€•, a interface grÃ¡fica incorporou caracterÃ-sticas NEXSTEP incluindo o Dock, mas tambÃ©m a nova interfac Aqua, bibliotecas como Cocoa e outros recursos de usabilidade e user-feeling que tanto diferenciam a Apple.

Por trÃ¡s disso tudo, o coraÃ§Ã£o BSD, com artÃ©rias Mach. Isso alinhava-se com outra intenÃ§Ã£o de Jobs: a troca de arquitetura, sair de PowerPC e ir para Intel. SÃ³ hoje, apÃ³s ler a bibliografia de Jobs sabÃ-amos que ele planejou a longo prazo, escolhendo como base um sistema focado em arquitetura Intel, sem se importar em ter que fazer adaptaÃ§Ãµes temporÃ¡rias para rodar tambÃ©m em PowerPC: a plataforma mais tarde seria descontinuada em favor de arquitetura Intel apenas, tornando a escolha FreeBSD com o tempo mais acertada ainda.

Em 2000 foi lanÃ§ado o novo Mac OS, a dÃ©cima (e tÃ£o diferente) versÃ£o, O Mac OS X (10 em Romano)

Curiosidade: a Apple repreende fortemente, dizer â€œMac OS Xisâ€•, deixando claro que Ã© Mac OS Ten â€“ Dez â€“ agora a OS 10, inclusive essa sendo uma das primeiras regras para falar publicamente como parceiro Apple, seja em treinamento, consultoria ou negÃ³cios.

E para garantir qualidade, mais uma vez, Jobs foi beber da fonte: contratou como engenheiro chefe de software Unix na

Apple, Jordan K. Hubbard, fundador e criador do FreeBSD (jkh@freebsd.org). O JKH ainda ocupa esse cargo, sendo a pessoa por trÃ¡s das bases Unix do OS X e do iOS (co-worker direto de Forstall). Hubbard foi apresentado a Steve Jobs atravÃ©s de um amigo em comum, John Lasseter, um dos 3 fundadores originais da Pixar, hoje animador na Disney. Lasseter, um dos grandes nomes da animaÃ§Ã£o grÃ¡fica de todos os tempos, alÃ©m de amigo pessoal de Jordan Hubbard e Steve Jobs, Ã© ainda amigo de McKusick, um dos pais do BSD Unix.

Curiosidade: A amizade Ã© tÃ£o prÃ³xima que curiosamente, o mascote do BSD (o daemon) originalmente rascunhado por McKusick, tem sua Ãºltima e definitiva versÃ£o assinada por John Lasseter, e o conceito por trÃ¡s do Hexley, o mascote do Darwin, Ã© tambÃ©m de Lasseter e arte de Jon Hooper.

O OS X tem desde entÃ£o evoluÃ-do, em recursos que normalmente sÃ£o focados na usabilidade e interesse geral do usuÃ¡rio final. Mas Ã© a evoluÃ§Ã£o por trÃ¡s de sistema de arquivos, gerencia de recursos que permite que tenhamos resultados como Versions, Resume, e agora recursos de seguranÃ§a como Gatekeeper e Mac Sandbox.

Os recursos de seguranÃ§a do OS X sÃ£o fatores diretamente herdados (e construÃ-dos juntos, a bem da verdade) da base BSD Unix. Um projeto chamado TrustedBSD financiado por 10 anos pela AgÃªncia de Pesquisas e Projetos AvanÃ§ados de Defesa (DARPA) do Departamento de Defesa dos EUA (DoD), passou a ser criado sob base FreeBSD (e para FreeBSD) com objetivo de implementar requisitos de seguranÃ§a da informaÃ§Ã£o de especificaÃ§Ãµes militares (baseadas no Orange Book / TCSEC â€“ Trusted Computer Systems Evaluation Criteria) e na especificaÃ§Ã£o POSIX.1e. A Apple e Sun Microsystems passaram a ter interesse direto nesse projeto e participaram apoiando o desenvlvimento dessas tecnologias no FreeBSD e posterior adoÃ§Ã£o em seus sistemas.

