



HOUSE: Marco de trabajo modular de arquitectura escalable y desacoplada para el uso de técnicas de *fuzzing* en HPC

Francisco Borja Garnelo Del Río, Francisco J. Rodríguez Lera,
Gonzalo Esteban Costales, Camino Fernández Llamas,
Vicente Matellán Olivera

Universidad de León

Índice de contenidos

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

1 Introducción


2 Trabajo relacionado

3 Marco de trabajo HOUSE

4 Resultados y discusión

5 Conclusiones





Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Sección 1

Introducción

Motivación

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Definición: (*fuzzing* [Sutton et al., 2007])

La automatización de generación y prueba de entradas malformadas en el software con el fin de encontrar comportamientos no esperados en el mismo.

- Habitualmente fallos repentinos y completos de un sistema o componente informático conocidos como crashes [IEEE Communications Society, 1990].

Definición: (HPC, high-performance computing)

Computación de alto rendimiento o supercomputación.



Motivación

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

■ Los problemas identificados en la literatura:

- 1 Complejidad de modificar una parte sin tener que montar algo complejo para ello (técnicas y herramientas).
- 2 Monopolio de las herramientas y confidencialidad de los datos.
- 3 Dificultad, especialización y puesta en marcha (aplicación práctica).



Objetivos

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- 1 Conocer el estado actual del *fuzzing* enfocándolo a su uso en entornos HPC.
- 2 Dar una visión global de las actividades previas, los procesos y componentes del *fuzzing* respecto a las últimas mejoras y las adaptaciones a entornos de computación escalable.





Introducción

**Trabajo
relacionado**

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Sección 2

Trabajo relacionado

Tipos de pruebas de seguridad en el software [Takanen et al., 2018]

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- **Activas o dinámicas**, implican la ejecución total o parcial del software.
 - **Proactivas**, automatizan la ejecución de forma instrumentada con herramientas de *fuzzing* (BFF, radamsa o AFL).
 - **Reactivas**, observan el comportamiento del software en un entorno de producción (SELinux o EPDR).
- **Pasivas o estáticas**, no implican la ejecución del software, habitualmente emplean el código fuente (CodeQL, Sonarqube, Yasca o Flawfinder).



Tipos de fuzzing [Takanen et al., 2018]

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- Se clasifican en función de los detalles de la implementación disponible.
 - *white-box* fuzzing, parte del código fuente.
 - *black-box* fuzzing, utiliza el software ya compilado en formato binario.
- Entre estos dos tipos se situaría un tercero:
 - *grey-box* fuzzing, combina características de ambos, como es el caso de las pruebas donde se utiliza instrumentación en el código fuente y el análisis BVA.





Introducción

Trabajo
relacionado

**Marco de
trabajo HOUSE**

Resultados y
discusión

Conclusiones

Sección 3

Marco de trabajo HOUSE

HOUSE

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Marco de trabajo con un enfoque funcional que facilita la coherencia y continuidad de forma global abordando el *fuzzing* como un flujo con distintas **fases**, organizando todos los procesos implicados, incluyendo también aquellos con relación directa.

- Modular.
- Abierto.
- Agnóstico a las herramientas y técnicas.

Fases del fuzzing

Introducción

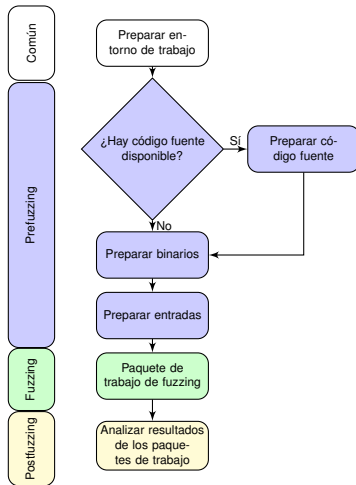
Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- 1** *Común o de preparación del entorno de trabajo.*
- 2** *Prefuzzing o de preparación de las pruebas software.*
- 3** *Fuzzing o de ejecución de las propias pruebas.*
- 4** *Postfuzzing o de análisis de resultados y tareas posteriores.*



Módulos de HOUSE

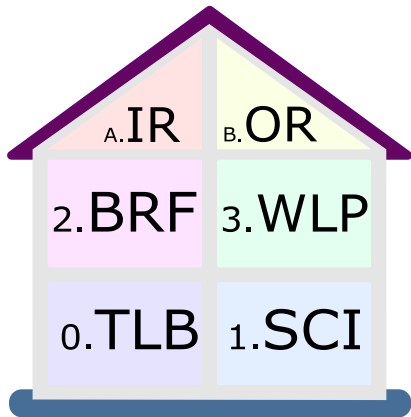
Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones



El nombre de HOUSE hace referencia a la clásica metáfora [Gerbrand van Dieijen, 2010] de describir un proceso como el diseño y construcción de una casa y, en este caso, describe el proceso de desarrollo seguro del software, pero desde el punto de vista del *fuzzing*.

