CapNet: Security and Least Authority in a Capability-Enabled Cloud

Anton Burtsev
University of California, Irvine

David Johnson, Josh Kunz, Eric Eide, Jacobus Van der Merwe
University of Utah
Modern clouds are vulnerable
Endpoints are inherently vulnerable
PoCs for my CVEs can be found on my github:
https://github.com/ScottyBauer/Android_Kernel_CVE_POCs

All bugs have been found via manual code review. I don't fuzz, I don't run tools I just look at the code.

List of my CVEs:

CVE-2017-11015 (October bulletin)
CVE-2017-11013 (October bulletin)
CVE-2017-9714 (October bulletin)
CVE-2017-11014 (October bulletin)
CVE-2017-0740
CVE-2017-9680
CVE-2017-8259
CVE-2017-0705
CVE-2016-5861
CVE-2016-10274
CVE-2016-5857
CVE-2016-5856
CVE-2016-5855
CVE-2016-5854
CVE-2016-5853
CVE-2017-0576
CVE-2017-0339
CVE-2017-0562
CVE-2017-0521
CVE-2017-0504
CVE-2017-0516
CVE-2017-0451
CVE-2017-0405
CVE-2016-6693 (shared with Seven Shen from Trend Micro Mobile Th
CVE-2016-3936
CVE-2016-3928
CVE-2016-3902
CVE-2016-3937
CVE-2016-6696
CVE-2016-6516

Submitted my first bug last year this month. Since then I've found 32 kernel bugs, gotten 32 CVEs and have made around 40k in bug bux.

6:48 PM - 30 Nov 2016

After ~3 months of work I have 2 confirmed remote Pixel Kernel bugs. Deets in 90ish days. Current mitigation -- throw your technology away.

11:22 PM - 1 May 2017
Broad network authority

Cloud network is the main attack amplifier
Legacy network-isolation primitives

- Global tenant-wide access control rules
  - E.g., security groups
- Lack of mutual isolation
- Lack of decentralized access control
  - Need to trust a third party

Ambient authority
Capability-enabled network
CapNet Architecture
Threat model

• We trust
  • Cloud provider infrastructure
    • Network switches
    • SDN controller
    • Hypervisors
    • Cloud software stack

• Hosts are malicious
  • Virtual and physical hosts on the network
  • Providers of third-party cloud services
CapNet Architecture

- Software defined network (SDN)
- CapNet runs as an SDN controller application
  - Tracks resources of the network
- By default nodes are completely isolated
  - No flows are allowed
Objects and capabilities

class object:
    def foo(cap):
        ...

(Capability).foo(...) = object.foo(...)

CapNet Architecture

• On the host, capabilities are just 64-bit numbers
  • Have no meaning outside of the host

• CapNet associates a Node object with each host on the network
  • Unique {switch, port} pair

• Capabilities are resolved through Node’s CSpace into pointers to other objects
CapNet

Objects

Node
grant reset()
object create(object_type type, specification spec = ()

Flow

RendezvousPoint
void send(cap c, string msg)
(cap, string) recv(int timeout)

Grant
any invoke(cap c, method m, args args)

Membrane
cap wrap(cap c)
void clear()

SealerUnsealer
cap seal(cap c)
cap unseal(cap c)

Operations on capabilities
cap mint(cap c, specification spec = ()
void revoke(cap c)
void delete(cap c)
Nodes

- Node is "born" with one special capability, \( rp0 \), connecting it to creator.
RendezvousPoints

- RendezvousPoints allow Nodes exchange capabilities
- Capability derivation trees (CDT)
Flows

- A unidirectional communication channel
- The ability to send packets to a particular network endpoint
Grant

```java
cap c, method m, args
```

- Support for legacy capability-oblivious hosts
- Passive administration
Grant

```
Grant.invoke(cap c, m, args)
Grant.grant(cap c)
Grant.take(capability_id cap_id)
```
Grant

invokes the `cap c, m, args` function.

Grant.grant(cap c)

Grant.take(capability_id cap_id)

grant.create(Flow)
Convenient network programming

• Example: connecting two nodes A and B

1. connect (cap gantA, cap grantB)
2. flowA = grantA.create(Flow)
3. flowB = grantB.create(Flow)
4. grantA.grant(flowB)
5. grantA.grant(flowA)
Decentralized Authority and Collaboration
Reset

- Reset the node to a clean, isolated state irrespective of its prior state and ownership
Reset

- Tracking and cleaning authority of the node
Reset preserves ownership

revoke([node])
reset([node])
Recursive isolation of capability graphs
Membranes

\[ rp = (m).wrap(rp) \]
Membranes

\[
\begin{align*}
\text{node} & \to (\text{rp}).\text{send}(\text{node}) \\
\text{node} & \to (\text{rp}).\text{recv}() \\
\text{grant} & \to (\text{node}).\text{reset}() \\
\text{flow} & \to (\text{grant}).\text{create}(\text{FLOW})
\end{align*}
\]
Membranes

( ).destroy()
SealersUnsealers

\[
\text{su}.\text{seal(r)}
\]

\[
\text{mint(r)}
\]

\[
\text{rp}.\text{send(r)}
\]

\[
\text{rp}.\text{recv}()
\]

\[
\text{f}()
\]

\[
\text{su}.\text{unseal(r)}
\]
SealersUnsealers go through membranes

\[(rp\).send(su)\]

\[su = (rp\).recv()\]
Protocols of Secure Collaboration
Secure provider protocol
Recursion
Trees and general graphs

- Membranes and reset allow the construction of trees in capability graphs
Trees and general graphs

• SealerUnsealer enable cloud topologies that are general graphs
Joint computation protocol
CapNet in OpenStack
Thank you!

Anton Burtsev aburtsev@uci.edu

Paper: SoCC’17

Source: https://gitlab.flux.utah.edu/tcloud/capnet

Test drive in CloudLab: https://www.cloudlab.us/p/TCloud/OpenStack-Capnet
Backup slides
CapNet Objects

- Node – hosts on the network
- RendezvousPoint – exchange of capabilities
- Flow – network flows
- Grant – support for unmodified hosts
- Membrane – transitive isolation of capability graphs
- SealerUnsealer – secure transport of capabilities