Isso vale a Apple o certificado oficial de sistema Unix, e tambÃ©m um dos Ãºnicos sistemas operacionais classificados com padrÃ£o de seguranÃ§a TCSEC/B3 pelo DoD dos EUA. Os recursos incorporados no OS X herdados dessa parceria de desenvolvimento em seguranÃ§a sÃ£o:
- POSIX.1e ACLs â€“ Os controles de acesso, as ACL, permissÃµes para usuÃ¡rios, grupos, etc que vocÃª configura com Cmd+i no seu Mac / Finder;
- MAC (Mandatory Access Control) â€“ OS X incorpora MAC em kernel entry points, ou seja sÃ£o pontos de entrada que o kernel olha uma polÃ-tica para decidir se a aplicaÃ§Ã£o ou usuÃ¡rio tem direito de executar alguma funÃ§Ã£o; as polÃ-ticas MAC mais famosas implementadas no OS X sÃ£o Seatbelt e Sandbox;
- BSM Audit â€“ Subsistema de auditoria de acesso a recursos;
- Atributos Extendidos no Sistema de Arquivos (Finder e comando xattr(1);

Esses sÃ£o alguns recursos que oferecem, acima de tudo, camadas de abstraÃ§Ã£o de seguranÃ§a avanÃ§adas para os programadores no OS X. Ou seja alguns desses recursos o usuÃ¡rio final nÃ£o vÃª tomando forma, outros apenas vÃª em modo Trusted (processo de hardening do OS X o qual nossos 50 artigos, aos poucos, ajudarÃ¡ vocÃª a implementar) ou em versÃµes OS X Server.

Aos poucos a Apple tem tentado trazer esses recursos pro usuÃ¡rio, jÃ¡ que eles jÃ¡ existem no sistema. Mas trazer os recursos de seguranÃ§a desse porte sem comprometer a usabilidade e facilidade com que o usuÃ¡rio estÃ¡ acostumado leva tempo, e vem aos poucos no sistema. Outros, mais radicais jÃ¡ estÃ£o disponÃ-veis.

AlÃ©m desses recursos, o OS X ainda herda da base BSD outros componentes de seguranÃ§a como:
- Security Framework: comando security(1) que implementa para o usuÃ¡rio final a Keychain;
- SeparaÃ§Ã£o imperativa de processos de usuÃ¡rios (MAC BSD_Partition), que o usuÃ¡rio nÃ£o vÃª, mas acontece quando se usa por exemplo o Fast User Switching no OS X;
- Firewall: Em Security no Systems Preference do seu OS X, vocÃª tem o IPFW, FreeBSD Firewall nativo, tanto IPv4 (ipfw) quanto IPv6 (ip6fw);
- Firewall2: A partir do OS X Lion, o Packet Filter (comando pfctl no terminal) tambÃ©m estÃ¡ disponÃ-vel no OS X;
- Geli: Subsistema de criptografia de disco e memÃ³ria virtual com chave simÃ©trica ou assimÃ©trica, o usuÃ¡rio vÃª como FileVault e FileVault2 no OS X;
- srm: Secure Remove, comando srm no terminal, variaÃ§Ã£o do comando rm -P no FreeBSD, apaga um arquivo em conformidade com o padrÃ£o militar (DoD) reescrevendo o arquivo diversas vezes, evitando recuperaÃ§Ã£o de dados apagados. O usuÃ¡rio final enxerga isso como â€œLimpeza segura da lixeiraâ€• nas preferÃªncias do Finder;
- Root Tampering: o poder de root (system administrator) parcialmente delegado para usuÃ¡rios administradores ou integralmente com o comando â€œsudo suâ€•, sem permitir que o usuÃ¡rio root logue, aumentando rastreabilidade e auditori
- Dtrace: rastreamento e controle de chamadas de sistemas de kernel;
- TCP Wrappers (controle de respostas em serviÃ§os de rede por aplicaÃ§Ã£o);
- chflags â€“ habilidade de transformar arquivos em imutÃ¡veis (mesmo para o administrador e system administrator â€“ aka root), invisÃ-veis, inapagÃ¡veis ou apenas concatenaÃ§Ã£o;
- BSD IPSec â€“ habilidade de criptografar pacotes de rede, criando tÃºneis seguros de propÃ³sito geral e VPNs nativamente;
- BSD OpenSSL â€“ implementaÃ§Ã£o com as modificaÃ§Ãµes do FreeBSD/OpenBSD para aceleraÃ§Ã£o de criptografia po hardware;
- Kext Control â€“ controle de carregamento de mÃ³dulos de kernel (kernel extensions no OS X);

- Mac Ports - uma das coisas mais legais do Mac OS X, a possibilidade de ter um subsistema de Ports, pra instalar aplicaÃ§Ãµes Open Source; criado pelo prÃ³prio JKH (autor original do Ports do FreeBSD), uma pena nÃ£o vir por padrÃ£o no OS X, mas pode ser facilmente instalado online;

Esse trecho do artigo constarÃ¡ da nova versÃ£o do FreeBSD S.S.A.

Original: http://www.tracanelli.com.br/blog/infra-estrutura-unix-no-mac-os-x/