Módulos de HOUSE

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Input Repository (A.IR)

Repositorio de entradas como semillas para generadores de entradas, diccionarios, archivos de muestra, etc.

Output Repository (B.OR)

Repositorio de salidas con los contextos de ejecuciones de los *fuzzers*, resultados, logs, etc.

Tool Box (O.TLB)

Colección de herramientas y utilidades auxiliares a cualquiera de las fases del *fuzzing*.



Módulos de HOUSE

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Source Code Integration (1.SCI)

Colección de códigos fuente, incluyendo parches, integraciones y optimizaciones previas al *fuzzing*.

Binaries Ready to Fuzz (2.BRF)

Colección de binarios preparada para el *fuzzing*.

WorkLoad Package (3.WLP)

Colección de paquetes con cargas de trabajo de cualquiera de las fases.



Flujo de trabajo: BPMN

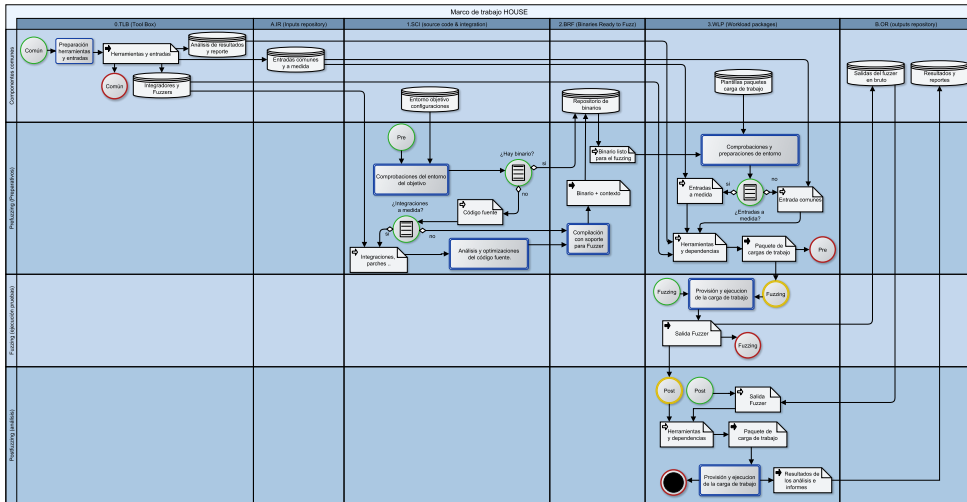
Introducción

Trabajo relacionado

Marco de trabajo HOUSE

Resultados y discusión

Conclusiones



Flujo de trabajo: Fase 1 - Común

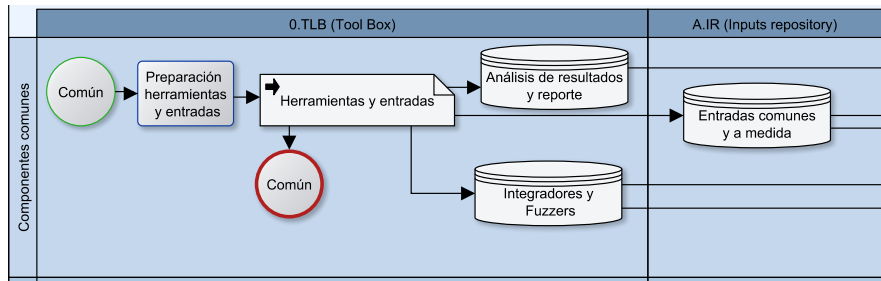
Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones



JNIC
2021
LIVE

Flujo de trabajo: Fase 2 - Prefuzzing

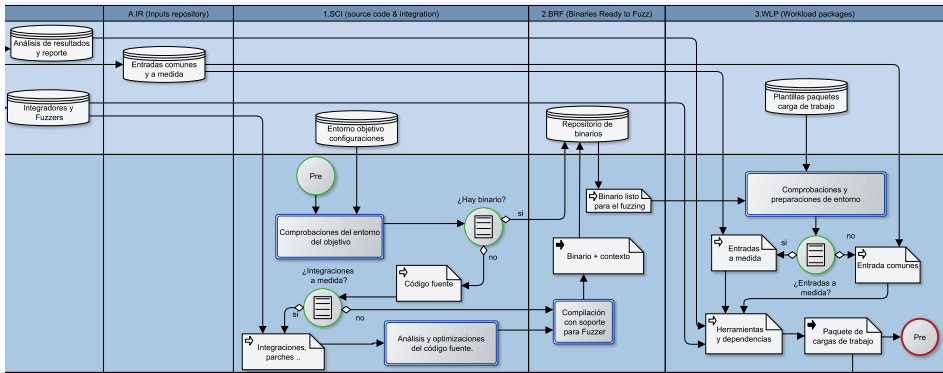
Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones



