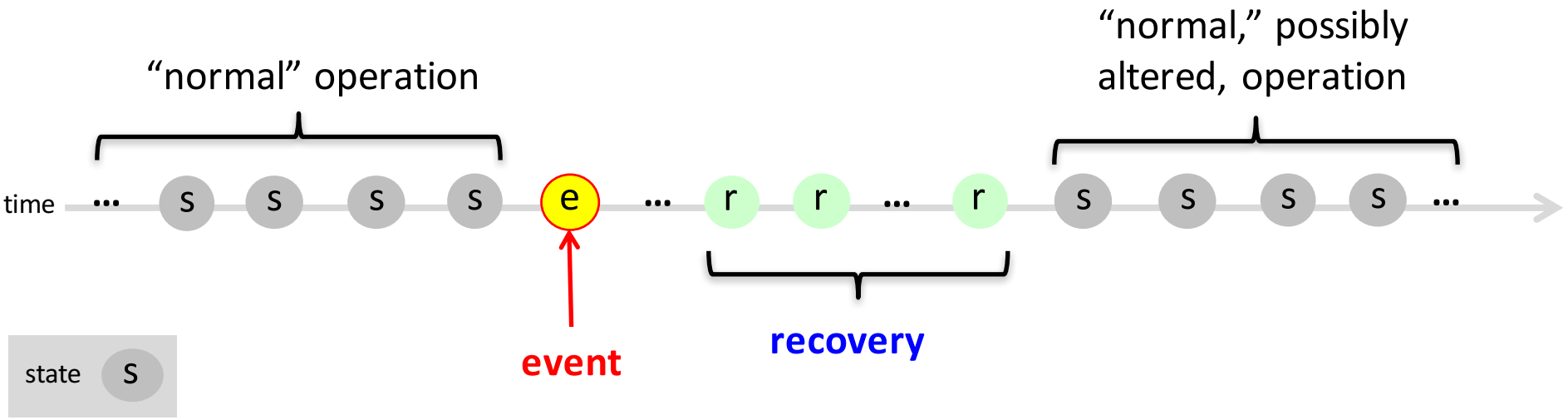
Resilience and System Level Security

July 14, 2016

Lee Badger
Computer Security Division
National Institute of Standards and Technology

Disclaimer: Any mention of a vendor or product is NOT an endorsement or recommendation by NIST.

Resilience, Slightly Structured

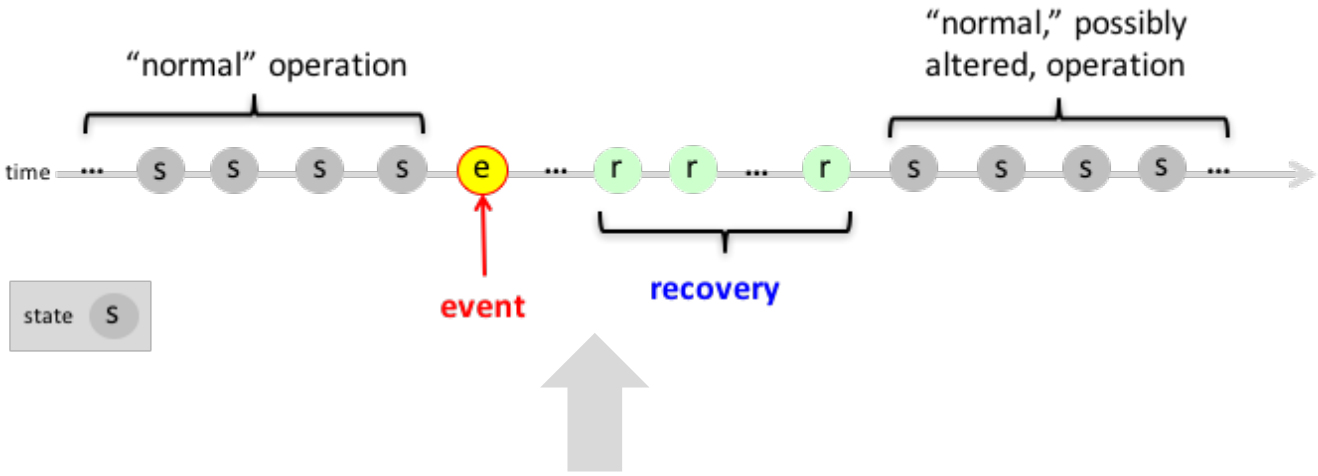


Focusing mostly on the **when**:

