定向覆盖模糊测试工具的设计与实现

毕业设计中期检查

雷尚远

南京邮电大学计算机学院

2023 年 4 月 17 日

What Fuzzing is?

Defination[1]

- **Fuzzing** Fuzzing is the execution of the PUT using input(s) sampled from an input space (the “fuzz input space”) that protrudes the expected input space of the PUT.
  - PUT: Program Under Test
- **Fuzz testing** Fuzz testing is the use of fuzzing to test if a PUT violates a correctness policy.
- **Fuzzer** A fuzzer is a program that performs fuzz testing on a PUT.
- **Bug Oracle** A bug oracle is a program, perhaps as part of a fuzzer, that determines whether a given execution of the PUT violates a specific correctness policy.
- **Fuzz Configuration** A fuzz configuration of a fuzz algorithm comprises the parameter value(s) that control(s) the fuzz algorithm.
- **Seed** A seed is a (commonly well-structured) input to the PUT, used to generate test cases by modifying it.

What Fuzzing is?

Fuzz Testing

Input: $\mathbb{C}$, $\mathtt{t}_{\text{limit}}$
Output: $\mathbb{B}$ // a finite set of bugs

1 $\mathbb{B} \leftarrow \varnothing$
2 $\mathbb{C} \leftarrow$ **Preprocess**($\mathbb{C}$)
3 **while** $\mathtt{t}_{\text{elapsed}} < \mathtt{t}_{\text{limit}} \wedge$ **Continue**($\mathbb{C}$) **do**
4 conf $\leftarrow$ **Schedule**($\mathbb{C}$, $\mathtt{t}_{elapsed}$, $\mathtt{t}_{limit}$)
5 tcs $\leftarrow$ **InputGen**(*conf*)
// $\mathtt{O}_{\text{bug}}$ is embedded in a fuzzer
6 $\mathbb{B}'$, execinfos $\leftarrow$ **InputEval**(*conf*, *tcs*, $\mathtt{O}_{bug}$)
7 $\mathbb{C} \leftarrow$ **ConfUpdate**($\mathbb{C}$, *conf*, *execinfos*)
8 $\mathbb{B} \leftarrow \mathbb{B} \cup \mathbb{B}'$
9 **return** $\mathbb{B}$

Fuzzing Algorithm

```
1  Input: ℂ, t_limit
2  Output: 𝔹 // a finite set of bugs
3  𝔹 ← ∅
4  ℂ ← Preprocess(ℂ)
5  while t_elapsed < t_limit ∧ Continue(ℂ) do
6      conf ← Schedule(ℂ, t_elapsed, t_limit)
7      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
8      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
9      ℂ ← ConfUpdate(ℂ, conf, execinfos)
10     𝔹 ← 𝔹 ∪ 𝔹′
11 return 𝔹
```

- $\mathbb{C}$:a set of fuzz configurations
- t_{limit}: timeout
- $\mathbb{B}$: a set of discovered bugs

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1 𝔹 ← ∅
2 ℂ ← Preprocess(ℂ)
3 while t_elapsed < t_limit ∧ Continue(ℂ) do
4     conf ← Schedule(ℂ, t_elapsed, t_limit)
5     tcs ← InputGen(conf)
      // O_bug is embedded in a fuzzer
6     𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7     ℂ ← ConfUpdate(ℂ, conf, execinfos)
8     𝔹 ← 𝔹 ∪ 𝔹′
9 return 𝔹
```

Preprocess $(\mathbb{C}) \rightarrow \mathbb{C}$

- **Instrumentation**
  - grey-box and white-box fuzzers
  - static/dynamic
- **Seed Selection**
  - weed out potentiallyredundant configurations
- **Seed Trimming**
  - reduce the size of seeds
- **Preparing a Driver Application**
  - library Fuzzing, kernal Fuzzing

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1  𝔹 ← ∅
2  ℂ ← Preprocess(ℂ)
3  while t_elapsed < t_limit ∧ Continue(ℂ) do
4      conf ← Schedule(ℂ, t_elapsed, t_limit)
5      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
6      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7      ℂ ← ConfUpdate(ℂ, conf, execinfos)
8      𝔹 ← 𝔹 ∪ 𝔹′
9  return 𝔹
```

