# Traffic analysis on software packages

Semester Project

Alexandre Dumur
Communication Systems, MA3

# Goals and Motivation

## Goals

- Guessing which software installation a victim is doing
- Track the state of a Machine by keeping a record of installed packages

## Motivation

- Create a profile of the victim
- Can lead to vulnerabilities that an attacker can exploit

# Why Privacy matters?

Profiling the victim can lead to several unpleasant outcome.

- Employer monitoring which software is installed on workers' machines.

## CVE-2017-1000379

| | |
|---|---|
| **Name** | CVE-2017-1000379 |
| **Description** | The Linux Kernel running on AMD64 systems will sometimes map the contents of PIE executable, the heap or ld.so to where the stack is mapped allowing attackers to more easily manipulate the stack. Linux Kernel version 4.11.5 is affected. |
| **Source** | CVE (at NVD; CERT, LWN, oss-sec, fulldisc, bugtraq, EDB, Metasploit, Red Hat, Ubuntu, Gentoo, SUSE bugzilla/CVE, Mageia, GitHub code/issues, web search, more) |
| **NVD severity** | high (attack range: local) |

https://security-tracker.debian.org/tracker/CVE-2017-1000379

# Model

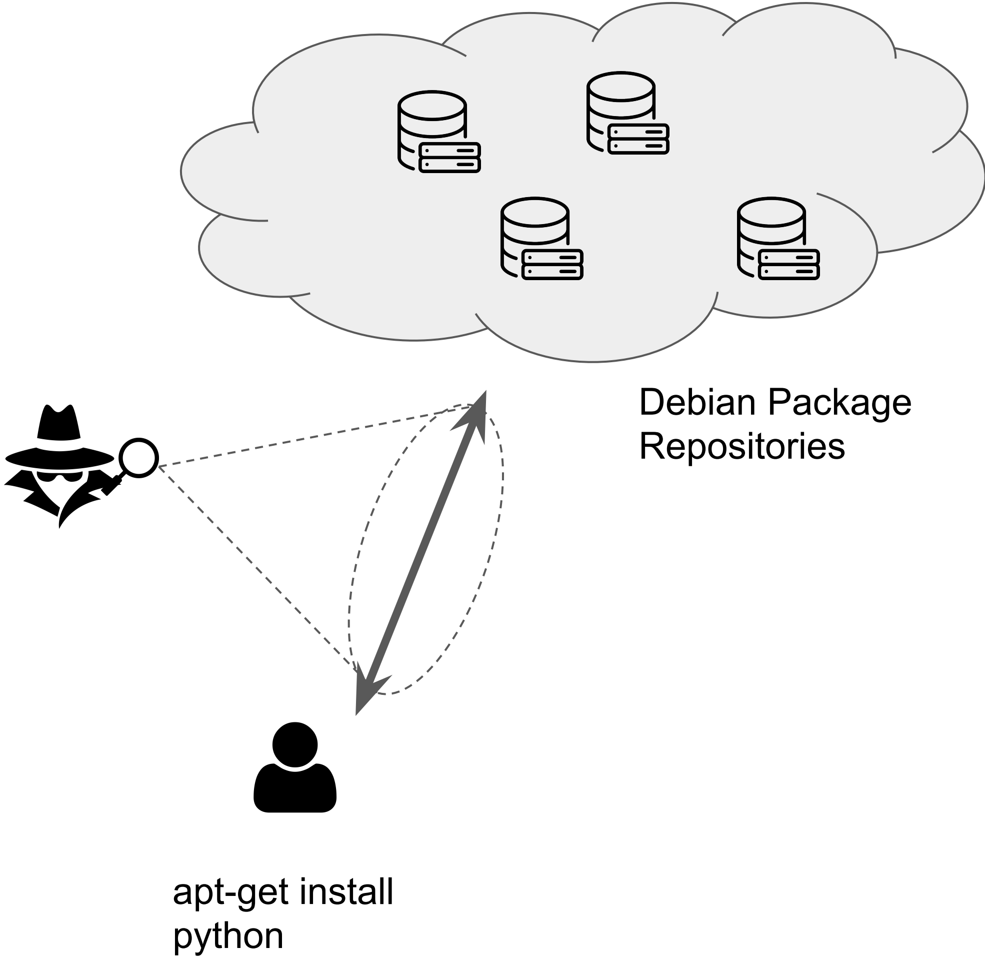
- The victim uses **APT** to install packages.