- **Proactive** resilience
  - Triggered via non-attack event
  - administratively-imposed or automated
- **Reactive** resilience
  - Triggered by an attack event --- maybe

Proactive Resilience

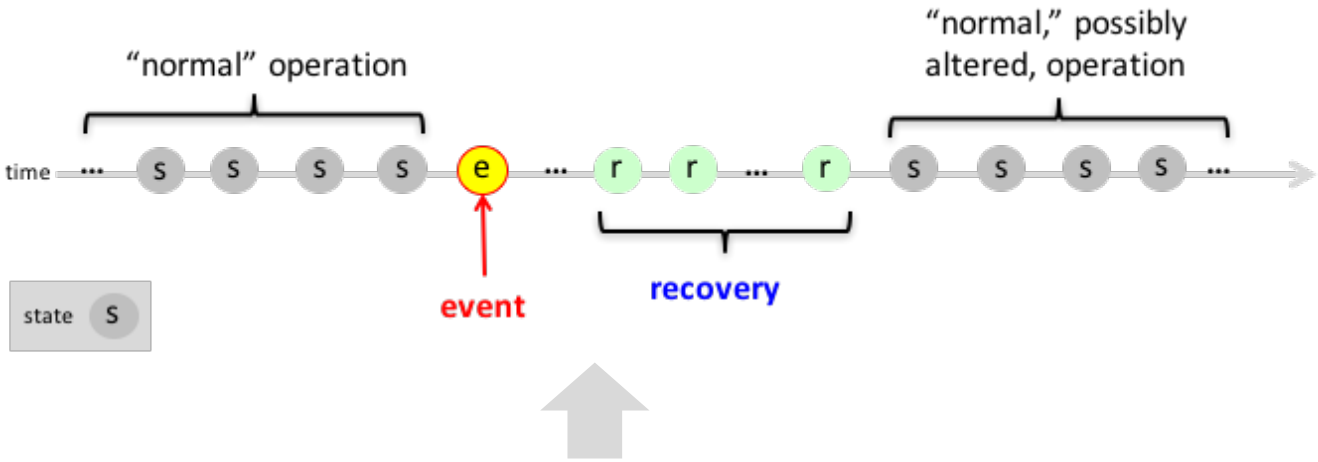
A few examples



- Required updates of authentication credentials
  - Yet another complex password… or RSA token…
  - Or, coming soon, use of the Common Access Card.
- Automated software diversity transforms.
- Error masking.
- Micro-reboot [Candea, Fox].
- Key refresh.
- Software rejuvenation [Trivedi]
- Self-cleaning Intrusion Tolerance [Sood].
- Log file rotation.
- Virtual Machine migration.
- more…

Reactive Resilience

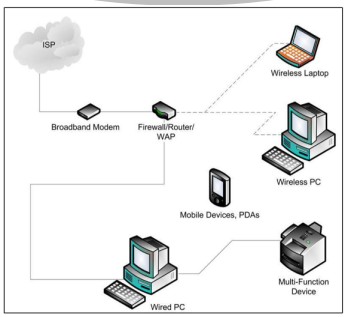
A few Intrusion Detection ideas



- Behavior deviated from a specification
  - How to get the specification
    - Logic induction [Ko], language-assisted [Ko], static analysis [Wagner, Dean]
- Behavior matched a bad pattern (misuse)
  - State Transition Analysis [Ilgun, Kemmerer]
  - Rule-based misuse detection [Lindqvist, Porras]
- Behavior is unusual (and presumed bad)
  - Statistical anomaly on users [IDES system]
  - Frequency distribution changes [Emerald system]
  - Sequence-based anomaly detection [Forrest et al]