Stop Condition

- $t_{elapsed} < t_{limit}$
- CONTINUE $(\mathbb{C}) \rightarrow \{True, False\}$
  - Determine whether a new fuzz iteration should occur

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1  𝔹 ← ∅
2  ℂ ← Preprocess(ℂ)
3  while t_elapsed < t_limit ∧ Continue(ℂ) do
4      conf ← Schedule(ℂ, t_elapsed, t_limit)
5      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
6      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7      ℂ ← ConfUpdate(ℂ, conf, execinfos)
8      𝔹 ← 𝔹 ∪ 𝔹′
9  return 𝔹
```

SCHEDULE ($\mathbb{C}$, $t_{elapsed}$, t_{limit}) $\rightarrow$ conf

- **function**
  - selecting a new fuzz configuration
- **depend on fuzzer**
  - finding the most number of unique bugs
  - maximizing the coverage
  - etc.
- **problem**
  - *exploration vs. exploitation*

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1  𝔹 ← ∅
2  ℂ ← Preprocess(ℂ)
3  while t_elapsed < t_limit ∧ Continue(ℂ) do
4      conf ← Schedule(ℂ, t_elapsed, t_limit)
5      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
6      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7      ℂ ← ConfUpdate(ℂ, conf, execinfos)
8      𝔹 ← 𝔹 ∪ 𝔹′
9  return 𝔹
```

InputGen (conf)→ tcs

- **function**
  - Generate testcases
- **classification**
  - Generation-based(*Model-based*)
  - Mutation-based(*Model-less*)

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1  𝔹 ← ∅
2  ℂ ← Preprocess(ℂ)
3  while t_elapsed < t_limit ∧ Continue(ℂ) do
4      conf ← Schedule(ℂ, t_elapsed, t_limit)
5      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
6      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7      ℂ ← ConfUpdate(ℂ, conf, execinfos)
8      𝔹 ← 𝔹 ∪ 𝔹′
9  return 𝔹
```

InputEval (conf, tcs, O_{bug}) $\rightarrow \mathbb{B}'$, execinfos

- **function**
  - Executes the PUT with the tcs and get execinfos.
  - Triage:analyzing and reporting test cases
  - deduplication, prioritization, and minimization.
- Continue ($\mathbb{C}$) $\rightarrow$ {True, False}
  - Determine whether a new fuzz iteration should occur

Fuzzing Algorithm

Input: $\mathbb{C}$, t_{limit}
Output: $\mathbb{B}$ // a finite set of bugs

1 $\mathbb{B} \leftarrow \varnothing$
2 $\mathbb{C} \leftarrow$ `Preprocess`($\mathbb{C}$)
3 **while** $t_{elapsed} < t_{limit} \wedge$ `Continue`($\mathbb{C}$) **do**
4 `conf` $\leftarrow$ `Schedule`($\mathbb{C}$, $t_{elapsed}$, t_{limit})
5 `tcs` $\leftarrow$ `InputGen`(*conf*)
// O_{bug} is embedded in a fuzzer
6 $\mathbb{B}'$, `execinfos` $\leftarrow$ `InputEval`(*conf*, *tcs*, O_{bug})
7 $\mathbb{C} \leftarrow$ `ConfUpdate`($\mathbb{C}$, *conf*, *execinfos*)
8 $\mathbb{B} \leftarrow \mathbb{B} \cup \mathbb{B}'$
9 **return** $\mathbb{B}$

INPUTEVAL (conf, tcs, O_{bug}) $\rightarrow \mathbb{B}'$, execinfos

- $t_{elapsed} < t_{limit}$
- CONTINUE ($\mathbb{C}$) $\rightarrow$ {True, False}
  - Determine whether a new fuzz iteration should occur

Fuzzing Algorithm