**Threat Model:**

- The attacker is **passive**
- The attacker **sees** the **traffic** (**encrypted** or **not**)

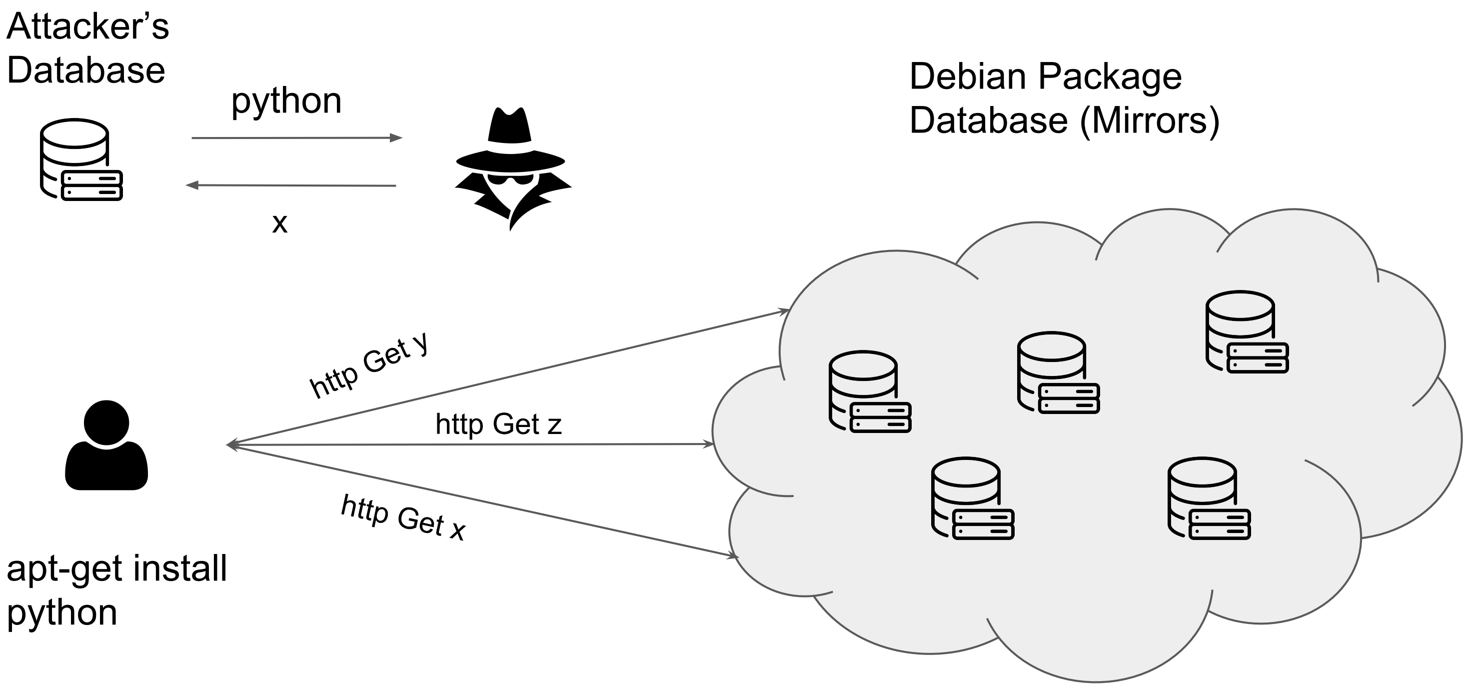
**Assumption:**

- The attacker **can filter** the packets corresponding to the **victim's installation**