The Complexity of Configurations

Standalone

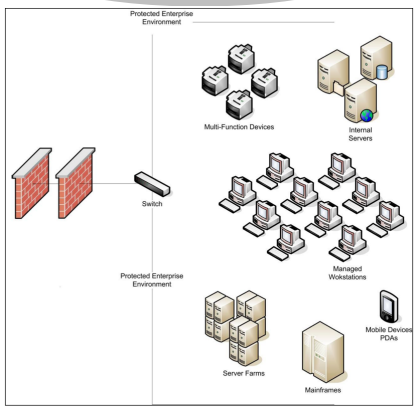


Managed

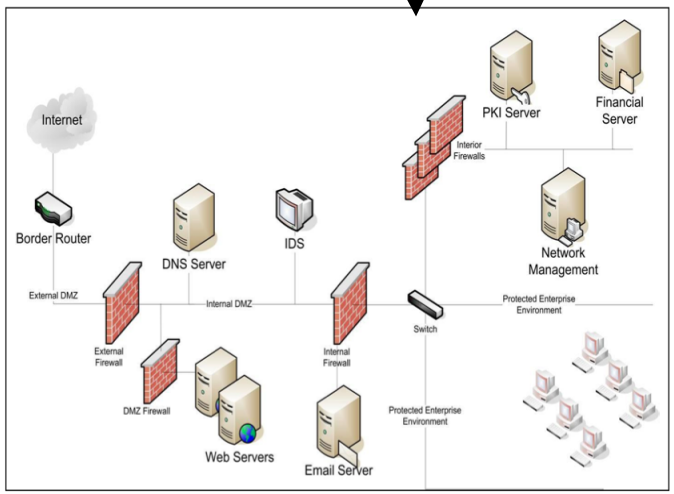
US Gov

Prescribed configurations
Mandated by policy

Legacy



Specialized Security-Limited Functionality



Selected Platforms (http://usgcb.nist.gov)

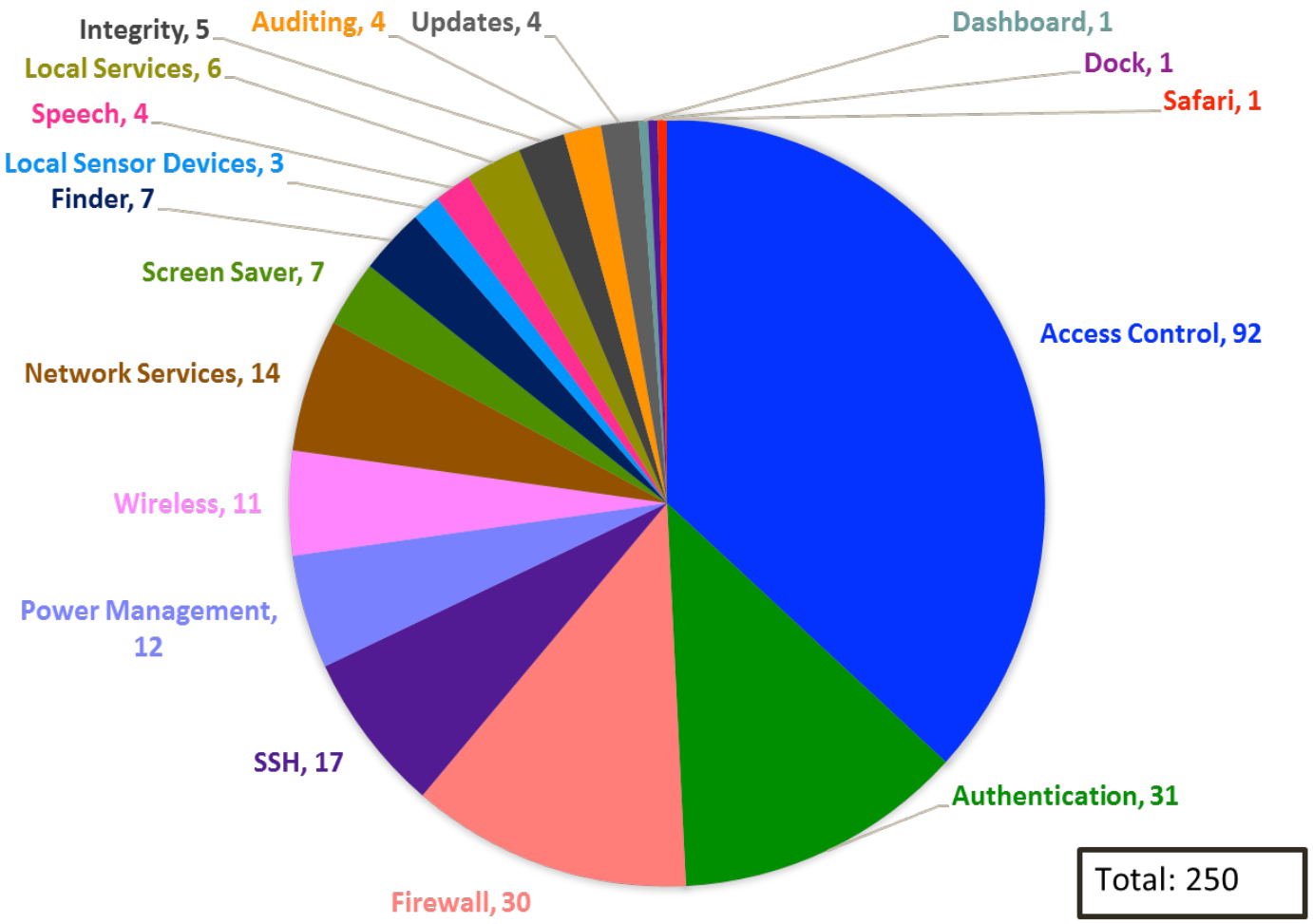
Windows 7	≥ **406** settings
IE8	≥ **114** settings
IE 7	≥ **106** settings
Windows XP	≥ **260** settings
Redhat Linux 5 Desktop	≥ **258** settings
…	

