Карта средств защиты ядра Linux

Александр Попов

Positive Technologies

10 июня 2019

Кто я

- Александр Попов
- Разработчик ядра Linux с 2012 года
- Исследователь информационной безопасности в

- Докладчик на конференциях: Linux Security Summit, SHA2017, Open Source Summit, LinuxCon, OSDay, Positive Hack Days, Linux Piter и др.

План доклада

1. Терминология
2. Карта средств защиты ядра Linux
3. Методика работы с данной картой

Определимся с терминами (1)

Операционная система (ОС) - это программное обеспечение, управляющее аппаратными и программными ресурсами компьютера и предоставляющее различные сервисы для компьютерных программ

Определимся с терминами (2)

Ядро ОС - основная (и самая интересная) часть ОС, управляющая выполнением процессов и их доступом к ресурсам вычислительной системы:

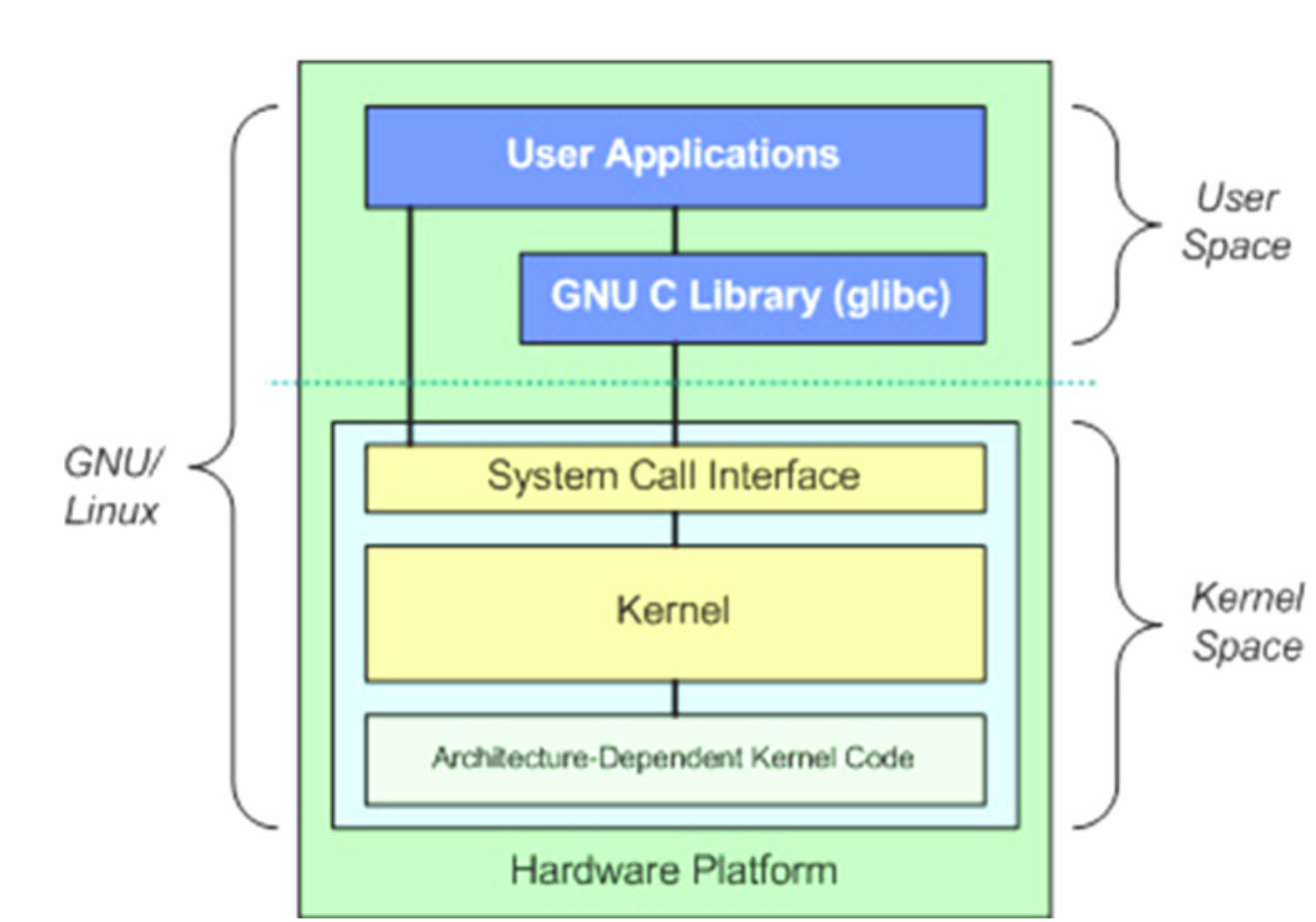
- процессорному времени,
- оперативной памяти,
- устройствам ввода-вывода,
- механизмам межпроцессного взаимодействия,
- средствам сетевого взаимодействия и др.

