SunSSH and OpenSSL Enhancements in OpenSolaris in 01/2008-06/2009

Jan Pechanec
Sun Microsystems
07/2009

Overview of SunSSH Changes

- A Short History of SunSSH and Its Versions
- SunSSH versus OpenSSH (differences)
- SunSSH with HW Crypto Support
- Security Fixes
- Support for 192/256 bit encryption
- New modes **`arcfour128`** and **`arcfour256`**
- Command Line Editing in sftp(1)
- New **`ChrootDirectory`** Option
- MAC Context Preservation For Transfer Speed-up

Overview of SunSSH Changes (cont.)

- **`AllowTCPForwarding=yes`** is the Default Now
- Various Other Fixes
- STC-2 SSH Test Suite Enhancements
- Future Plans & Ideas

Overview of OpenSSL Changes

- Move OpenSSL from **`/usr/sfw`** to **`/`**
- Move OpenSSL to SFW Consolidation
- Upgrade from 0.9.8a to 0.9.8k
- Performance Improvements

A very short history of SunSSH

- version **1.0** based on OpenSSH **2.3p1** (11/2000)
- **1.1** based on OpenSSH **3.5p1** (10/2002)
  - > full 3.5p1 was taken and our patches from 1.0 reapplied
  - > in Solaris 10 since its FCS (now in S9 through backport)
- a fork due to a need for Solaris specific changes
- we resync only individual features and fixes after 2003
- very different code in PAM, GSS-API, privilege separation, auditing, g11n (new code)
  - > plus a lot of minor changes
- current version in OpenSolaris is **1.4**

On SunSSH Versioning

- new versions added for protocol or security fixes only
  - **ie. the version number is NOT about features (= RFEs)**
  - version string is mostly used to trigger compatibility extensions. It's the first thing the client gets:

```
$ nc localhost 22

SSH-2.0-Sun_SSH_1.4
```

- fixes can be backported to S10 and S9 out of order
  - if 1.2 fix not backported but one for 1.3 is, we must branch
  - that's why current version in S10 is 1.1.2
    - SunSSH in S9 might even get another level
  - backported fix for 1.2 would change S10's version to 1.3

SunSSH versus OpenSSH

- *"OpenSSH supports this, SunSSH does not. Why don't you just start using OpenSSH?"*
  - there are quite a few differences between those 2 now
  - we have changes our customers rely on
    - that are not in OpenSSH
    - while we try to give our changes back to the upstream not all of them would be accepted due to the different opinions on suggested solutions
    - most importantly, **all** would have to be accepted by the upstream otherwise we would introduce regressions
  - shipping both seems unlikely as well, we would have to modify OpenSSH to fit into SMF, for example
    - effectively forking it again

SunSSH with HW Crypto Support

- problem: “*SunSSH is slow on Niagara :-(”*
- 6445288 *ssh needs to be OpenSSL engine aware*
- because SunSSH is single threaded
  - ...as well as most of other SSH implementations
- and T1/T2 CPUs are quite slow comparing to CPUs used in ,,typical" 1-4 CPU machines
  - we could not even saturate 100Mbit ethernet with T5120 and SunSSH using default AES mode
  - the transfer speed was only ~9MB/s (~75Mbit/s)
- most time spent in crypto operations

Some numbers on SSH data transfer

- before: transfer 500MB of data through SunSSH
  - ie. all crypto done in software
- **~9MB/sec** on UltraSPARC T2 (~75Mb/sec)

```
$ time dd if=/dev/zero bs=1024k count=500 | ssh t5220-sfb-06 'cat >/dev/null'

real    0m53.581s
user    0m11.951s
sys     0m6.153s
```

- **~25MB/sec** on AMD64, 2.4GHz CPU (~200Mb/sec)

```
$ time dd if=/dev/zero bs=1024k count=500 | ssh zup.czech 'cat >/dev/null'

real    0m20.101s
user    0m16.359s
sys     0m2.115s
```

How to speed up the encryption?

- SW parallelism
  - **encrypt in parallel using threads**
  - from cipher modes we support, only AES-CTR can be used
  - that's the default mechanism used in SunSSH
    - and also in OpenSSH since version 5.2p1 (02/2009)
  - this way is suitable for all Niagara machines
- HW solution through the Crypto Framework
  - **offload encryption to n2cp(7d) or mca(7d)**
  - suitable for machines with HW crypto providers
    - so, not useful for Niagara-1 without SCA-6000

The way we decided to go for now...