```
Input: ℂ, t_limit
Output: 𝔹 // a finite set of bugs
1  𝔹 ← ∅
2  ℂ ← Preprocess(ℂ)
3  while t_elapsed < t_limit ∧ Continue(ℂ) do
4      conf ← Schedule(ℂ, t_elapsed, t_limit)
5      tcs ← InputGen(conf)
       // O_bug is embedded in a fuzzer
6      𝔹′, execinfos ← InputEval(conf, tcs, O_bug)
7      ℂ ← ConfUpdate(ℂ, conf, execinfos)
8      𝔹 ← 𝔹 ∪ 𝔹′
9  return 𝔹
```

stop condition

- $t_{elapsed} < t_{limit}$
- CONTINUE $(\mathbb{C})$ $\rightarrow \{True, False\}$
  - Determine whether a new fuzz iteration should occur

Fuzzing Algorithm

Input: $\mathbb{C}$, t_{limit}
Output: $\mathbb{B}$ // a finite set of bugs

1 $\mathbb{B} \leftarrow \varnothing$
2 $\mathbb{C} \leftarrow$ `Preprocess`$(\mathbb{C})$
3 **while** $t_{elapsed} < t_{limit} \wedge$ `Continue`$(\mathbb{C})$ **do**
4 `conf` $\leftarrow$ `Schedule`$(\mathbb{C}, t_{elapsed}, t_{limit})$
5 `tcs` $\leftarrow$ `InputGen`$(conf)$
  // O_{bug} is embedded in a fuzzer
6 $\mathbb{B}'$, `execinfos` $\leftarrow$ `InputEval`$(conf, tcs, O_{bug})$
7 $\mathbb{C} \leftarrow$ `ConfUpdate`$(\mathbb{C}, conf, execinfos)$
