ESEC/FSE Visions and Reflections 2020

Nov. 10th, 2020

Inferring and Securing Software Configurations Using Automated Reasoning

Paul Gazzillo
University of Central Florida

https://pgazz.com @paul_gazzillo

UNIVERSITY OF CENTRAL FLORIDA

highly-configurable software is widespread

Linux kernel

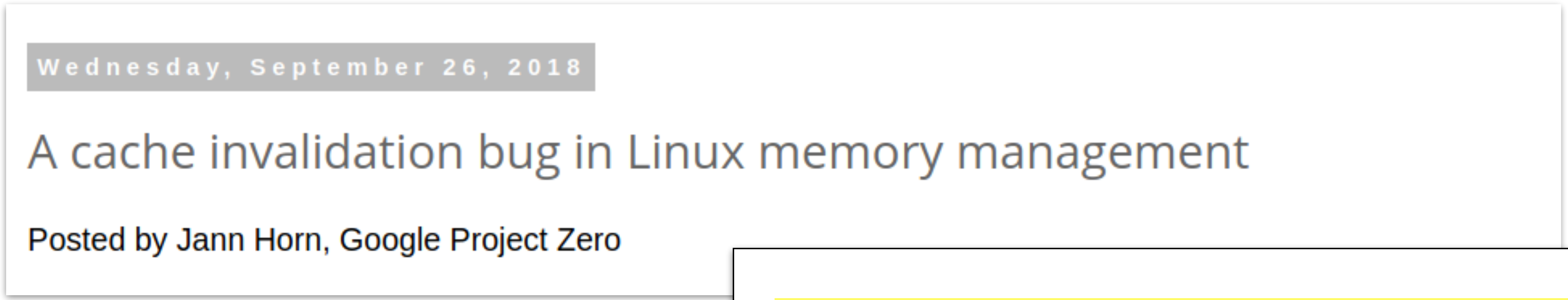
- 70% of mobile devices
- 70% of IoT developers
- 40% of servers

Apache web server

- 40% of servers

billions of devices

misconfiguration vulnerabilities are prevalent



Wednesday, September 26, 2018

A cache invalidation bug in Linux memory management

Posted by Jann Horn, Google Project Zero

"This exploit shows how much impact the kernel configuration can have on how easy it is to write an exploit for a kernel bug."

#6 in OWASP top ten most critical security risks

most common risk reported

misconfiguration vulnerabilities are rooted in software configuration management

manages *change* to a software system

allows customizing software without reprogramming

falls outside of classic program analysis

vision: a world without misconfiguration vulnerabilities

solution: formal methods to validate and generate software configurations

challenges: a lack of existing specifications,
an enormous state space

research goals

create a rigorous definition of configuration specification

mechanize the generation of valid configurations

automatically discover secure configurations

Motivating Example: Optionsbleed

a Limit directive restricts access to HTTP methods in an Apache webserver

```
<Limit PUT DELTE BIND>
</Limit>
```

optionsbleed leaks arbitrary memory contents of an apache webserver

invalid http method exposes a use-after-free bug

```
<Limit PUT DELTE BIND>
</Limit>
```