Flujo de trabajo: Fase 3 - *Fuzzing*

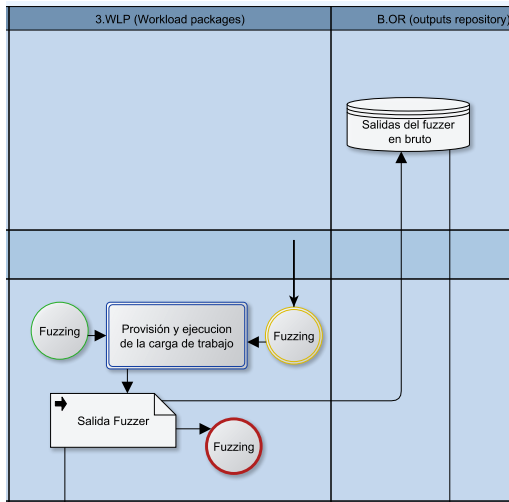
Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones



Flujo de trabajo: Fase 4 - *Postfuzzing*

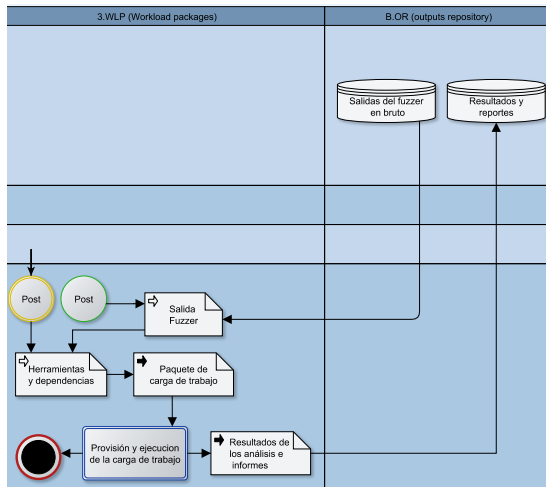
Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones





Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

**Resultados y
discusión**

Conclusiones

Sección 4

Resultados y discusión

Prueba de concepto - Entorno

Introducción

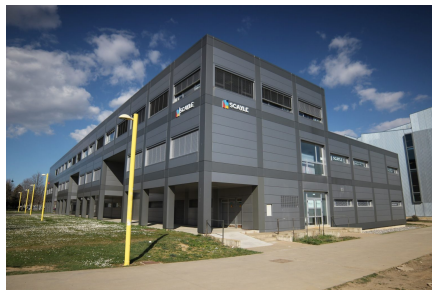
Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- HPC Caléndula.
 - Arquitecturas Intel, Cascade Lake y Haswell.
 - Sistema operativo CentOS 7.7
 - Políticas de seguridad y requisitos del **ENS**.
 - Slurm como sistema de gestión de cargas de trabajo.
- Limitaciones y características.
 - 50 trabajos máximo.
 - Nodos estándar del HPC.
 - Nodo debug.



Prueba de concepto - Características y logros

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- Desarrollo de scripts para Slurm para gestionar con cargas de trabajo las distintas fases.
- Implementar casos de uso básicos con el *fuzzer* AFL++.
- Instrumentación con afl-gcc.
- Tipo *fuzzing*: *white-box* de cobertura de código.
- PUT: `date` y `expr` de la colección de utilidades coreutils.
- Caso básico de control: Buffer Overflow.
- Entrada de datos para automatizaciones: Entrada estándar (stdin)
- Minimizar o prescindir de adaptación de los nodos o crear nuevos perfiles software.



Prueba de concepto - Dificultades

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- Las principales dificultades identificadas son:
 - Limitaciones de uso de herramientas basadas en ASAN en 64bits.
 - Monitorización de *crashes* usando ABRT.
 - Librerías dinámicas del sistema.
 - Rutas estáticas definidas en compilación.
 - Protecciones de seguridad sistema operativo.



