
# Continuous Intrusion: Why CI tools are an attacker's best friends

Nikhil Mittal

As a single withered tree, if set aflame, causes a whole forest to burn, so does a single improperly configured Continuous Integration tool destroys a whole enterprise network. – Chanakya (350-275 BCE)

# About me

- Twitter - @nikhil_mitt
- Blog – http://labofapenetrationtester.com
- Github - https://github.com/samratashok/
- Creator of Kautilya and Nishang
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks
  - Clubhack'10, Hackfest'11, Clubhack'11, Black hat Abu Dhabi'11, Black Hat Europe'12, Troopers'12, PHDays'12, Black Hat USA'12, RSA China'12, EuSecWest'12, Troopers'13, Defcon'13, Troopers'14, DeepSec'14, HITB'15, Defcon HHV and PCV'15

# Agenda

- What are Continuous Integration (CI) tools?
- How do CI tools work?
- Abusing popular CI tools
  - Jenkins/Hudson
  - TeamCity
  - Go
  - CruiseControl
- Common Abuse Set
- Defense – Users and CI Tool Developers
- What others are doing
- Conclusion

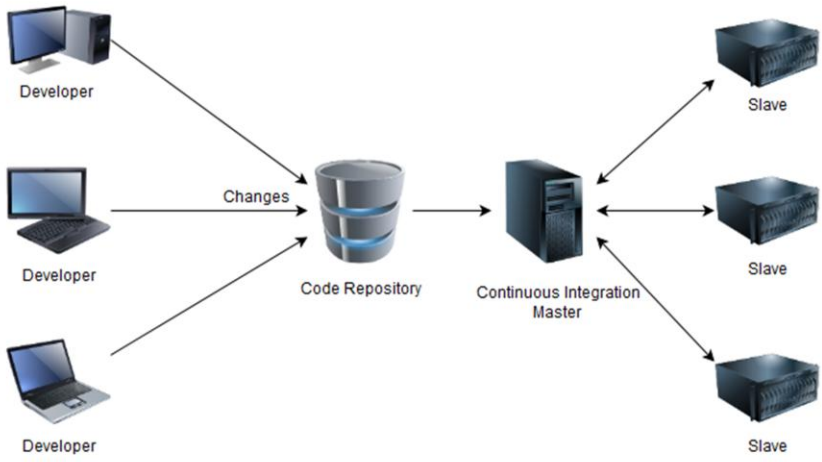
## What are Continuous Integration Tools?

- “Continuous integration describes a set of software engineering practices that speed up the delivery of software by decreasing integration times. Software that accomplish this practice is called continuous integration software.” - Wikipedia



More about Continuous Integration:
http://www.martinfowler.com/articles/continuousIntegration.html

# Basic diagram of Continuous Integration



# Why CI tools are important for Hackers?

- In a typical industry setup, the build jobs are executed on master build server(s) and/or slave machines.
- If a hacker manages to get access to a CI tool, he effectively owns substantial part of the build process, source code (most of the times) and high privilege access to all the machines running slaves/agents.
- This access could be used for lateral movement to get access to more machines.
- I have not been through a single Pen Test or Red Team engagement where access to a CI tool did not result in Domain Admin access. **Yeah, I am 1337 like that!**

## Abusing popular CI tools.

- I have tested three open source tools (Jenkins, CruiseControl and Go) and two propitiatory tool (TeamCity and Hudson*) for this presentation.
- We will mostly have a look at feature abuse and mis-configurations of the above, they may or may not be called vulnerabilities.



* Hudson was not evaluated separately. Most of the things which apply on Jenkins should apply on Hudson as well.