e.g.

```
...
idref="xccdf_gov.nist_rule_credential_validation" selected="true"/>
idref="xccdf_gov.nist_rule_security_group_management" selected="true"/>
idref="xccdf_gov.nist_rule_user_account_management" selected="true"/>
idref="xccdf_gov.nist_rule_process_creation" selected="true"/>
idref="xccdf_gov.nist_rule_logoff" selected="true"/>
idref="xccdf_gov.nist_rule_logon" selected="true"/>
idref="xccdf_gov.nist_rule_special_logon" selected="true"/>
idref="xccdf_gov.nist_rule_file_system" selected="true"/>
idref="xccdf_gov.nist_rule_registry" selected="true"/>
idref="xccdf_gov.nist_rule_audit_policy_change" selected="true"/>
idref="xccdf_gov.nist_rule_authentication_policy_change" selected="true"/>
idref="xccdf_gov.nist_rule_sensitive_privilege_use" selected="true"/>
idref="xccdf_gov.nist_rule_ipsec_driver" selected="true"/>
...
```

Credit: NIST SP 800-70-rev2
National Checklist Program (http://web.nvd.nist.gov/view/ncp/repository)

A Specific Configuration: OS X 10.10 Yosemite



Set individually or In groups.

Interaction between locally-applied and "managed" settings values hard to pin down!

The actual meaning of a setting depends on how reading software interprets it.

Credit: DRAFT NIST Special Publication 800-179 "Guide to Securing Apple OS X 10.10 Systems for IT Professionals: *A NIST Security Configuration Checklist*" csrc.nist.gov

www.tolerantsystems.org

Tolerant Systems

Intrusion Tolerant Systems	Organically Assured and Survivable Information Systems	Self-regenerative Systems	Application Communities	Multi-framework Programming
Jaynarayan Lala Former Program Manager, DARPA	**Jaynarayan Lala** Former Program Manager, DARPA	**Todd Hughes** Program Manager, DARPA	**James Donlon** Program Manager, DARPA	**Lee Badger** Former Program Manager, DARPA
	Lee Badger Former Program Manager, DARPA	**Lee Badger** Former Program Manager, DARPA	**Lee Badger** Former Program Manager, DARPA	

Several DARPA Projects Touching on Resilience

AWDRAT (MIT,Teknowledge)

CORTEX (Honeywell)

DAWSON (GITI)

PMOP (MIT, Teknowledge)

GENESIS (UVA, CMU)

LRTSHS (MIT)

Steward (JHU, Purdue)

VICI (Komoku)

RAMSES (GITI, Stony Brook U.)

CSISM (BBN, Adventium, PACE)

LMRAC (MIT, Determina)

DPASA (BBN)

OASIS Dem/Val

And more……

A few Observations and Idea Sketches

- **Mission/workflow specifications (rules, constraints) facilitated adaptation.**
  - Detection via spec violation is very helpful!
  - Tradeoffs: need to write the specifications.
  - **Idea:** further research in expressing mission/workflows
    - And runtime checking.
    - Big semantic gap.

