OpenSolaris for security geeks

Darren Moffat – Sr. Staff Engineer

Solaris Network & Security Technologies

Agenda

- Overview of Solaris security features in ON/OpenSolaris
- Using RBAC in OpenSolaris & Your code
  - Privileges, Authorisations, the *_attr databases
- Wheres my crypto ?
- Q&A
  - Feel free to interupt at any time with clarifications or questions (or corrections!).

ON/OpenSolaris Security Features

- Kerberos
- GSS
- SSH
- OpenSSL
- IPFilter
- Crypto Framework
- RBAC
- Privileges
- PAM
- bart(1)
- TCP Wrappers
- SASL

Why do we need Privileges?

- Lots of software running as *root*
  - Root software has power to do anything
- Customer concern about why software needs to run as root
- Any *root* software breach is catastrophic
- Most *root* software only needs 1-2 special powers

Solaris Least Privilege

- Define atomic privileges associated w/root
- Change OS to check for privilege
  - not uid 0
- Change root run software to possess only required privileges
  - Paradigm shift since it needs to happen for Solaris and layered software
  - Solaris 10 GA will have some daemons converted
  - More daemons expected in Update releases
  - Good things for OpenSolaris process learning!

What do privileges look like?

- Privileges seen as strings in user-land
- Kernel maps these to bit maps
- Even kernel consumers use strings
- Privileges arranged in “sets”
  - These sets are stored in process credentials
  - Visible via /proc/<pid>/priv [ppriv(1)].
- Changes throughout kernel (priv checks)
  - E.g. Network stack for TCP privileged port range

Privilege Sets

- 48 fine grained privileges instead of uid == 0
  - ppriv -lv : Shows privilege and what it protects.
- Each process has 4 privilege sets in its' process credential:
- Inheritable set (I)
  - The set of privileges child processes get on exec.
- Permitted set (P)
  - The maximum set of privileges for the process
- Effective set (E)
  - Subset of P that are currently asserted as needed by the process
- Limit set (L)
  - Upper bound a process and its children can obtain (takes effect on exec)

Solaris's 48 Privilege Names

“contract_event”	Process/Request critical/reliable events
“contract_observer”	Obsever events other than euid
"cpc_cpu”	Access to per-CPU perf counters
"dtrace_kernel"	DTrace kernel tracing
"dtrace_proc"	DTrace process-level tracing
"dtrace_user"	DTrace user-level tracing
"file_chown"	Change file's owner/group IDs
"file_chown_self"	Give away (chown) files
"file_dac_execute"	Override file's execute perms
"file_dac_read"	Override file's read perms
"file_dac_search"	Override dir's search perms
"file_dac_write"	Override (non-root) file's write perms
"file_link_any"	Create hard links to diff uid files
"file_owner"	Non-owner can do misc owner ops
"file_setid"	Set uid/gid (non-root) to diff id
"ipc_dac_read"	Override read on IPC, Shared Mem perms
"ipc_dac_write"	Override write on IPC, Shared Mem perms
"ipc_owner"	Override set perms/owner on IPC
"net_icmpaccess"	Send/Receive ICMP packets
"net_privaddr"	Bind to privilege port (<1023+extras)
"net_rawaccess”	Raw access to IP
"proc_audit”	Generate audit records
"proc_chroot”	Change root (chroot)
"proc_clock_highres"	Allow use of hi-res timers
"proc_exec"	Allow use of execve()
"proc_fork"	Allow use of fork*() calls
"proc_info"	Examine /proc of other processes
"proc_lock_memory"	Lock pages in physical memory
"proc_owner"	See/modify other process states
"proc_priocntl"	Increase priority/sched class
"proc_session"	Signal/trace other session process
"proc_setid"	Set process UID
"proc_taskid"	Assign new task ID
“proc_zone”	Signal/trace processes in other zones
“sys_acct”	Manage accounting system (acct)
“sys_admin	System admin tasks (node/domain name)
"sys_audit"	Control audit system
"sys_config"	Manage swap
"sys_devices"	Override device restricts (exclusive)
"sys_ipc_config"	Increase IPC queue
"sys_linkdir"	Link/unlink directories
"sys_mount"	Filesystem admin (mount,quota)
"sys_net_config"	Config net interfaces,routes,stack
"sys_nfs"	Bind NFS ports and use syscalls
"sys_res_config"	Admin processor sets, res pools
"sys_resource"	Modify res limits (rlimit)
"sys_suser_compat"	3rd party modules use of suser
"sys_time"	Change system time