- **...was through the hardware solution**
- not suitable for Niagara-1
  - > which can only accelerate RSA/DSA/DH through the *modexp* operation offered by ncp(7d)
- SCA-6000 can always help on T1
  - > mca(7d) is the driver name
- n2cp(7d) on T2 machines was our primary target
  - > SCA-6000 is significantly slower
- combining with the software solution is the long term plan. No ETA for that though.

Some numbers after the fix

- now: transfer 500MB of data through SSH
- previously on T2 (**~75Mb/sec**)

```
$ time dd if=/dev/zero bs=1024k count=500 | ssh t5220-sfb-06 'cat >/dev/null'

real    0m53.581s
user    0m11.951s
sys     0m6.153s
```

- and with the PKCS#11 engine (**~190Mb/sec**)

```
$ time dd if=/dev/zero bs=1024k count=500 | ssh t5220-sfb-06 'cat >/dev/null'

real    0m20.839s
user    0m11.962s
sys     0m6.179s
```

Some Thoughts on the Results

- **current results seem quite good for the 1st phase**
  - > we'll try to continue to search for bottlenecks
  - > see “Future Plans & Ideas” slides
- with **~190Mb/s** on T2 it seems using threads to make use of more cores should easily saturate 1Gb link
  - > BTW, T2 has 8 cores, each with a crypto chip
- even parallelization without any HW acceleration might get interesting results
  - > imagine 128 CPUs
  - > however, we can just speculate (we have no prototype)

Final Notes on HW Crypto Support

- integrated in Nevada build 99 (PSARC/2008/520)
- if in troubles (hitting a bug in the Crypto Framework provider for example), you can always switch the engine off via "`-o UseOpenSSLEngine=no`"
- performace tests run with 0.9.8a only
  - Nevada build 118 integrated 0.9.8k which is faster for some cipher modes than 0.9.8a
  - speed on SPARC machines is generally the same though
- the project was also backported to S10u7 and already being used by our customers

Security Fixes

- 6684003 *fix CVE-2008-1483 in SunSSH*
  - a bug inherited from the upstream
  - fixing the possibility of hijacking the X11 connections
- 6761890 *ssh protocol security vulnerability may be used to reveal some plaintext*
  - CBC modes only affected, and very hard to successfully mount an attack in the real world
  - to mitigate, we do not offer CBC modes on the server side by default on OpenSolaris any more (but we do in S10 due to potential compatibility problems)
  - this bumped up the version to 1.3 (and 1.1.1 in S10)

Support for 192/256 Bit Encryption

- supporting only 128 bit keys was an artifact of having SUNWcry* packages
- 6617424 *aes192/aes256 support is missing from ssh/sshd*
  - integrated in Nevada build 87
- a minor code resync from OpenSSH code base
- presently, using 192/256 bit keys looks like burning CPU cycles while **`aes128-ctr`** mode seems to be good enough for now
  - some people do not share such opinion though
    - (update 08/2009) also, read latest news on attacking aes192/256

Support for arcfour(128|256) modes

- both modes discard the first 1536 bytes of the generated stream
  - to mitigate known attacks against the plain RC4 algorithm
  - existing **`arcfour`** mode moved to the back of the default cipher list
- 6799060 *implement arcfour128 and arcfour256 in SunSSH*
- the new modes are defined in RFC 4345
- straightforward resync from OpenSSH
- integrated into Nevada build 109

Command Line Editing in sftp(1)

- using the tecla(5) library shipped with Solaris
- 6480741 *command line editing is desired for sftp(1)*
- command line editing and history only
  - file name completion not implemented yet
  - not that easy for remote files
- integrated to Nevada build 110
- BTW, OpenSSH participates in Google's Summer of Code 2009, renovating its SFTP implementation

ChrootDirectory option

- OpenSSH team integrated **`chroot(2)`** support to OpenSSH 5.1p1
- the user is jailed into the directory after logging in
- if "**`internal-sftp`**" is used as path in the **`Subsystem`** option for sftp, no other configuration is needed
  - that's the default now but only for new installations
- for plain SSH connections, the chroot directory must contain at least the dynamic loader, the shell and its libraries. See **`sshd_config(4)`**.

ChrootDirectory option (cont.)