- **Redundancy with discardable components facilitated service maintenance, provided a chance to adapt.**
  - Enabled fallback, diagnosis of attacks.
  - Components sometimes automatically repairable.
  - **Idea:** apply discardable components approach to modern execution environments
    - Virtual machines, containers, microservices.

- **Secure configurations hard to define and author.**
  - The NIST Secure Content Automation Protocol (SCAP) provides a basis for representing configurations.
    - E.g., see the National Checklist Program (http://www.nist.gov/itl/csd/scm/ncp.cfm)
    - But content authoring is often labor-intensive, skills-intensive, and error-prone.
  - **Idea:** additional research into generative approaches to content creation (e. g. templating, wizards, macros).

System Level Security

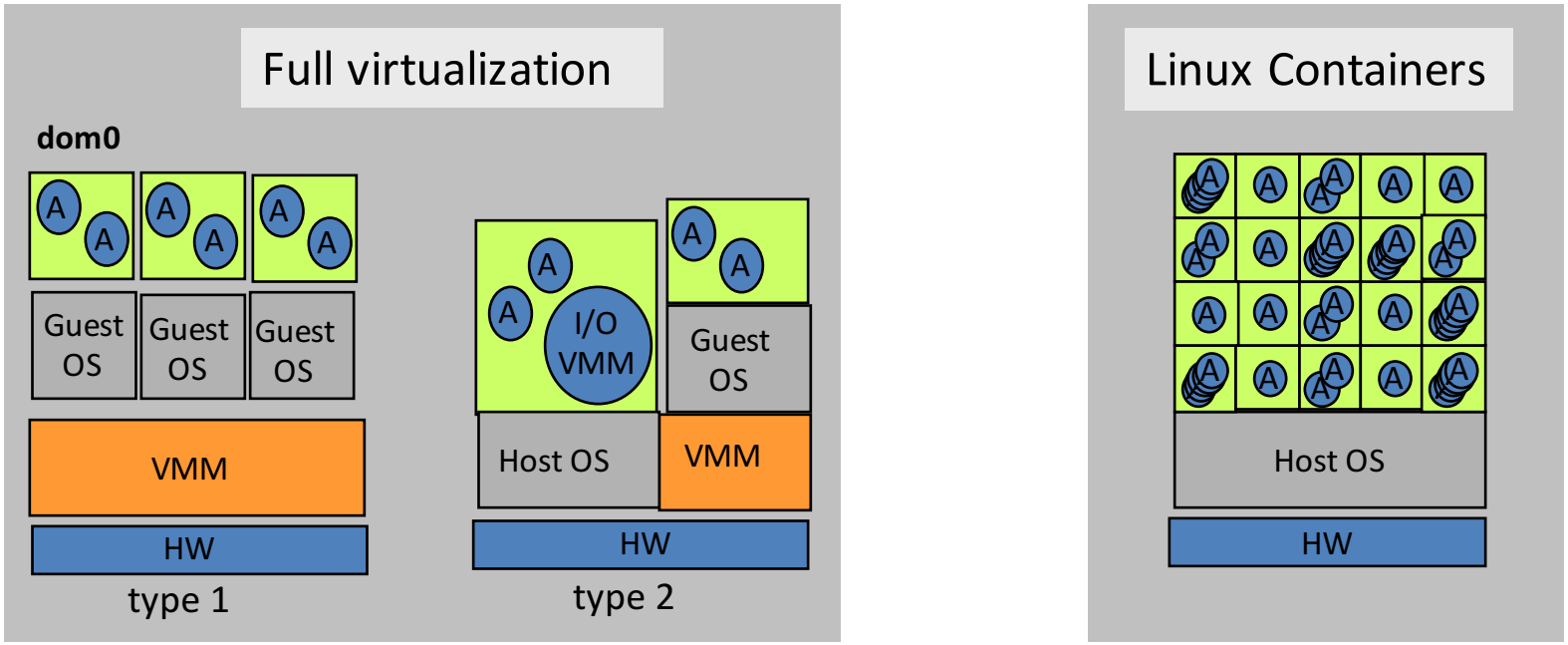
Take advantage of emerging systems architecture patterns to strategically improve assurance.