Basic — Non-root privileges

Interesting — Some interesting privileges

Zones are Less Privileged

“contract_event”	Process/Request critical/reliable events
“contract_observer”	Obsever events other than euid
"cpc_cpu”	Access to per-CPU perf counters
"dtrace_kernel"	DTrace kernel tracing
"dtrace_proc"	DTrace process-level tracing
"dtrace_user"	DTrace user-level tracing
"file_chown"	Change file's owner/group IDs
"file_chown_self"	Give away (chown) files
"file_dac_execute"	Override file's execute perms
"file_dac_read"	Override file's read perms
"file_dac_search"	Override dir's search perms
"file_dac_write"	Override (non-root) file's write perms
"file_link_any"	Create hard links to diff uid files
"file_owner"	Non-owner can do misc owner ops
"file_setid"	Set uid/gid (non-root) to diff id
"ipc_dac_read"	Override read on IPC, Shared Mem perms
"ipc_dac_write"	Override write on IPC, Shared Mem perms
"ipc_owner"	Override set perms/owner on IPC
"net_icmpaccess"	Send/Receive ICMP packets
"net_privaddr"	Bind to privilege port (<1023+extras)
"net_rawaccess”	Raw access to IP
"proc_audit”	Generate audit records
"proc_chroot”	Change root (chroot)
"proc_clock_highres"	Allow use of hi-res timers
"proc_exec"	Allow use of execve()
"proc_fork"	Allow use of fork*() calls
"proc_info"	Examine /proc of other processes
"proc_lock_memory"	Lock pages in physical memory
"proc_owner"	See/modify other process states
"proc_priocntl"	Increase priority/sched class
"proc_session"	Signal/trace other session process
"proc_setid"	Set process UID
"proc_taskid"	Assign new task ID
“proc_zone”	Signal/trace processes in other zones
“sys_acct”	Manage accounting system (acct)
“sys_admin	System admin tasks (node/domain name)
"sys_audit"	Control audit system
"sys_config"	Manage swap
"sys_devices"	Override device restricts (exclusive)
"sys_ipc_config"	Increase IPC queue
"sys_linkdir"	Link/unlink directories
"sys_mount"	Filesystem admin (mount,quota)
"sys_net_config"	Config net interfaces,routes,stack
"sys_nfs"	Bind NFS ports and use syscalls
"sys_res_config"	Admin processor sets, res pools
"sys_resource"	Modify res limits (rlimit)
"sys_suser_compat"	3rd party modules use of suser
"sys_time"	Change system time
Interesting	Some interesting privileges
Basic	Non-root privileges
Removed	Not available in Zones

Viewing process privileges

NFS daemon

```
# ppriv `pgrep nfsd`
357:    /usr/lib/nfs/nfsd
flags = PRIV_AWARE
        E:
basic,!file_link_any,!proc_exec,!proc_fork,!proc_info,!proc_s
ession,sys_nfs
        I:
basic,!file_link_any,!proc_exec,!proc_fork,!proc_info,!proc_s
ession
        P:
basic,!file_link_any,!proc_exec,!proc_fork,!proc_info,!proc_s
ession,sys_nfs
        L:
basic,!file_link_any,!proc_exec,!proc_fork,!proc_info,!proc_s
ession

# pcred `pgrep nfsd`
357:    e/r/suid=1  e/r/sgid=12
```

Coding for privileges (userland)