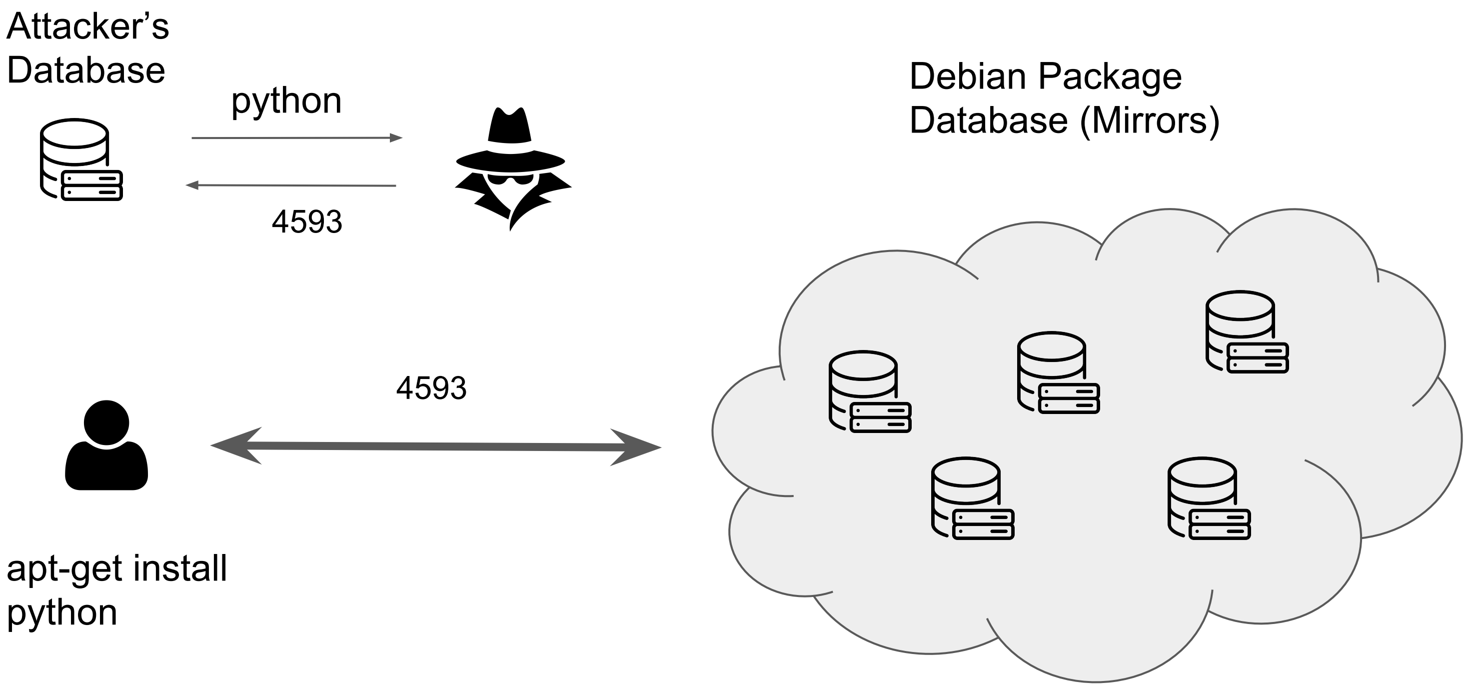


# Attack on HTTP requests - Overview

y and z are python's dependencies packages that APT needs to resolve



# Attack on HTTPs / Downloaded Size - Overview



# Necessary condition to succeed in the attacks - Theoretical Results

**Attack on HTTP:**

- ✓ Field Filename is unique: can be used to determine HTTP get request