8 $\mathbb{B} \leftarrow \mathbb{B} \cup \mathbb{B}'$
9 **return** $\mathbb{B}$

stop condition

- $t_{elapsed} < t_{limit}$
- CONTINUE $(\mathbb{C})$
  $\rightarrow$ {True, False}
  - Determine whether a new fuzz iteration should occur

Why Grey-box Fuzzing ?

classification of fuzzing

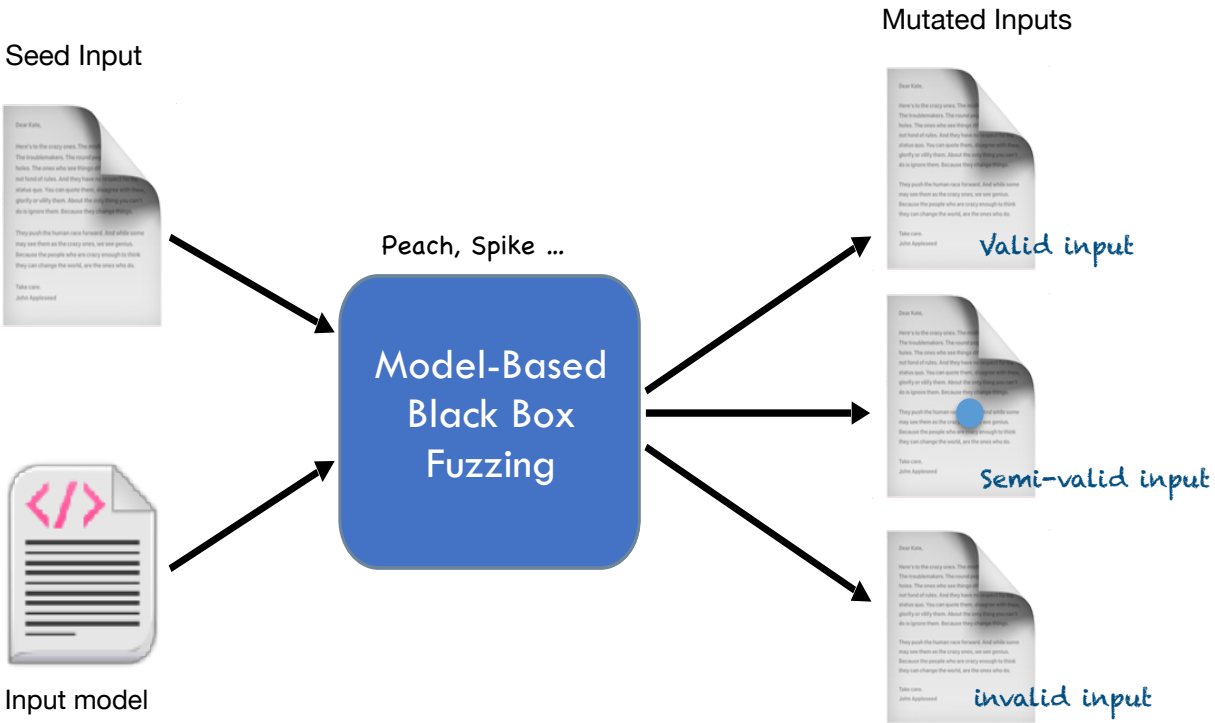
- **Black-box Fuzzing**
  - no program analysis, no feedback
- **White-box Fuzzing**
  - mostly program analysis
- **Grey-box Fuzzing**
  - no program analysis, but feedback

Why Grey-box Fuzzing ?

- Black-box Fuzzing

  Defination: techniques that do not see the internals of the PUT,and can observe only the input/output behavior of the PUT, treating it as a black-box[1].

  -No program analysis, no feedback

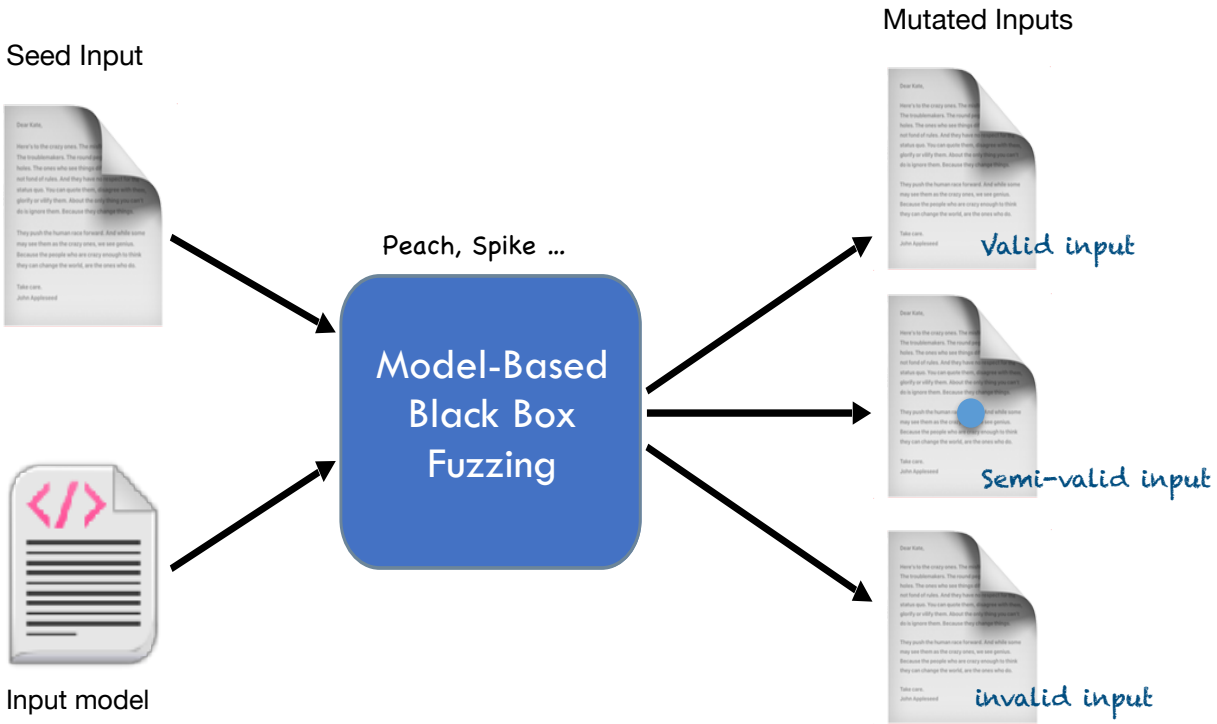


Why Grey-box Fuzzing ?

- Black-box Fuzzing

  Defination: techniques that do not see the internals of the PUT,and can observe only the input/output behavior of the PUT, treating it as a black-box[1].

  - No program analysis, no feedback



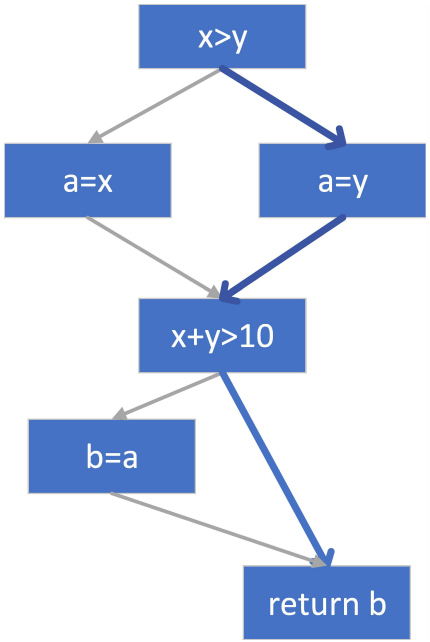