Linux Kernel

- **Ядро Linux** - основа семейства операционных систем GNU/Linux и ОС Android.

- "One of the most successful collaborative development projects in history"
  https://www.linuxfoundation.org/2017-linux-kernel-report-landing-page/
  - 90% of the public cloud workload,
  - 62% of the embedded market share,
  - 99% percent of the supercomputer market share,
  - 82% percent of the world's smartphones,
  - 15,600 individual developers from over 1,400 different companies since 2005
  - almost 25 million lines of code (v4.13)

Интерфейсы ядра Linux



https://www.ibm.com/developerworks/linux/library/l-linux-kernel/

Определимся с терминами (3)

- Основная задача информационной безопасности — целесообразная и сбалансированная защита конфиденциальности, целостности и доступности данных
- Security is risk management (Bruce Schneier)
  https://www.schneier.com/essays/archives/2007/01/information_security_1.html
- Для оценки рисков необходима модель угроз информационной системы
- Хороший пример: Базовая модель угроз безопасности персональных данных при их обработке в информационных системах персональных данных (ФСТЭК)
  https://fstec.ru/component/attachments/download/289

Уязвимости

- Одна из классификаций угроз информационной безопасности – по используемой уязвимости
- **Уязвимость** - недостаток в программно-аппаратном обеспечении информационной системы, который может быть использован («проэксплуатирован») для реализации угрозы безопасности (для атаки)
- **Эксплоит** - программа или последовательность команд, использующая уязвимости в ПО и применяемая для атаки на информационную систему

Описание уязвимостей

Для систематизации описания множества уязвимостей используются:

- единая база данных уязвимостей CVE (Common Vulnerabilities and Exposures) https://cve.mitre.org/
- CVSS-вектора (Common Vulnerability Scoring System) https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

Метрики CVSS v3

CVSS v3 Vector

NA

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N) | Adjacent Network (AV:A) | Local (AV:L) | Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L) | High (AC:H)

Privileges Required (PR)*

None (PR:N) | Low (PR:L) | High (PR:H)

User Interaction (UI)*

None (UI:N) | Required (UI:R)

Scope (S)*

Unchanged (S:U) | Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N) | Low (C:L) | High (C:H)

Integrity Impact (I)*

None (I:N) | Low (I:L) | High (I:H)

Availability Impact (A)*

None (A:N) | Low (A:L) | High (A:H)

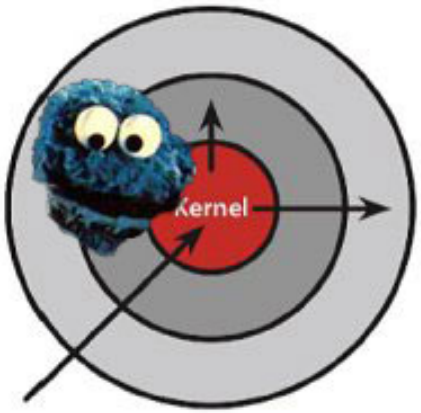
* - All base metrics are required to generate a base score.

Устранение уязвимостей в ядре Linux

- Гигантская кодовая база ядра Linux развивается с огромной скоростью:
  - между релизами v4.8 и v4.13 в ядро добавлялось 7500 строк кода - ЕЖЕДНЕВНО
  - средняя скорость merge при этом была 8.5 патчей/час
  - в каждом релизе участвовало $>$ 1500 разработчиков
- У нас есть санитайзеры, фаззер syzkaller, инструменты статического анализа, НО...
- Уязвимости вносятся быстрее, чем исправляются – см. "Сказку о тысяче ядерных багов" Дмитрия Вьюкова

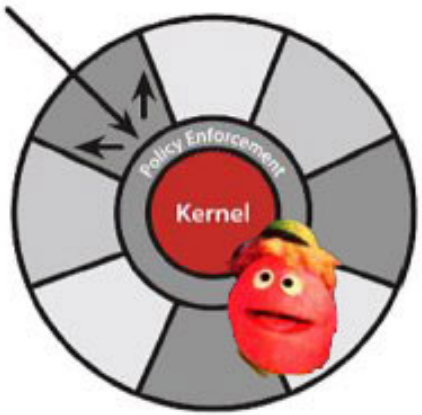
Уязвимости и механизмы разграничения доступа

- Механизмы разграничения доступа в ядре Linux реализуются с помощью LSM (Linux Security Modules)
- Примеры LSM из ванильного ядра: APPARMOR, SELINUX, SMACK, TOMOYO, YAMA
- LSM не защищает от эксплуатации ядерных уязвимостей
- Цитирую grsecurity:



Discretionary Access Control

Once a security exploit gains access to priveleged system component, the entire system is compromised.