- See privileges(5)
- `#include <priv.h>`
- Golden Rules:
  - Basic WILL expand in the future
  - Start with basic and remove what you know you don't need, then add what you know you do, drop the rest
- Convience functions in `<priv_util.h>`
  - See `usr/src/cmd/cmd-inet/usr.sbin/ping/ping.c`

Coding with privileges (kernel)

- Drivers:
  - drv_prv(9f) instead of suser().
    - Basically: do I have sys_devices ?
    - May use when device file perms not enough
- Others + priv aware drivers
  - priv_policy(9f) + priv_policy_only(9f) + priv_policy_choice(9f)
- `usr/src/uts/common/os/policy.c`
  - Single place for priv checks, wrapper in secpolicy_*() calls
  - Generally add here if part of OpenSolaris rather than unbundled

RBAC databases

- `exec_attr`: Run this command with this privilege
- `prof_attr`: Grouping mechanism for exec_attr, auth_attr + other keywords
- `auth_attr`: Authorisations
- `user_attr`: Per User policy
- `policy.conf`: System default policy

Using authorisations

- When do I use an authorisation ?
  - When you are running with privilege but not all users should benefit from all of your privilege.
- Simple example: cdrw(1)
  - Forced privelege via setuid, checks authorisation

    `solaris.device.cdrw`
- Complex example: svc.{startd,configd}
  - System daemons check privs of door caller

Checking an authorisation

- Only userland checks authorisations, kernel is only interested in privileges
- `C: chkauthattr(username, authname)`
  - `usr/lib/libsecdb/`
- `Shell: auths(1)`
  - `usr/src/cmd/auths`

Getting other _attr entries

- Things like `lock_after_retries` are stored in user_attr.
- Currently inconsistent on what can be in user_attr/prof_attr. Starting to fix this.
- Current APIs hard to use but public:
  - getuserattr(3secdb) + kva_match(3secdb) + parse policy.conf manually
- Example: `usr/src/cmd/pfexec/pfexec.c`

GSS & SASL

- SASL userland only
  - `usr/src/lib/libsasl`
- GSS user & kernel
  - `usr/src/lib/libgss`
  - `usr/src/cmd/gssd`
  - `usr/src/uts/common/gssapi`
  - `usr/src/uts/common/gssapi/mechs`
  - Some of this should be in `usr/src/common/gssapi`