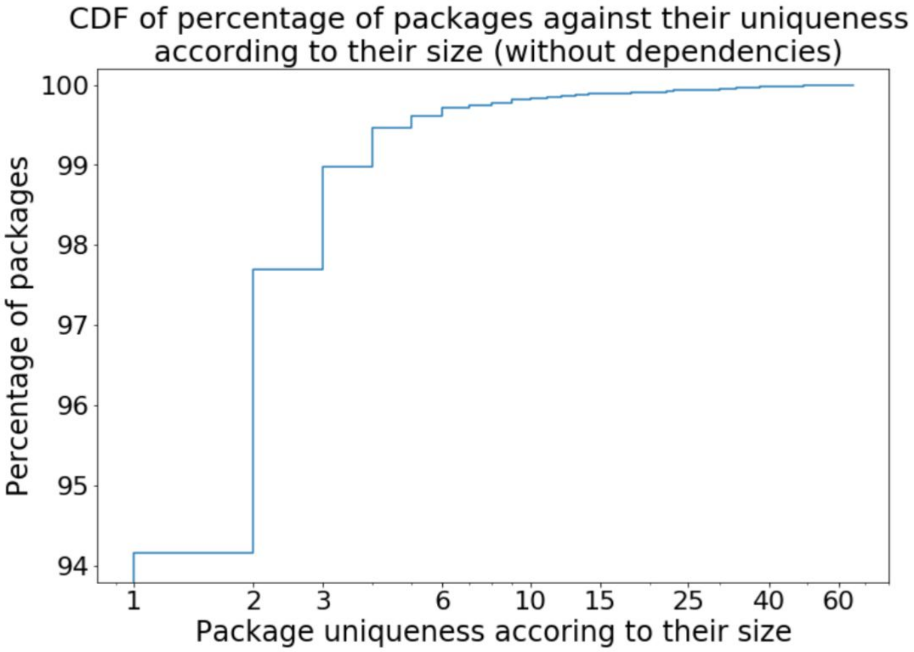
**Attack on Size:**

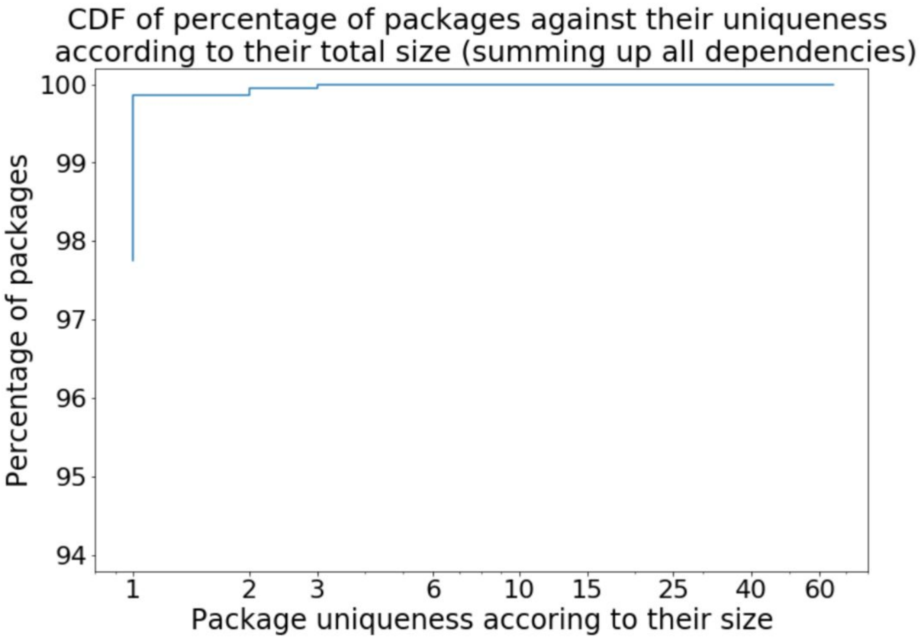
Packages' size are unique:

- ✓ When not considering dependencies of packages in the size
- ✓ When considering all the dependencies of packages in the size
- ✓ When considering a state in between the above

So in all those cases, theoretically, it works.