Mandatory Access Control

Kernel policy defines application rights, firewalling applications from compromising the entire system.

Blackhats with kernel exploits

Basement dwelling 12-year olds armed with kernel exploit released past Tuesday. A SELinux disabling payload in the exploit turns your entire MAC policy into laughing stock. You spend the rest of the weekend removing SSH backdoors.

Red Hat and Security-Enhanced Linux (SELinux): It's really about the neat diagrams.

Kernel Self Protection Project

- Чтобы повысить безопасность ядра нужно больше, чем исправление ошибок
- Ядро Linux должно безопасно отрабатывать в ошибочной ситуации
- Идеи **grsecurity & PaX** - во многом источник вдохновения
- **Цель**: устранение классов уязвимостей и методов их эксплуатации
- Ссылки:
  - KSPP wiki: http://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project
  - Обзор KSPP (Kees Cook): https://outflux.net/slides/2018/lca/kspp.pdf

Безопасность ядра Linux

Безопасность ядра Linux – очень сложная предметная область. Ключевые понятия:

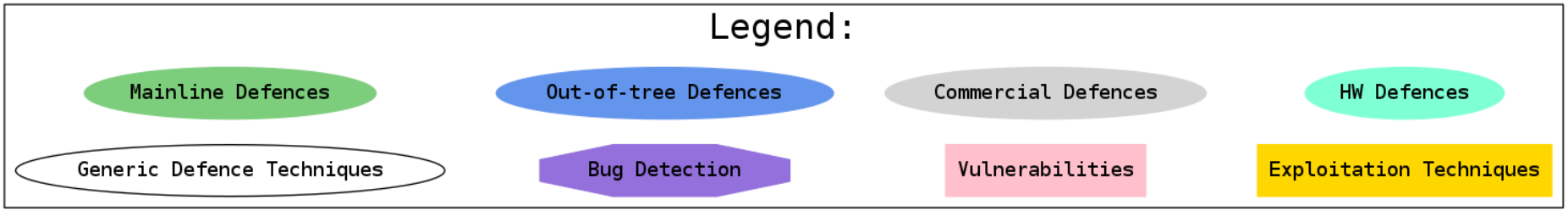
- Классы уязвимостей
- Техники их эксплуатации
- Механизмы выявления ошибок
- Технологии защиты
  - Входящие в mainline
  - Поставляемые отдельно (в т.ч. коммерческие)
  - Требующие аппаратной поддержки

Drawn by Daniel Reeve, made by weta

Все они имеют сложные взаимосвязи...
Было бы полезным иметь графическое представление!

Linux Kernel Defence Map

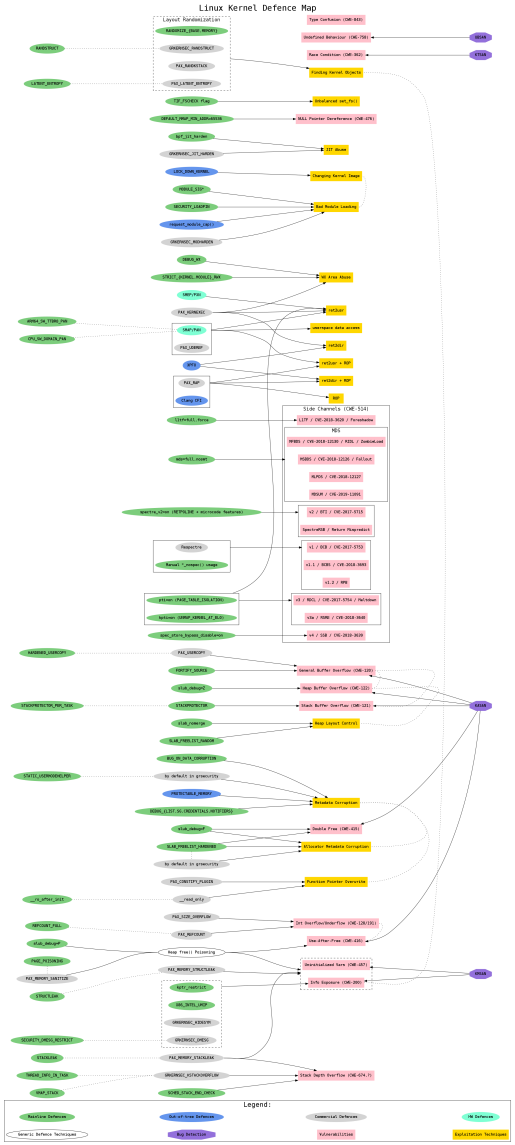
- Поэтому я разработал **Карту средств защиты ядра Linux**
  https://github.com/a13xp0p0v/linux-kernel-defence-map
- Ключевые понятия