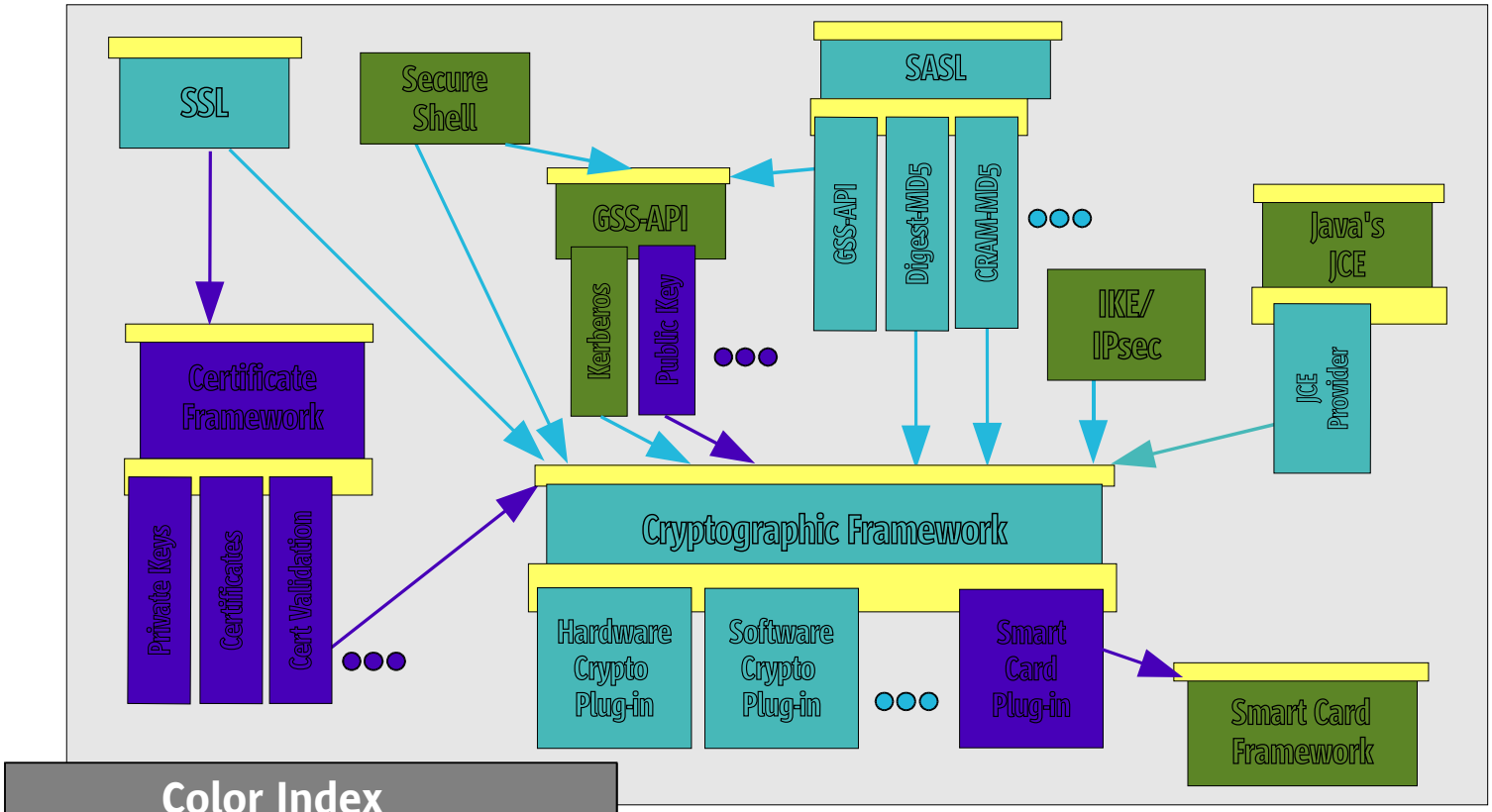
Kerberos Support

- Bundled Kerberos-aware applications
  - Telnet, ftp, rsh, rlogin, rdist (ON)
  - Mozilla (Desktop), Apache (SFW), Secure Shell (ON)
- Enhanced interoperability and security
  - TCP and IPv6 Support
  - AES-128, AES-256, 3DES, RC4-HMAC
- Ease of deployment
  - *kclient* (ON) automated system setup
  - *pam_krb5_migrate (ON)* automated KDC population
- Incremental KDC DB propagation (ON)