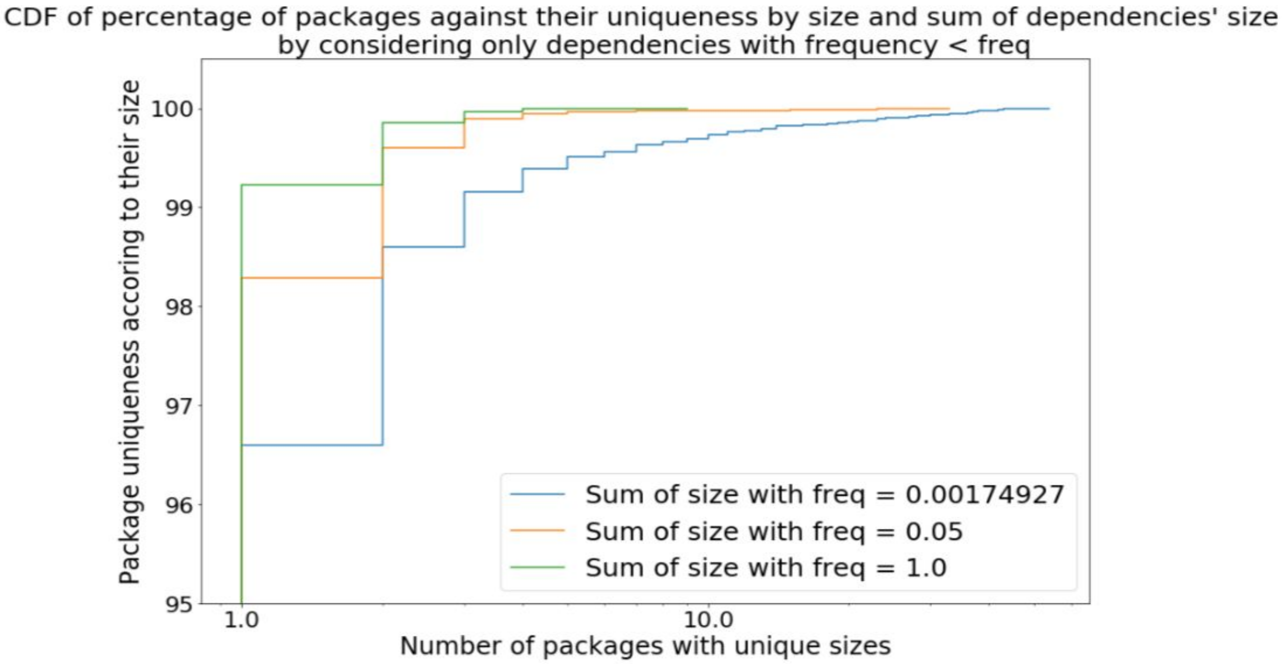
# Size uniqueness when no/all dependencies are considered.



# Is the size still unique for a victim state in between?

**Metric**: frequency of a package listed as dependency. The higher the frequency the more likely it will be installed.



# The methodology we used in this work to do the attacker

1. **Crawl** - Build the attacker's database. (216 LOC)
2. **Capture** - Record packet traces. (485 LOC)
3. **Attack** - Matching capture traces to packages. (389 LOC)
   3.1. **HTTP Matching** - match HTTP GET fields.
   3.2. **Size Matching** - match size of packet traces to the actual downloadable size.

(1090 LOC)

# Practical Results - HTTP request Attack

5*100 captures for packages with different total number of dependencies (0, 1 ,6, 10, 15)