- Каждая линия, соединяющая объекты на карте, обозначает их взаимное влияние, суть которого следует выяснять в документации
- N.B. Данная карта не не затрагивает способы уменьшения поверхности атаки

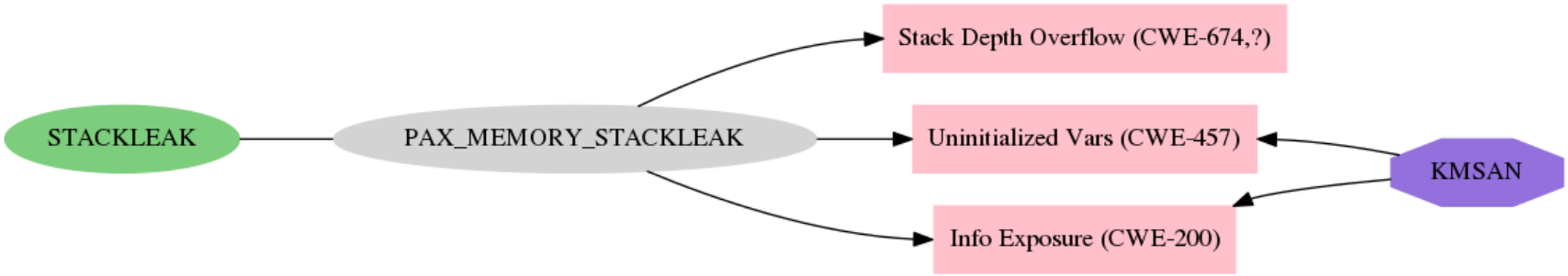
Linux Kernel Defence Map: общий вид

https://github.com/a13xp0p0v/linux-kernel-defence-map



Linux Kernel Defence Map: часть про STACKLEAK

https://github.com/a13xp0p0v/linux-kernel-defence-map



Linux Kernel Defence Map: реализация

- Карту нужно обновлять (как минимум, каждый релиз ядра)
- Хочется иметь исходник в текстовом виде и вести его в VCS
- Не хочется вручную расставлять объекты (с минимальным количеством пересечений связей)
- Поэтому исходник на языке DOT, схему генерирует GraphViz:

```
# dot -Tpng map.dot -o map.png
```

Посмотрим на карту более пристально

Linux Kernel Defence Map: источники

https://github.com/a13xp0p0v/linux-kernel-defence-map

- Свойства безопасности grsecurity
- Документация по безопасности в исходниках ядра
- Рекомендации KSPP по конфигурации ядра
- Текущий статус переноса функций grsecurity в Android Open Source Project
- Обзор тенденций в безопасности ОС от исследовательского центра Microsoft (MSRC)

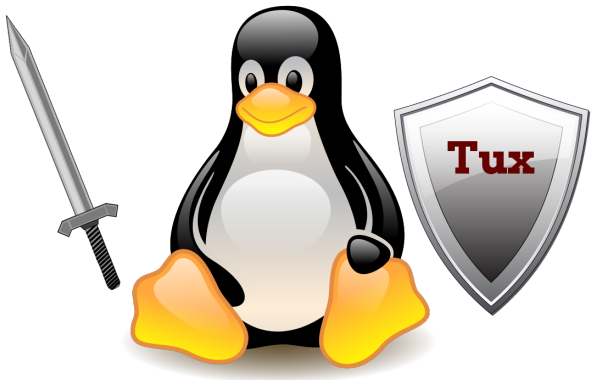
kconfig-hardened-check

Для автоматизации проверки `.config` ядра Linux

я разработал https://github.com/a13xp0p0v/kconfig-hardened-check

В заключение

- Карта средств защиты ядра Linux крайне полезна для:
  - отслеживания актуального состояния безопасности ядра,
  - разработки моделей угроз ИС на базе GNU/Linux,
  - выбора соответствующих средств защиты.
- Приглашаю к участию в ее открытой разработке!



Спасибо! Вопросы?

alex.popov@linux.com
@a13xp0p0v

http://blog.ptsecurity.com/
@ptsecurity

POSITIVE TECHNOLOGIES