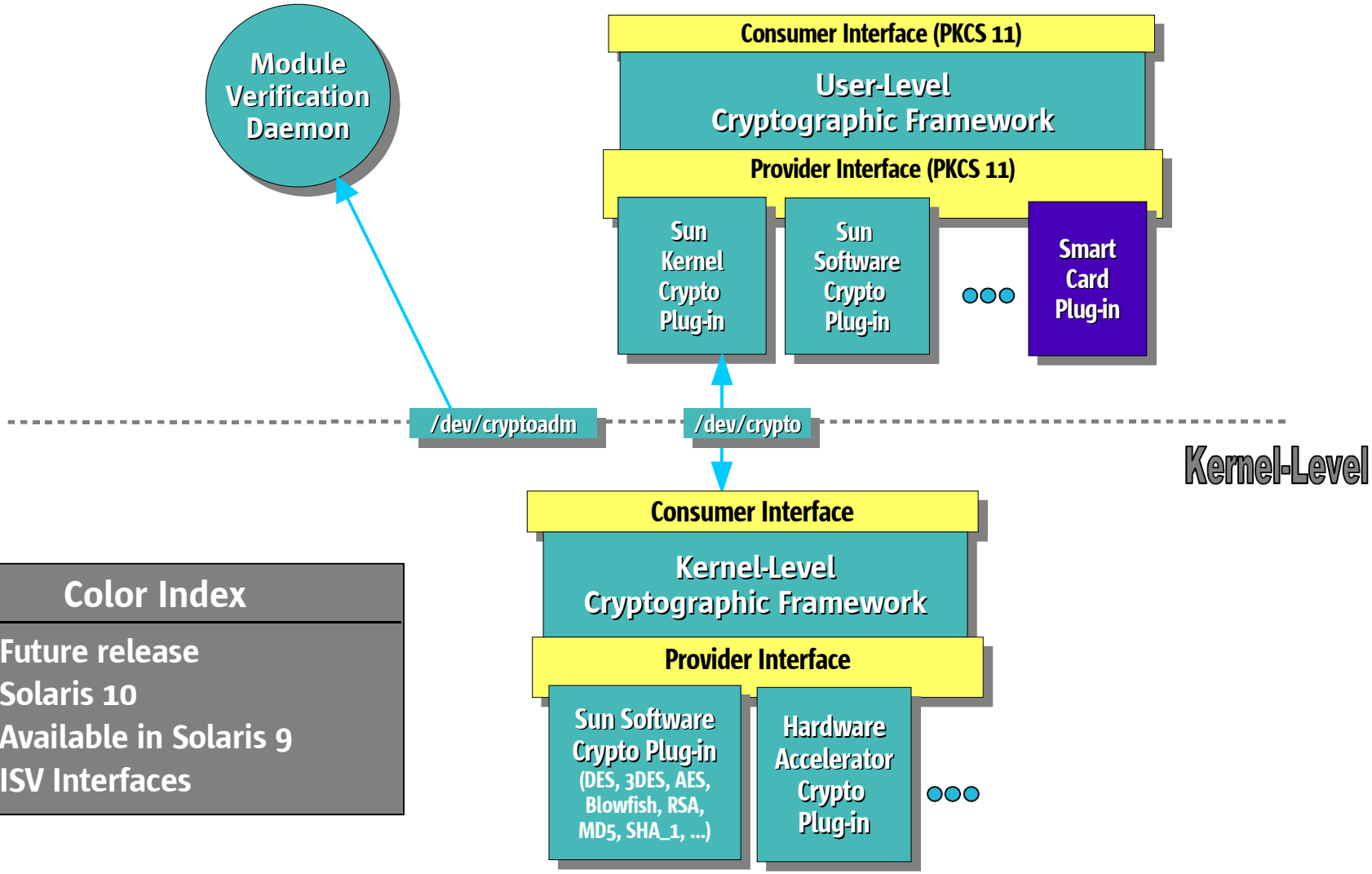
Kerberos

- Programming interfaces only available as a GSS Mech
  - `usr/src/uts/common/gssapi/mechs/krb5`
- Derived from MIT code.
- Uses Crypto Framework (user & kernel)
- Sun PAM module:
  - `usr/src/lib/pam_modules/krb5/`
- Special strmod for in.telnetd/in.rlogind
  - `usr/src/uts/common/io/cryptmod.c`

Network Security Architecture



Cryptographic Framework



Crypto Framework Features

- Standards based pluggable framework
- Userland (PKCS#11)
  - `usr/src/lib/pkcs11/`
  - `usr/src/cmd/cmd-crypto/`
- Kernel
  - `usr/src/common/crypto/{io,core,api}`
- Administrative policy
- OpenSSL ENGINE for PKCS#11
  - Apache mod_ssl uses this by default
- Java JCE provider for PKCS#11 (JDK 1.5)