- 5043377 *provide chroot capability in SunSSH*
  - PSARC/2009/155 ChrootDirectory option for SunSSH server
- integrated into Nevada build 112
- partial resync from OpenSSH, some parts different due to different privilege separation code we use
  - BTW, bugs #1562, #1564, #1566 filed against OpenSSH
- SunSSH started to use privileges for the 1st time
  - user can now even login without **`PRIV_EXEC`** and **`PRIV_FORK`**, see "**`defaultpriv`**" in **`user_attr(4)`**
    - but those privileges would not be assigned to the shell
    - ie. you could run the shell's internal commands only

MAC Context Preservation

- each SSH packets is guarded by HMAC for integrity protection using the same key (until the next key re-exchange)
- previously, HMAC was initialized with that (same) key for every packet
  - involves some memory copying and hash operations to prepare a context
- the “idea” of the context preservation is to initialize once and reuse the context for all the packets
- resync from OpenSSH
  - 6616927 *preserve MAC contexts between packets*

MAC Context Preservation II.

- OpenSSH release notes claimed 12-16% speed up with **`arcfour128+hmac-md5`**
- hmm, that does not seem to be true
  - > my expected speed-up based on some computations was around 5%
    - – based on isolated tests with MD5, how much is spent in crypto in general, and what is the ratio of RC4:MD5 wrt speed
  - > tests confirmed that
- to summarize, we get 4-5% for the faster **`arcfour128+hmac-md5`** case and 1.5% for the default case with AES-CTR in 128 bits

`AllowTCPForwarding=yes` is the default now

- setting it to “no” does not increase security of the server (see the man page as to why)
- was rather a bug when introducing SSH in Solaris
- the real default in the code was always “yes” but **`/etc/ssh/sshd_config`** set it explicitly to “no”
  - manual page said (quite correctly) the default was “yes”
  - which added to a great confusion among our users
- the new default is not set in the config file any more
  - see 6805294 below for more information
- integrated to snv_118 (6830483, PSARC/2009/353)

Various Other Fixes

- we fix other bug fixes along the way, for example:
- 6496644 *deprecate UseLogin and remove code supporting this feature from sshd(1m)*
  - this option depricated in an old PSARC case was still accepted and made all connections to fail
- 6635417 *more memory leaks in SunSSH*
  - finally, we can add memory leak checks support to the STC-2 SSH test suite
- 6820920 *Sun SSH daemon crashes if /usr/bin/locale isn't present*
- and many more (around 40 within the period)

STC-2 SSH Test Suite Enhancements

- STC = Solaris Test Collection
- the up-to-date test suite is extremely important for the continuous development
  - test suite was written based on OpenSSH regression code from 4.3p2 release (02/2006)
  - independently developed since then
  - not open sourced yet (would need some work)
    - it will hopefully happen “soon”
- **really easy to run**
  - it needs some manual configuration of the system but configure/execute phases tell you exactly what to do if something is found missing

STC-2 SSH Enhancements (cont.)

- 15 putbacks within the period covered by this presentation (01/2008-06/2009)
  - some containing many CRs
- 3 new test cases **`solaris-auditing`**, **`privs-and-chroot`**, and **`max-conn`**
- many other test cases fixed or enhanced
- important stuff still missing
  - GSS-API test case must be written from scratch
  - memory leak tests missing
  - some command line options not tested at all

Future Plans & Ideas (SunSSH)

- 6655613 *resync server's conditional Match block from OpenSSH*
  - conditional configuration on the server side based on various criteria (user name, group, host, address)
  - can set a subset of options based on that (auth options, chroot, x11 and forwarding options, banner, and more)
- 6805294 *sshd_config should not be shipped with explicit default values*
  - any uncommented default value in the configuration file becomes part of the settings, immune to code changes
  - that's a potentially big issue with critical options like **`Ciphers`** and **`MACs`**.

Future Plans & Ideas II.

- 6749535 *ssh could precompute AES-CTR stream in larger chunks and XOR data with it*
  - the Crypto Framework has quite a big overhead
  - currently, up to 8KB data chunks are processed but system sometimes ships data in smaller ones
  - however, with the counter mode we could precompute the encrypted stream in larger chunks and XOR with data later
  - perf tests on a T2 machine with the PKCS#11 engine:

```
type             16B       1KB        8KB         64KB
aes-128-ctr 664.74k  33282.39k  147720.87k  251374.25k per second
```

  - not everything is crypto but I expect ~30% speed-up with this approach. Also, think threads using more cores!

Future Plans & Ideas III.

- 6357779 *SSHv2 x.509 support desired*
  - would fix the today's leap-of-faith step when accepting the server keys during the 1st connection
    - such keys should be transfered via a different channel and put to the **`known_hosts`** file but that usually is not the case
  - only informational RFC and some drafts exist
  - this might be omitted, and we could relay on PKINIT only
    - ie. using the X.509 certificates through GSS-API
  - alpha code for 6357779 was succesfully tested in San Jose during Connectathon 2008, against Van Dyke's and Attachmate's SSH implementations

Future Plans & Ideas IV.