- Modern software/service packaging strategies are flexible, dynamic, and efficient, but:
- **Isolation is configuration-based.**
- Can assurance be maintained or improved?
- Reasons for both Optimism and Concern.
- Building blocks include: physical machines, physical networks, virtual machines, virtual networks, web browsers, containers, microservices, and more.

- **Operating System Containers**
  - "A container is an object isolating some resources of the host, for the application or system running in it." From the Ubuntu lxc(7) man page.
- **Microservices**
  - "An approach to designing software as a suite of small services, each running in its own process and communicating with lightweight mechanisms." From M. Fowler, "Microservices Architecture", http://martinfowler.com/articles/microservices.html

Virtualization vs Containers

A application



- Ubuntu/vbox5.0.24 base VM: 5,101 M
- Ubuntu base container: 332 M
  - Control groups: namespace, cpu, memory,
  - Name spaces: UTS, IPC, User, PID, Network
  - Device Drivers
  - Configure to "isolate" an application or a system

Control group info from the Ubuntu lxc man page (note: "l" in "lxc" is lowercase L).

Kick the Tires: Installing

From Scratch Installation

```
lbadger@N105745-01:~$ sudo lxc-create -n ubu-c -t ubuntu
[sudo] password for lbadger:
Sorry, try again.
[sudo] password for lbadger:
Checking cache download in /var/cache/lxc/xenial/rootfs-amd64 ...
Installing packages in template: apt-transport-https,ssh,vim,language-pack-en
Downloading ubuntu xenial minimal ...
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id 790BC7277767219C42C86F933B4FE6ACC0B21F32)
I: Retrieving Packages
I: Validating Packages
...
##
# The default user is 'ubuntu' with password 'ubuntu'!
# Use the 'sudo' command to run tasks as root in the container.
##

lbadger@N105745-01:~$
```

Make a new Container: fast

```
lbadger@N105745-01:~$ time sudo lxc-create -n ubu3-c -t ubuntu
...
##
# The default user is 'ubuntu' with password 'ubuntu'!
# Use the 'sudo' command to run tasks as root in the container.
##


real    0m3.046s
user    0m1.980s
sys     0m1.280s
lbadger@N105745-01:~$
```

Kick the Tires: Running

We've made some containers

```
lbadger@N105745-01:~$ sudo lxc-ls --fancy
NAME   STATE   AUTOSTART GROUPS IPV4 IPV6
ubu-c  STOPPED 0         -      -    -
ubu2-c STOPPED 0         -      -    -
ubu3-c STOPPED 0         -      -    -
lbadger@N105745-01:~$
```

Run one of them

```
lbadger@N105745-01:~$ sudo lxc-execute -n ubu-c /bin/bash
init.lxc.static: initutils.c: mount_fs: 36 failed to mount /proc : Device or res
ource busy
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@ubu-c:/# whoami
root
root@ubu-c:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1204     4 ?        Ss   15:05   0:00 /init.lxc.stati
root        23  0.0  0.0  18220  3296 ?        S    15:05   0:00 /bin/bash
root        34  0.0  0.0  34424  2944 ?        R    15:05   0:00 ps aux
root@ubu-c:/# ls
bin   dev  home            lib    media  opt   root  sbin  sys  usr
boot  etc  init.lxc.static  lib64  mnt    proc  run   srv   tmp  var
root@ubu-c:/# exit
lbadger@N105745-01:~$
```

Run a single command in a container (and exit)

```
lbadger@N105745-01:~$ time sudo lxc-execute -n ubu-c echo 'HI-FROM-A-CONTAINER!'
init.lxc.static: initutils.c: mount_fs: 36 failed to mount /proc : Device or res
ource busy
HI-FROM-A-CONTAINER!

real    0m0.642s
user    0m0.008s
sys     0m0.008s
lbadger@N105745-01:~$
```

Complex Configuration

- Architecture
- Hostname
- Halt signal
- Reboot signal
- Stop signal
- Init command
- Init id
- Psedo ttys
- Console
- /dev dir
- Mount points

- Root fs
- Avail syscalls
- Control group
- Network
  - Type
  - Link
  - Mtu
  - Name
  - Hwadr
  - Ipv4
  - Ipv4 gateway
  - Ipv6
  - Ipv6 gateway

- Lifecycle hooksx
- Logging

Info from ‘man lxc.container.conf’

A few Observations and Idea Sketches

- **Container configurations are highly expressive, but easy to get wrong**
  - Configuration templates and change tracking already being addressed: e.g., Docker, LXC templates
  - **Idea:** further research in semantically checking container configurations; e.g., a container "lint" utility.