- You have no view of the PUT,but have some view of the input/output domain
- Fuzzing process is not changed according to some feedback
- Random mutated (not effective)

Why Grey-box Fuzzing ?

- White-box Fuzzing

  Defination: techniques that generates test cases by analyzing the internals of the PUT and the information gathered when executing the PUT[1].

  - Requires heavy-weight program analysis and constraint solving.



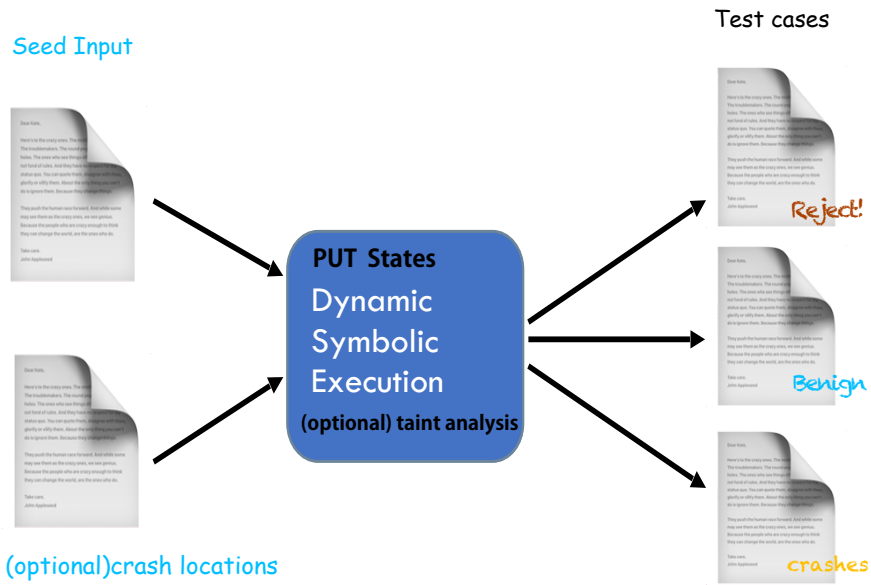
Cover more paths

$x \le y \wedge x+y \le 10$

$x \le y \wedge \neg x+y \le 10$

$\neg x \le y$

Static Analysis
- **Symbolic Excution**
- **Constrains Satisfation iterative algorithm**

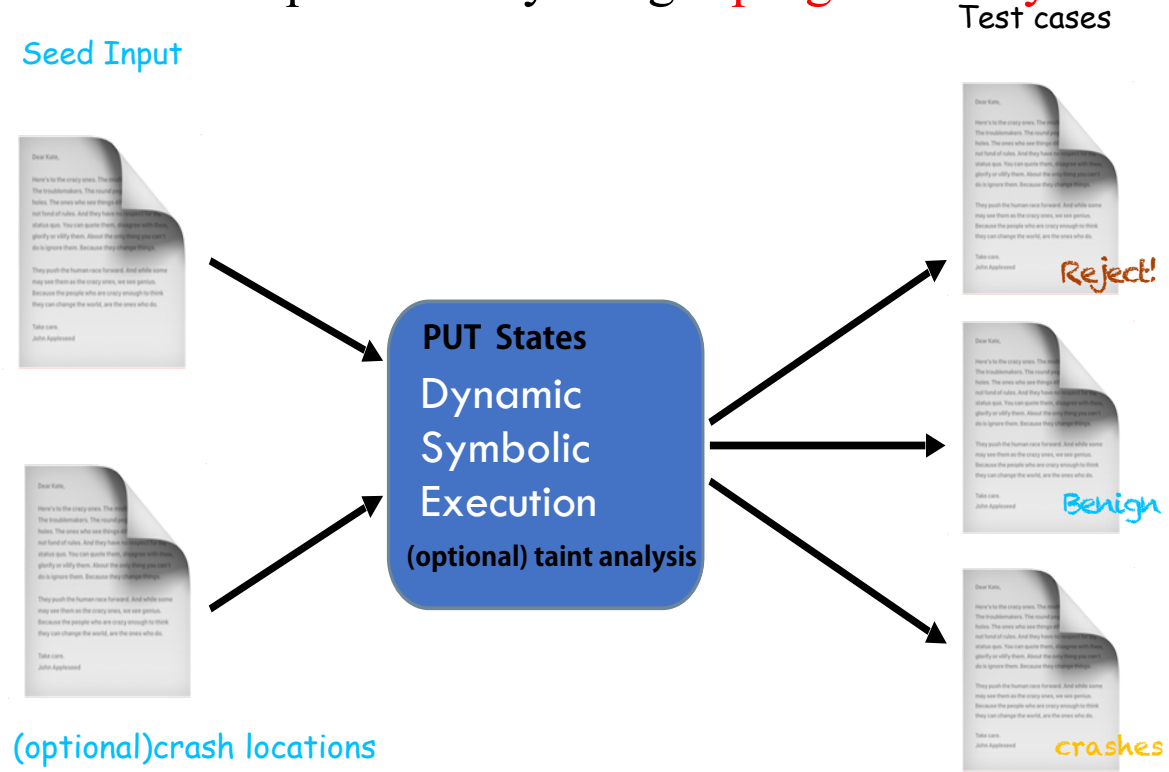


Why Grey-box Fuzzing ?

- White-box Fuzzing

  Defination: techniques that generates test cases by analyzing the internals of the PUT and the information gathered when executing the PUT[1].

  - Requires heavy-weight program analysis and constraint solving.



- You have the view of the PUT state(CFG,CG)
- Static analysis (effective but not efficient!)