- 6628064 *High Performance SSH/SCP - HPN-SSH*
  - SSH does not perform very well on high speed links (≥ 1Gb/sec) with high latency
  - TCP window handling must be fixed to significantly speed up the bulk data transfer
  - there are patches that can be applied against OpenSSH
- paralellization of crypto operations
  - already mentioned on one of previous slides (with 6749535)
  - might help on any machine with more CPUs
    - AES CTR stream would be being precomputed independently, using a pool of threads

References (SunSSH)

- http://www.opensolaris.org/os/community/security/projects/SSH
  - SunSSH home pagbe; includes information on history, current and past development, documentation, patches etc.
- http://www.openssh.org
  - our upstream is always a good source for information
- manual pages
  - see Documentation section on SSH page at opensolaris.org
- http://docs.sun.com – more detailed documentation on SSH than manual pages
- SunSSH and the OpenSSL PKCS#11 Engine Changes in 2008
  - detailed presentation on the HW crypto support in SunSSH
  - see http://mediacast.sun.com/users/janp2
- “On SunSSH Versioning”, “The `ChrootDirectory` option resynced to SunSSH”
  - look for those entries on my blog http://blogs.sun.com/janp
- Pics for the cover and back slides from Barcelona
  - http://www.devnull.cz/barcelona

OpenSSL Changes Overview

- up until recently, only the PKCS#11 engine changes and security fixes went to 0.9.8a version within the last few years
- then, JohnZ moved OpenSSL from /usr/sfw(lib|bin|include) to /(lib|bin|usr/include)
- and Mark Phalan's recent changes brought in some big enhancements in this area during the last few Nevada builds
  - OpenSSL was moved to SFW consolidation
  - its version upgraded from 0.9.8a to 0.9.8k
  - and we got significant performance improvements

OpenSSL – Move from /usr/sfw to /

- 6449514 *move OpenSSL from /usr/sfw to /usr, /lib*
- integrated by John Zolnowski to snv_104
- PSARC/2006/555 and PSARC/2007/674
- no need to use **`-R/L/I`** options anymore
  - good for the 3rd party apps that didn't know about **`/usr/sfw`** in Solaris
  - with **`/usr/sfw`** going away, there is no need to put it to your **PATH**
    - which was not the default, confusing many users
- this RFE was a long needed change

OpenSSL – Move to SFW

- integrated by Mark Phalan to Nevada build 117
- will simplify future upgrades and maintenance
  - just drop in a new tarball
  - and verify that patches (eg. PKCS#11 engine) still work
- compiler freedom
  - gcc vs cc for best performance
  - cc is used now since it provides faster code for OpenSSL
- delivering a FIPS container should be now possible
  - SFW allows the original verified build system to be used
  - and we could even provide more OpenSSL versions
    - the question is whether we would ever want that

OpenSSL – Upgrade to 0.9.8k

- Nevada build 118 (by Mark)
- we got a lot of bug fixes with the upgrade as well
- ...and speed improvements
  - new and better assembler code available for AMD64
- pleasing our users
  - 0.9.8a version, while patched against all security bugs found, was still confusing our users and triggered complains about security risks
  - security vulnerabilities since now will be (mostly) fixed with a new OpenSSL release
    - we still might use patches for low priority security bugs

OpenSSL – Speed Improvements

- objective: meet OpenSSL performance as on Linux
- enable ASM for AMD64
  - Nevada 118
  - hmac(md5): 150%, aes cbc: 170%, rsa: 325%, ...
- enable ASM for x86
  - fix in progress (planned for snv_120)
  - hmac(md5): 135%, aes cbc: 310%, rsa: 185%, ...
- enable better ASM for RSA on SPARC
  - RSA 2048 200% (32bit)
  - anything else on SPARC stays the same

OpenSSL – Post Move Plans

- WAN Boot
  - alive and used in the new installer
  - now uses a private copy of OpenSSL in ON
    - extracting and applying security patches might be still needed
  - we want to remove OpenSSL from ON completely
    - but WAN Boot needs a static OpenSSL library
  - will want to build static libraries for WAN Boot from SFW
    - and deliver them via private package for ON build machines
- isaexec for openssl(1openssl) for x86 only
  - SPARC's 64 bit code is not generaly faster than 32 bit
- isaexec for ssh(1)/sshd(1M)

AMD64 Perf Improvements (in %)