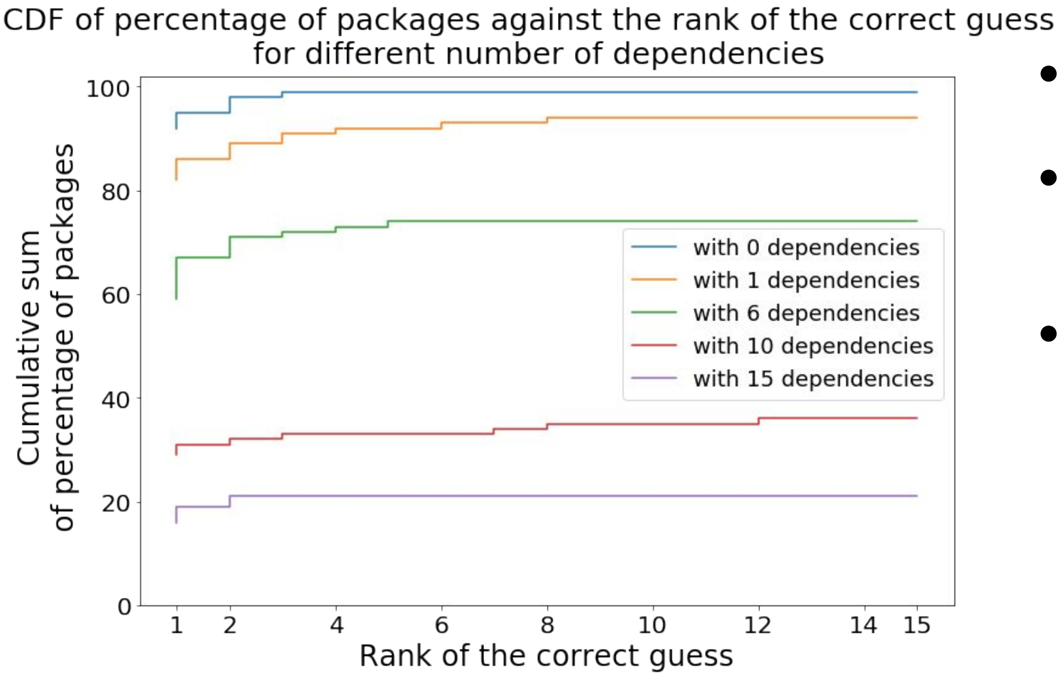
## Attack on HTTP GET

**Results**

| #Dependencies | 0 | 1 | 6 | 10 | 15 |
|---|---|---|---|---|---|
| Succeeded | 98 | 94 | 94 | 98 | 95 |
| Failed | 2 | 6 | 6 | 2 | 5 |

# Practical results Attack on Downloaded Size

## Attack on downloaded size results




## analysis

- Blue line fits well the theoretical results
- Accuracy is lower for packages with bigger number of dependencies.
- Curves are rather flat: the guess is either totally false or the correct guess is the first one.

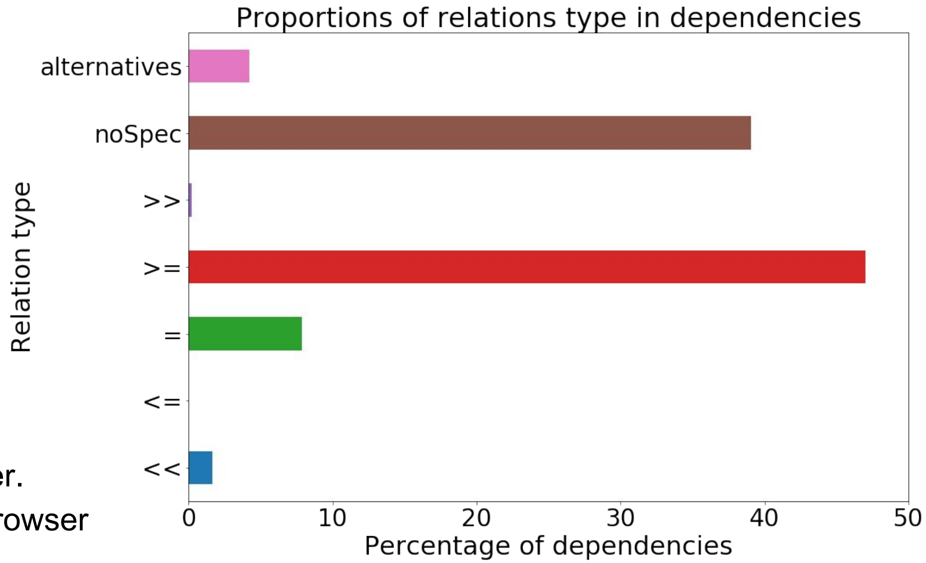
# Theoretical vs Practical results - Explanation

- It is difficult for the attacker to resolve the same packages than APT
  - Dependencies field : Version or Package not exact match: (>>, >=, =, <=, <<, | )
  - Virtual packages

## Example:

```
Package:  mutt
Version:  1.3.17-1
Depends:  libc6 (>= 2.2.1),
          www-browser,
          apache2 | nginx (= 2.1)
```