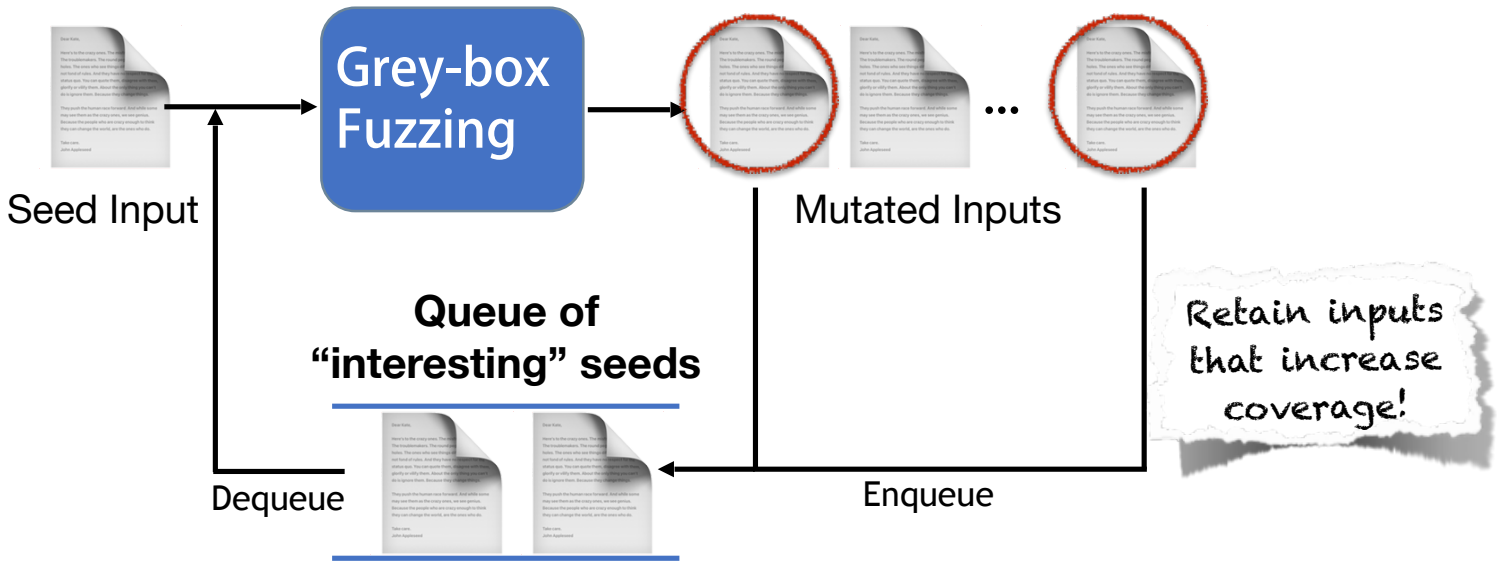
Why Grey-box Fuzzing ?

- Grey-box Fuzzing

  Defination: techniques that can obtain *some* information internal to the PUT and/or its executions to generates test cases[1].

  - Uses only lightweight instrumentation to glean some program structure

  - And coverage feedback



Why Directed Grey-Box Fuzz?

- 大家都会 LaTeX，好多学校都有自己的 Beamer 主题

Why Directed Grey-Box Fuzz?

- 大家都会 LaTeX，好多学校都有自己的 Beamer 主题
- 中文支持请选择 XeLaTeX 编译选项

- 有一些 LaTeX 自带的
- 有一些 Tsinghua 的
- 本模板来源自 THU Beamer Theme
- 但是最初的 link [2] 已经失效了
- 这是原作者在 16-17 年做的一些 ppt：戳我

这一份主题与原始的 THU Beamer Theme 区别在于

- 顶栏的小点变成一行而不是多行
- 中文采用楷书
- 修改了主题色为南邮校徽颜色
- 参考文献格式按照毕设标准进行了修改
- 更多该模板的功能可以参考
  `https://www.latexstudio.net/archives/4051.html`
- 下面列举出了一些 Beamer 的用法，部分节选自
  `https://tuna.moe/event/2018/latex/`

Why Beamer

- LaTeX 广泛用于学术界，期刊会议论文模板

Microsoft® Word	LaTeX
文字处理工具	专业排版软件
容易上手，简单直观	容易上手
所见即所得	所见即所想，所想即所得
高级功能不易掌握	进阶难，但一般用不到
处理长文档需要丰富经验	和短文档处理基本无异
花费大量时间调格式	无需担心格式，专心作者内容
公式排版差强人意	尤其擅长公式排版
二进制格式，兼容性差	文本文件，易读、稳定
付费商业许可	自由免费使用

排版举例

无编号公式