Userland Crypto Providers

- pkcs11_softtoken.so.1
  - `usr/src/lib/pkcs11/pkcs11_softtoken/`
  - Default PKCS#11 v2.11 software provider
  - DES, 3DES, AES, RC4 (<=128 bit); RSA, DSA, D-H, MD5, SHA-1, SSL HMAC
  - On disk (encrypted) persistant keystore
- pkcs11_softtoken_extra.so.1
  - Supports symmetric algorithms > 128-bit keys
  - Delivered via Encryption Kit in SUNWcry package
- Softtoken supports:
  - Asymmetric algorithms for signing & verification
  - Object and key management

Crypto for Kernel Programmers

- Kernel SPI borrows some PKCS#11 concepts
  - (eg mechanism vs algorithm)
- Sessionless/provider independent interface
- Support of contiguous and scatter-gather data format, inline and different output buffer
- Can be safely called from both process and interrupt context
  - supports synchronous and asynchronous calls

Kernel Crypto Functions

#include <sys/crypto/api.h>

- Functions offered:
  - single and multipart symmetric encryption, digest and MAC, eg:
    - `crypto_encrypt_init(), crypto_encrypt_update()`
    - `crypto_encrypt_final()`
  - encryption & MAC combo (dual ops)
    - `crypto_encrypt_mac()`
  - key checking, context templates
    - `crypto_key_check(), crypto_create_ctx_template()`
  - signup for callback on special events
    - providers changing, flow control, etc...
    - `crypto_notify_events()`

Kernel Crypto Overview

`usr/src/uts/crypto/core`

- Thread pool providing a user context for the execution of asynchronous calls
- Global software queue for asynchronous requests to software providers
- 1 task queue per hardware provider

Kernel Crypto Overview (cont.)

- Multiple hardware providers per mechanism are supported
  - New calls dispatched to the least loaded one
  - Fall back to software providers in case of unavailability of hardware
- Single software provider (per mech) but support for /platform variants of it
- DR-ready SPI
- Providers are loaded on demand

drv/crypto & drv/cryptoadm

usr/src/uts/common/crypto/io

- /dev/crypto is PKCS#11 over ioctls, private interface used by pkcs11_kernel to talk to misc/kcf.
  - `usr/src/uts/common/sys/crypto/ioctl.h`
  - `usr/src/uts/common/crypto/io/crypto.c`
- /dev/cryptoadm is cryptoadm(1m) private interface to misc/kcf
  - `usr/src/uts/common/sys/crypto/ioctladmin.h`
  - `usr/src/uts/common/crypto/io/cryptoadm.c`

Q & A

Contact Info:
Darren.Moffat@Sun.COM
http://blogs.sun.com/darren

sudo vs Solaris RBAC

Cross Platform	N	Y
Kerberos Support	Y[6]	Y
Solaris BSM Audit	Y	N
RUID	Y	Y[9]
EUID	Y	N[9]
RGID	Y	N
EGID	Y	N
Hierarchical Profiles	Y	N[10]
Network Wide Policy	Y [1]	Y [2]
Host Specific Policy	Y [3]	Y [4]
Netgroup Policy	N	Y
Require Password	N[11]	Y
Allow no Password	Y [5]	Y
Cached Authentication	N [6]	Y
Restrict Users	Y	N
Profile Shells	Y	N
Control cmd arguments	N	Y
Privileges/Capabilities Aware	Y	N
Authenticate as Self	N[7]	Y
Control Sensitive Environment Variables	Y[8]	Y
Control UMASK	N	Y
Fine grained Policy Admin	Y	N
Default Profiles for OS Admin	Y	N

Notes

1 All supported Nameservices
2 Assumes “rdist” or LDAP
3 Follows nsswitch: files can override remote nameservice
4 Host/network/netgroup policy in config
5 Not for NIS+ roles
6 When configured for su(1) in pam.conf(4)
7 No for Roles but Yes for just profiles
8 When used as a role su(1) rules apply
9 stay_setuid provides similar functionality
10 Profiles are approximately the same as sudo Cmd_Alias
11 Roles may require a passord[5] profile shells don't