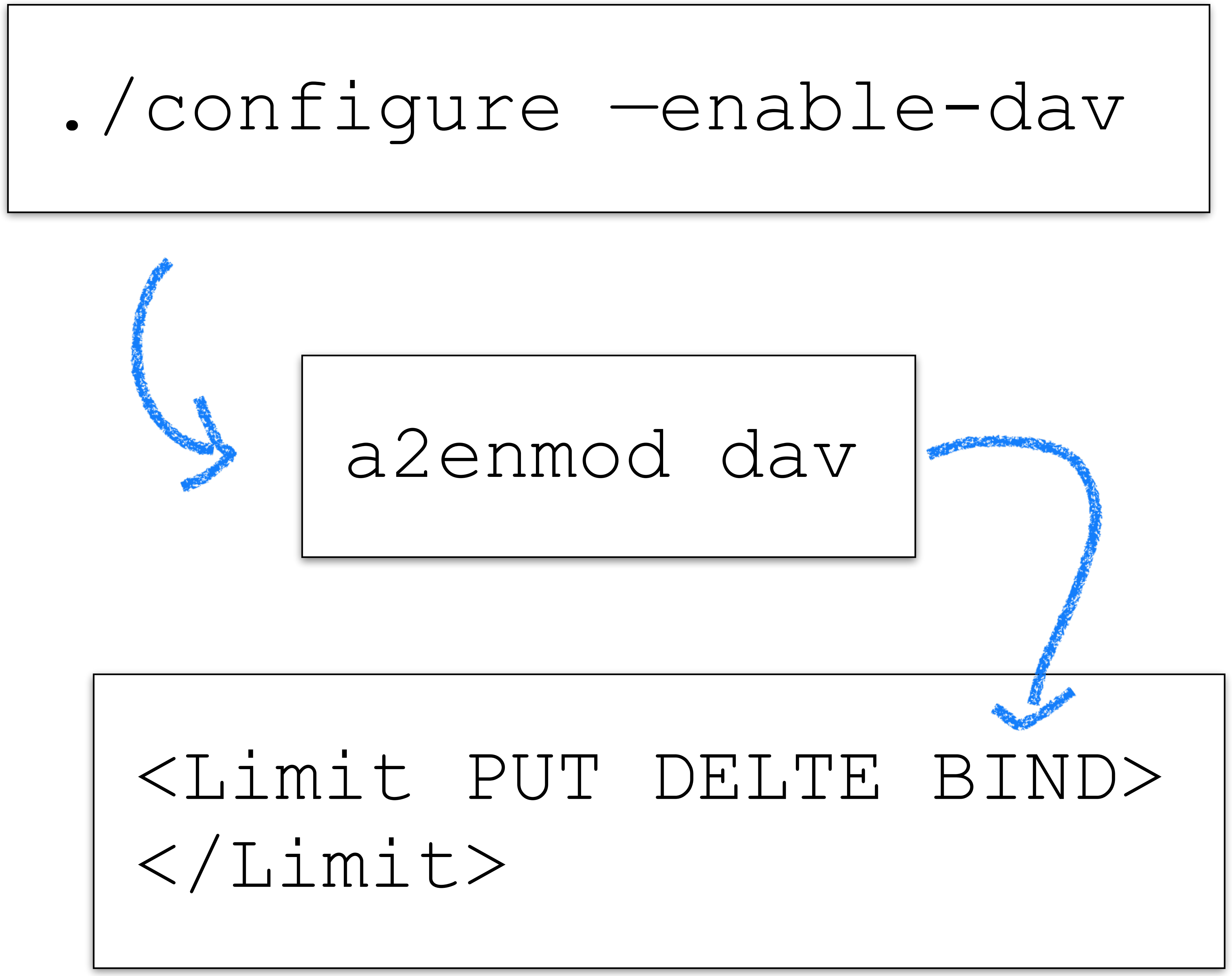
subtle interactions between configuration mechanisms influence optionbleed's occurrence

```
<Limit PUT DELTE BIND>
</Limit>
```

BIND is only valid with the WebDAV HTTP extension

subtle interactions between configuration mechanisms influence optionbleed's occurrence

WebDAV is enabled only with a compile-time flag and run-time module loader



```
./configure —enable-dav
```

```
a2enmod dav
```

```
<Limit PUT DELTE BIND>
</Limit>
```

Solution Approach: Automatically Validate and Generate Software Configurations

automation needs a unified global view of configuration specifications

configuration options are long-lived values, global to an entire software system

formalize valid configurations as constraints among all configuration options

build

```
./configure —enable-dav
```

module

```
a2enmod dav
```

limit

```
<Limit PUT DELTE BIND>
</Limit>
```

formalize valid configurations as constraints among all configuration options

```
      limit.method = PUT
or    limit.method = DELETE
or   (limit.method = BIND
      and build.enable-dav = True
      and module.dav = True)
```

build

```
./configure —enable-dav
```

module

```
a2enmod dav
```

limit

```
<Limit PUT DELTE BIND>
</Limit>
```

formalize valid configurations as constraints among all configuration options

```
    limit.method = PUT
or  limit.method = DELETE
or (limit.method = BIND
    and build.enable-dav = True
    and module.dav = True)
```

configuration validitity is satisfiability

build

```
./configure —enable-dav
```

module

```
a2enmod dav
```

limit

```
<Limit PUT DELTE BIND>
</Limit>
```

research tasks

an intermediate configuration language

formal modeling and analysis

testing and bug-finding

security and prevention

conclusion

- highly-configurable software is widespread
- misconfiguration vulnerabilities are prevalent
- vision: a world without misconfiguration
- challenges: lack of real-world specification, an enormous configuration space
- solution approach: formal modeling of software configuration

supported by NSF CCF-1941816

https://pgazz.com
@paul_gazzillo