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t] = \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s,a)$$

多行多列公式[1]

$$\begin{aligned} Q_{\text{target}} &= r + \gamma Q^\pi(s', \pi_\theta(s') + \epsilon) \\ \epsilon &\sim \text{clip}(\mathcal{N}(0,\sigma), -c, c) \end{aligned} \tag{1}$$

[1] 如果公式中有文字出现，请用 \mathrm{} 或者 \text{} 包含，不然就会变成 clip，在公式里看起来比 clip 丑非常多。

编号多行公式

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(a^2 + \Big(a^2 + 2a\Delta x + (\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 2a\Delta x + 2^2(\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 3a\Delta x + 3^2(\Delta x)^2\Big) \\
+ \dots \\
+ \Big(a^2 + 2\cdot(n-1)a\Delta x + (n-1)^2(\Delta x)^2\Big)\Big) \\
= \frac{1}{3}\left(b^3 - a^3\right) \quad (2)
\end{aligned}
$$

LaTeX 常用命令

命令

\chapter 章	\section 节	\subsection 小节	\paragraph 带题头段落
\centering 居中对齐	\emph 强调	\verb 原样输出	\url 超链接
\footnote 脚注	\item 列表条目	\caption 标题	\includegraphics 插入图片
\label 标号	\cite 引用参考文献	\ref 引用图表公式等	

环境

table 表格	figure 图片	equation 公式
itemize 无编号列表	enumerate 编号列表	description 描述

LaTeX 环境命令举例