`www-browser` is a Virtual Package providing a web browser.
`konqueror`, `firefox-esr` are packages providing web browser



# On Targeting linux kernel update

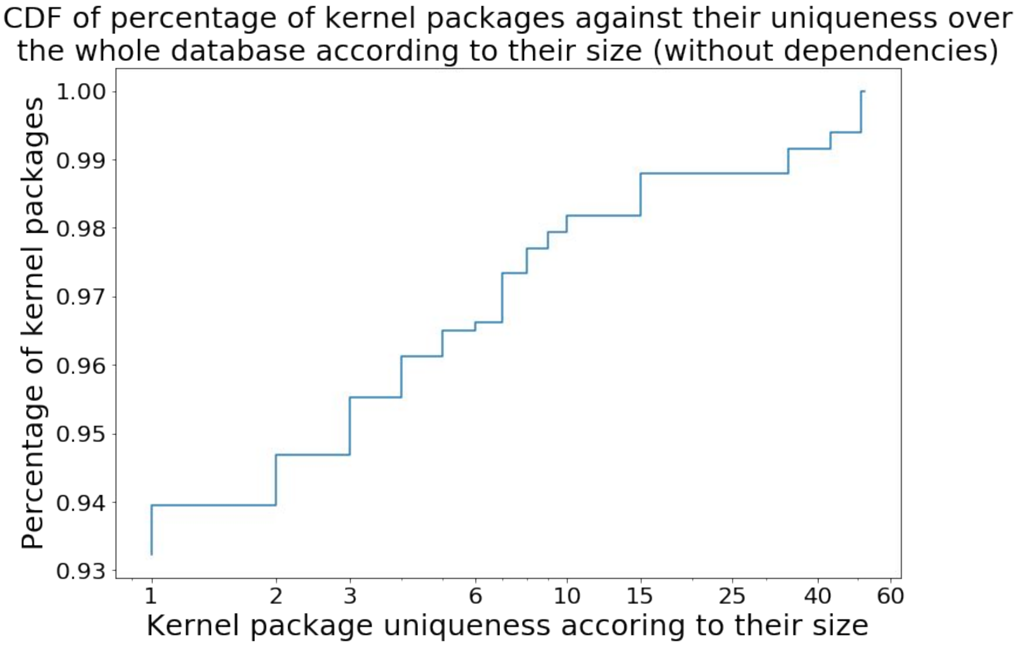
Kernel version is a sensitive information. Any vulnerabilities in such packages can compromise the entire system.

✓ Kernel packages are unique in size (with and without dependencies)

✓ A lot of dependencies are shared between 2 kernels packages.
  - Because from one kernel to another, the kernel requires more or less the same dependencies.
  - In practice, it results in a better attack as we saw in the previous slides.

# Uniqueness of kernel packages

**When no dependencies are considered**




**When all dependencies are considered**

Cdf of the percentage of packages according to their uniqueness when the dependencies are considered

| Uniqueness | 1 | 2 | 3 |
|---|---|---|---|
| Percentage | 99.1% | 99.6% | 100% |

# Conclusion

- **Attack on HTTP** is **easy** and **accurate** but assumes **no encryption**.
    - **Default settings** does **not** use HTTPS.
    - **Most Debian Mirrors** do **not** support HTTPS.

- **Attack on Size** is more **complex**, **less accurate** but **can** assume **encryption**
    - Uncertainty grows with the number of dependences (Difficulty for the attack to resolve same dependencies as APT)
    - Can work well to target kernel update.

## Improvement

- **Attack on Size:**
    - Acquire more knowledge about how APT resolves virtual packages and dependencies
    - Considering multiple flows (Could isolate size of individual package or group of packages).

# Conclusion II

**Outcomes**

- Theoretical results on the feasibility of the attack.
- Ready-to-use code that simulates an attacker and a victim performing updates.

**Future work**

- Analyze how the attack behave on Tor and when the victim uses a VPN.