- **Lightweight containers can promote the principle of least privilege.**
  - "The Protection of Information in Computer Systems", J. Saltzer, M. Shroeder.
    - Economy-of-mechanism, fail-safe-defaults, complete-mediation, open-design, separation-of-privilege, least-privilege, least-common-mechanism, psychological-acceptability
  - **Idea:** develop analysis techniques/tools to generate custom containers that approximate least-privilege for important classes of programs.

Microservices

- **Microservices**
  - "An approach to designing software as a suite of small services, each running in its own process and communicating with lightweight mechanisms." From M. Fowler, "Microservices Architecture", http://martinfowler.com/articles/microservices.html

- Not really a new idea:
  - Remember web services?
  - Remember the Mach microkernel or GNU HURD?

- But some goals do appear to be different:
  - Services should be easy to replace.
    - So connective protocols need to be simple.
  - Services should implement business capabilities.
  - Services should have their own refresh cycles.
  - Services should be programming-language agnostic.

Credit: info from martinfowler.com/articles/microservices.html

A “Hello World” Microservice”

```
# hello.py

from nameko.rpc import rpc

class GreetingService(object):
    name = “greeting_service”

    @rpc
    def hello(self, name):
        return “Hello, {}!”.format(name)
```

Import the necessary framework.

Define the service.

Decorator exposes the function that implements the service.

Return a string to the client.

- This example is from: nameko.readthedocs.io/en/stable/index.html.
- Nameko is one of numerous frameworks that can be used.
- Used here for convenience because it’s simple Python, and open source.

A “Hello World” Microservice”

```
# hello.py

from nameko.rpc import rpc

class GreetingService(object):
    name = “greeting_service”

    @rpc
    def hello(self, name):
        return “Hello, {}!”.format(name)
```

Import the necessary framework.

Define the service.

Decorator exposes the function that implements the service.

Return a string to the client.

- This example is from: nameko.readthedocs.io/en/stable/index.html.
- Nameko is one of numerous frameworks that can be used.
- Used here for convenience because it’s simple Python, and open source.

But this would be too simple...

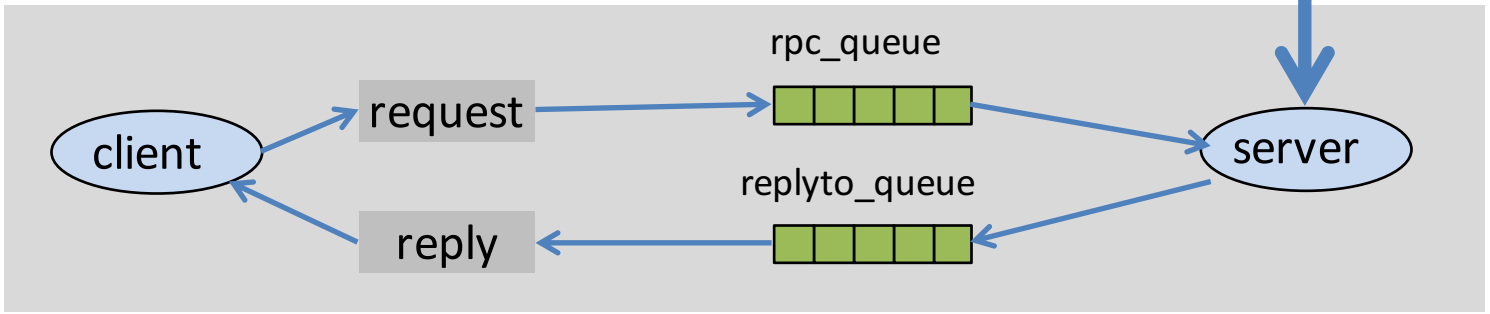
Under the Hood: Queuing

```
# hello.py

from nameko.rpc import rpc

class GreetingService(object):
    name = “greeting_service”

    @rpc
    def hello(self, name):
        return “Hello, {}!”.format(name)
```



From: www.rabbitmq.com/tutorials/tutorial-six-python.html

- Nameko depends on rabbitmq, an open source queuing framework.