```
\begin{itemize}
  \item A \item B
  \item C
  \begin{itemize}
    \item C-1
  \end{itemize}
\end{itemize}
```

- A
- B
- C
  - C-1

LaTeX 环境命令举例

```
\begin{itemize}
  \item A \item B
  \item C
  \begin{itemize}
    \item C-1
  \end{itemize}
\end{itemize}
```

- A
- B
- C
  - C-1

```
\begin{enumerate}
  \item 巨佬 \item 大佬
  \item 萌新
  \begin{itemize}
    \item[n+e] 瑟瑟发抖
  \end{itemize}
\end{enumerate}
```

1. 巨佬
2. 大佬
3. 萌新

   n+e 瑟瑟发抖

LaTeX 数学公式

```
$V = \frac{4}{3}\pi r^3$

\[
  V = \frac{4}{3}\pi r^3
\]

\begin{equation}
  \label{eq:vsphere}
  V = \frac{4}{3}\pi r^3
\end{equation}
```

$V = \frac{4}{3}\pi r^3$

$$V = \frac{4}{3}\pi r^3$$

$$V = \frac{4}{3}\pi r^3 \tag{3}$$

- 更多内容请看 这里

```
\begin{table}[htbp]
  \caption{编号与含义}
  \label{tab:number}
  \centering
  \begin{tabular}{cl}
    \toprule
    编号 & 含义 \\
    \midrule
    1 & 4.0 \\
    2 & 3.7 \\
    \bottomrule
  \end{tabular}
\end{table}
公式~(\ref{eq:vsphere}) 的
编号与含义请参见
表~\ref{tab:number}。
```

Table 1: 编号与含义

编号	含义
1	4.0
2	3.7

公式 (3) 的编号与含义请参见表 1。

作图

- 矢量图 eps, ps, pdf
  - METAPOST, pstricks, pgf . . .
  - Xfig, Dia, Visio, Inkscape . . .
  - Matlab / Excel 等保存为 pdf
- 标量图 png, jpg, tiff . . .
  - 提高清晰度，避免发虚
  - 应尽量避免使用

Figure 1: 这个校徽就是矢量图，虽然看起来不像，但确实是矢量图格式

- 一月：完成文献调研
- 二月：研究 THU Beamer Theme 的实现
- 三、四月：修改 NJUPT Beamer 主题
- 五月：论文撰写

[1] MANÈS V J, HAN H, HAN C, et al. The art, science, and engineering of fuzzing: A survey[J]. IEEE Transactions on Software Engineering, 2019, 47(11): 2312–2331.

[2] UNKNOWN. THU Beamer Theme[C/OL] // None. 2015: 1–10. http://far.tooold.cn/post/latex/beamertsinghua.

Thanks!