Prueba de concepto - Experimentos

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- El desarrollo de la investigación se han probado otros enfoques:
 - Uso de ASAN para binarios de 32bits.
 - Junyper notebook como gestor de cargas de trabajo medio de playbooks en python.
 - Uso de contenedores singularity.
 - Uso chroot en espacio de usuario con PRoot - Termux
 - Uso de *modules* para gestión del entorno del sistema operativo.



Prueba de concepto - Estado y antecedentes

Introducción

Trabajo relacionado

Marco de trabajo HOUSE

Resultados y discusión

Conclusiones

- Inicialmente se trató de replicar el experimento [Cioce et al., 2019] usando instrumentación con GCC adaptado a un entorno HPC moderno, y una versión derivada de AFL reciente.
- Actualmente se está trabajando en la adecuación de instrumentación LLVM y se planea añadir el soportar otras arquitecturas con QEMU.

```
CALENDULA [root@node01:~]# ./status.sh

Individual fuzzers
=====
-- FUZZER --
>>> name: MF987802 state: RUNNING (0 days, 0 hrs) node: cn3064 PID: 10441 JOBID: 987802 <<<

Fuzzer activity metrics:

last_path      : none seen yet
last_crash     : none seen yet
last_hang      : none seen yet
cycles_wo_finds : 225
cycle 227, lifetime speed 496 execs/sec, path 0/1 (0%)
pending 0/0, coverage 0.14%, no crashes yet

Performance metrics:

AveCPU AveDiskRead AveDiskWrite AveRSS AveVMSize MaxDiskRead MaxDiskWrite MaxRSS MaxVMSize
-----
00:00.000 453399 71987 4554K 418232K 453399 71987 4554K 418232K

-- FUZZER --
>>> name: SF987803 state: RUNNING (0 days, 0 hrs) node: cn3064 PID: 17885 JOBID: 987803 <<<

Fuzzer activity metrics:

last_path      : none seen yet
last_crash     : none seen yet
last_hang      : none seen yet
cycles_wo_finds : 220
cycle 221, lifetime speed 505 execs/sec, path 0/1 (0%)
pending 0/0, coverage 0.14%, no crashes yet

Performance metrics:

AveCPU AveDiskRead AveDiskWrite AveRSS AveVMSize MaxDiskRead MaxDiskWrite MaxRSS MaxVMSize
-----
00:00.000 448135 67339 4543K 418260K 448135 67339 4543K 418260K

Summary stats
=====
Fuzzers alive : 2
Total run time : 3 minutes, 51 seconds
Total execs : 115 thousands
Cumulative speed : 1001 execs/sec
Average speed : 500 execs/sec
Pending paths : 0 faves, 0 total
Pending per fuzzer : 0 faves, 0 total (on average)
Crashes found : 0 locally unique
Cycles without finds : 225/220
Time without finds : 0
```





Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

Sección 5

Conclusiones

Conclusiones

Introducción

Trabajo
relacionado

Marco de
trabajo HOUSE

Resultados y
discusión

Conclusiones

- El *fuzzing* está cobrando cada vez más protagonismo como actividad para garantizar la calidad y la seguridad del software.
- Grandes empresas y organizaciones se suman al *fuzzing* para poner en valor sus desarrollos o servicios de forma comercial.
- Los desarrollos con intereses privados y servicios comerciales alrededor del *fuzzing* provocan dependencia tecnológica y suponen un riesgo a la confidencialidad de la información para su adopción en un ciclo de desarrollo seguro.
- HOUSE propone un modelado del flujo de trabajo del *fuzzing* caracterizado por tener un enfoque abierto y agnóstico a las herramientas.
- HOUSE permite resolver o simplificar las tareas más tediosas y complejas relacionadas con la aplicación de técnicas de *fuzzing*.




Gracias por su atención



<https://github.com/b0rh/HOUSE>

Referencias I

-  Cioce, C. R., Loffredo, D. G., and Salim, N. J. (2019).
Program Fuzzing on High Performance Computing Resources.
Technical Report SAND2019-0674, 1492735.
-  Gerbrand van Dieijen (2010).
Metaphors in software development.
-  IEEE Communications Society (1990).
IEEE Standard Glossary of Software Engineering Terminology.
Office, 121990(1):1.
-  Sutton, M., Greene, A., and Amini, P. (2007).
Fuzzing: Brute Force Vulnerability Discovery.
Addison-Wesley, Upper Saddle River, NJ.



Referencias II



Takanen, A., DeMott, J., Miller, C., and Kettunen, A., editors (2018). *Fuzzing for Software Security Testing and Quality Assurance*. Artech House Information Security and Privacy Series. Artech House, Norwood, MA, 2 ed. edition.