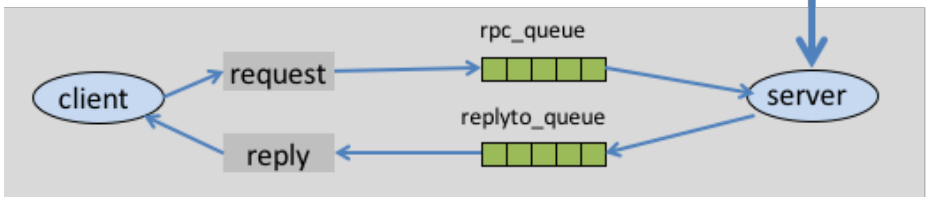
Kick the Tires: Microservices

```
# hello.py

from nameko.rpc import rpc

class GreetingService(object):
    name = "greeting_service"

    @rpc
    def hello(self, name):
        return "Hello, {}!".format(name)
```



Launch service

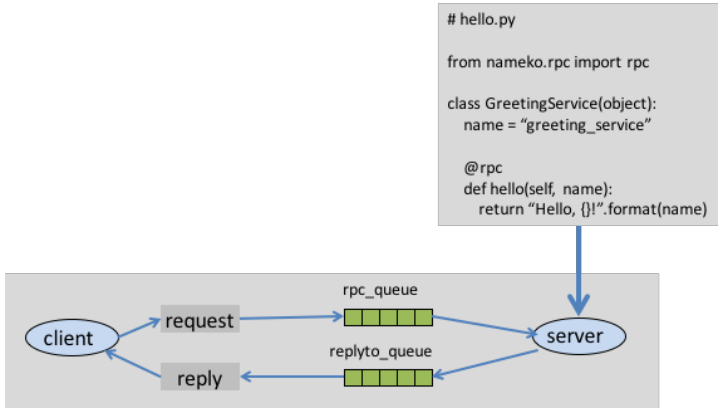
```
lbadger@N105745-01:~/Documents/dev/nameko$ nameko run hello
starting services: greeting_service
Connected to amqp://guest:**@127.0.0.1:5672//
```

Access service

```
lbadger@N105745-01:~$ nameko shell
Nameko Python 2.7.11+ (default, Apr 17 2016, 14:00:29)
[GCC 5.3.1 20160413] shell on linux2
Broker: amqp://guest:guest@localhost
>>> n.rpc.greeting_service.method('My Master')
u'Hello, My Master!'
>>>
```

Note: the rabbitmq queuing system must already be running: start it with the "rabbitmq-server" command.

Some Achievable Properties



- Decoupling of logic from computing resources.
- Explicit inter-service interface specifications.
  - Support Saltzer/Shroeder principles
- Independent update cycles.

```
from nameko.rpc import rpc, RpcProxy

class Service(object):
    name = "service"

    other_rpc = RpcProxy("another service")

    @rpc
    def hello(self):
        pass
```

- A dependency on another microservice.
- Dynamically linked when a "worker" object is created.
- A worker object exists only for the duration of a single method's execution.
  - (in the nameko framework)
- This is a form of "software rejuvenation".
  - (the concept that restarting software components clears out some bugs)

Credit: nameko.readthedocs.io/en/stable/key_concepts.html

A few Observations and Idea Sketches

- **Trusted Microservices**
  - Properly formulated, could some services (and their messaging fabrics) be viewed as Reference Monitors?
    - Concept from the Anderson Report in the 1970s: always invoked, tamperproof, verified.
  - **Idea:** research aspects of microservices interfaces and interactions and how assurance arguments could (or could not) be constructed for systems implemented with microservices.

- **Interposition-based Enhancements**
  - Interposition on the right interfaces can augment, transform, deny, or monitor uses of the interfaces.
    - However, interposition can also destabilize systems, and impose slowdowns.
  - **Idea:** research interposition strategies that are compatible with microservices-based systems.

Thanks