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
md2	135.26	134.28	132.47	132.12	131.84
mdc2	0.00	0.00	0.00	0.00	0.00
md4	149.72	138.16	124.82	110.10	101.54
md5	155.25	155.18	157.56	158.13	157.59
hmac(md5)	119.40	132.13	140.27	150.34	156.60
sha1	160.90	171.03	200.14	202.45	203.81
rmd160	124.81	118.66	110.12	103.12	100.48
rc4	100.02	100.65	100.81	100.78	100.76
des cbc	100.09	99.69	99.99	100.03	100.31
des ede3	100.29	100.17	100.20	100.19	100.19
idea cbc	0.00	0.00	0.00	0.00	0.00
rc2 cbc	100.10	100.02	100.00	100.01	100.01
rc5-32/12 cbc	0.00	0.00	0.00	0.00	0.00
blowfish cbc	99.54	99.99	99.99	100.00	99.98
cast cbc	100.00	100.32	100.00	99.99	99.95

AMD64 Perf Improvements II. (in %)

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
aes-128 cbc	103.05	147.06	166.36	171.44	173.08
aes-192 cbc	100.51	143.19	163.24	168.30	170.55
aes-256 cbc	107.20	145.08	160.81	164.78	166.14
sha256	210.01	219.69	225.72	230.18	231.81
sha512	218.38	224.36	229.57	228.90	227.84

	sign	verify	sign/s	verify/s
rsa 512 bits	30.75	39.47	325.19	247.34
rsa 1024 bits	28.26	37.14	353.71	270.83
rsa 2048 bits	29.15	35.20	343.10	283.06
rsa 4096 bits	30.27	36.90	330.95	271.04
	sign	verify	sign/s	verify/s
dsa 512 bits	44.75	33.00	223.50	302.07
dsa 1024 bits	34.48	29.53	290.07	338.81
dsa 2048 bits	31.54	29.42	316.96	339.68

AMD64 Improvements in 32 bits (%)

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
md2	100.12	100.06	100.37	100.03	100.04
mdc2	0.00	0.00	0.00	0.00	0.00
md4	100.00	99.97	99.53	100.05	99.98
md5	100.69	104.41	111.90	124.79	137.52
hmac(md5)	115.36	119.17	125.04	133.39	139.18
sha1	104.42	107.10	109.56	112.07	113.79
rmd160	101.10	101.13	100.41	100.07	99.79
rc4	99.63	99.88	99.96	99.92	99.88
des cbc	148.35	147.67	146.88	147.33	146.92
des ede3	129.68	129.45	129.54	129.58	129.22
idea cbc	0.00	0.00	0.00	0.00	0.00
seed cbc	0.00	0.00	0.00	0.00	0.00
rc2 cbc	100.23	99.89	100.02	100.02	100.03
rc5-32/12 cbc	0.00	0.00	0.00	0.00	0.00
blowfish cbc	125.92	121.08	120.94	120.53	120.90
cast cbc	99.71	100.01	100.01	99.99	100.01

AMD64 Improvements in 32 bits II. (%)

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
aes-128 cbc	182.08	261.64	297.22	307.45	310.41
aes-192 cbc	188.00	268.64	300.92	310.48	313.42
aes-256 cbc	195.56	272.84	302.46	310.81	313.37
sha256	100.44	100.84	100.40	100.84	101.32
sha512	99.11	99.37	99.14	99.08	99.14
aes-128 ige	227.30	245.34	256.29	259.75	259.05
aes-192 ige	228.39	254.24	264.41	268.04	266.93
aes-256 ige	237.31	260.65	269.74	272.43	271.22

	sign	verify	sign/s	verify/s
rsa 512 bits	71.22	71.23	140.33	138.58
rsa 1024 bits	61.30	61.05	163.10	164.27
rsa 2048 bits	54.86	56.14	182.06	177.86
rsa 4096 bits	53.09	55.04	187.32	181.63

	sign	verify	sign/s	verify/s
dsa 512 bits	65.37	67.06	152.86	149.00
dsa 1024 bits	55.73	56.80	179.42	176.07
dsa 2048 bits	53.34	54.98	187.44	181.82

References (OpenSSL)

- http://www.openssl.org
  - home page for the OpenSSL project
- manual pages
  - openssl(5) is Solaris specific manual page on OpenSSL
  - “(1|3|7)openssl” sections are manual pages extracted from the original OpenSSL tarball
- “OpenSSL version string format changed”
  - http://blogs.sun.com/janp/entry/openssl_version_string_format_changed
- Mark Phalan's blog entries on OpenSSL
  - http://blogs.sun.com/mbp

Questions?

Jan Pechanec
https://blogs.sun.com/janp