# References

## Papers


- Tao Wang, Ian Goldberg, 2016. *On Realistically Attacking Tor with Website Fingerprinting*. Proceedings on Privacy Enhancing Technologies, pages 21 - 36.
- Justin Cappos. Justin Samuel Scott Baker John H. Hartman, 2008. *A Look In the Mirror: Attacks on Package Managers*, [online] Available at: `https://isis.poly.edu/~jcappos/papers/cappos_mirror_ccs_08.pdf` [Accessed 29 December 2018]


## APT Docs


- Debian Policy Manual v4.3.0.1, *7.1. Syntax of relationship fields*. [online] Available at: `https://www.debian.org/doc/debian-policy/ch-relationships.html` [Accessed 2 January 2019]
- Steve Ovens, 2018. *The evolution of package managers*. [online] Available at: `https://opensource.com/article/18/7/evolution-package-managers` [Accessed 20 December 2018]


## Images


- https://www.kisspng.com/png-logo-computer-icons-clip-art-white-hat-hacker-icon-5595839/
- https://www.kisspng.com/png-computer-icons-data-model-database-server-server-i-4261325/download-png.html
- https://security-tracker.debian.org/tracker/CVE-2017-1000379

# Annex - APT

## Metadata

- **metadata** of the APT packages are **available locally** in **/var/lib/apt/lists/**
- Are **kept up-to-date** by running **apt-get update**
- Metadata contains the following fields:
    - **Package Name** - Name of the package
    - **Package Version** - Version of the package
    - **Size** - Size of the package in compressed form.
    - **Filename** - Path of the package archive relative to the base directory of the repository
    - **Depends, Recommends** - required and recommended dependencies (APT installs by default all the ***depends*** and ***recommends*** packages that are unresolved).

**Problem:** *Depends* and *Recommends* fields only contains first-level dependencies
→ Extra work for the attacker to have the full list of dependencies

# Annex - Algo Resolving dependencies

**Algorithm 1** Find all dependencies of a package given its first level dependencies

**Input:** $s_{in}$, set of first level dependencies of a package

**Output:** $s_{out}$, set of all dependencies of the underlying package

**function** $\textsc{ResolveDependencies}(s_{in})$
  $s_{out} \leftarrow s_{in}$
  **for** $d$ in $s_{in}$ **do**
    $s_d \leftarrow \textsc{getFirstLevelDependencies}(d)$
    **if** $\textsc{Len}(s_d) == 0$ **then return** $s_{out}$
    **end if**
    $s_r \leftarrow \textsc{ResolveDependencies}(s_d)$
    $s_{out} \leftarrow s_{out} \cup s_r$
  **end for**
  **return** $s_{out}$
**end function**

# Annex - distance formula used for guess ranking

$$\mathrm{dist}(j) = |s_{captured} - \delta(|\mathscr{D}_{\mathrm{j}} \setminus \mathscr{M}| + 1) - s_j - \sum_{i \in \mathscr{D}_{\mathrm{j}} \setminus \mathscr{M}} s_i|$$

Where:

- $s_{captured}$ is the captured size of packet traces
- $\mathscr{D}_{\mathrm{j}}$ is the set of dependencies of package j
- $s_i$ is the size of package $i$
- $\delta$ is the average size of a http header.
- $\mathscr{M}$ is the set of marked packages (packages that are already installed by the victim, i.e. state of the victim)
- $\mathscr{D}_{\mathrm{j}} \setminus \mathscr{M}$ is the set of dependencies of package $j$ without the marked packages.

# Annex - Possibility to know when the attacker is wrong

When the attacker is by far wrong about his guess, the distance function of his first guess package is big.

| | correct guesses ranked <= 15 | correct guesses ranked > 15 |
|---|---|---|
| mean | 27.2 | 656.7 |
| std | 267.4 | 2104.5 |
| min | 0 | 0 |
| 25% | 1 | 8 |
| 50% | 2 | 45 |
| 75% | 3 | 308 |
| max | 3870 | 20548 |

**Table 7.2:** Analysis on the distance of the first guessed package for correct guess ranked < 15 and correct guess ranked > 15

# Annex - CDF of Number of dependencies per packages)