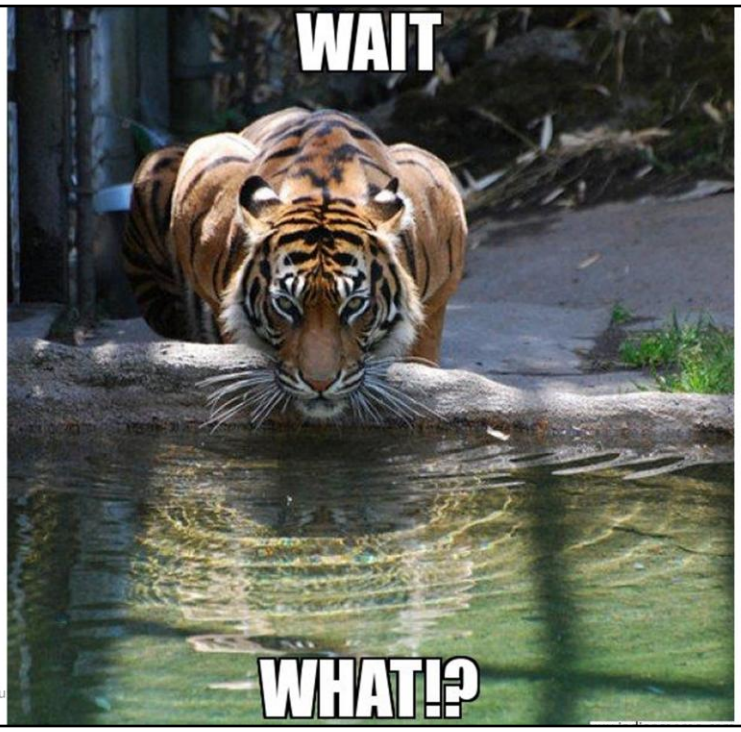
# Jenkins

- Jenkins (http://jenkins-ci.org/) is arguably the most widely used Continuous Integration tool. It is an open source tool which supports distributed/agents build process.
- All the testing has been done on Version 1.633.
- Hudson (http://hudson-ci.org/) was not tested separately since Jenkins and Hudson are very similar in code base and operations.

# Jenkins

Some Security issues

- No authentication in the default installation.
- No protection against brute force attacks.
- No password complexity/policy for user passwords.
- Runs with SYSTEM or high privilege user on Windows (never seen it running with non-admin privileges).
- Prior to version 1.580, all users of Jenkins and console output of builds could be seen without authentication. Still, most Jenkins instances are configured the same way (Read privilege to Anonymous).


WAIT
WHAT!?

# Jenkins

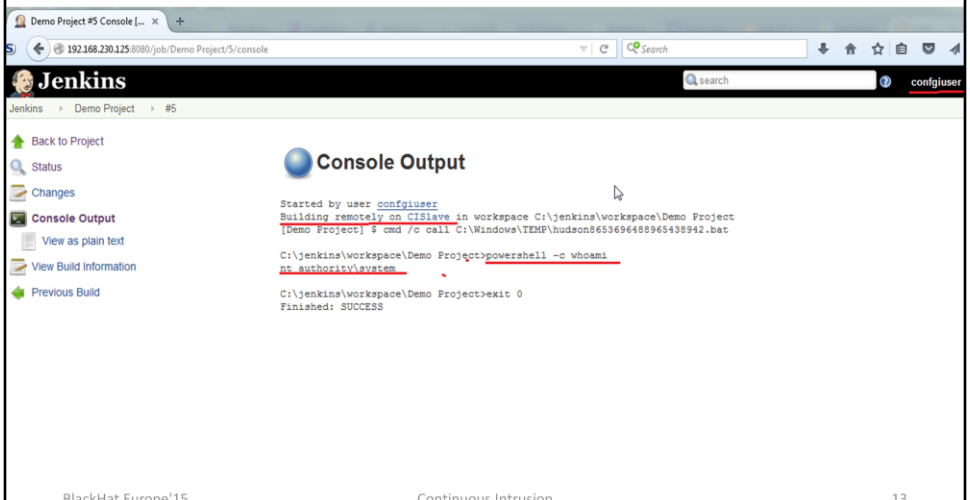
Abusing Build Steps

- If a user has the ability to “Add build step” (non-admin users can do that, if configured so) he can execute commands and scripts on the host Operating System.
- Due to the distributed nature of Jenkins, it is possible to execute commands and scripts on large number of slaves/agents.



http://www.labofapenetrationtester.com/2014/08/script-execution-and-privilege-esc-jenkins.html

# Executing PowerShell/Windows commands using “Add Build Step”






The rights of the user to add or change build configuration are managed using Matrix based security or Project-based Matrix Authorization Strategy.
https://wiki.jenkins-ci.org/display/JENKINS/Matrix-based+security

When running commands on a Windows machine we can leverage PowerShell to execute advanced scripts using this method.

# Jenkins

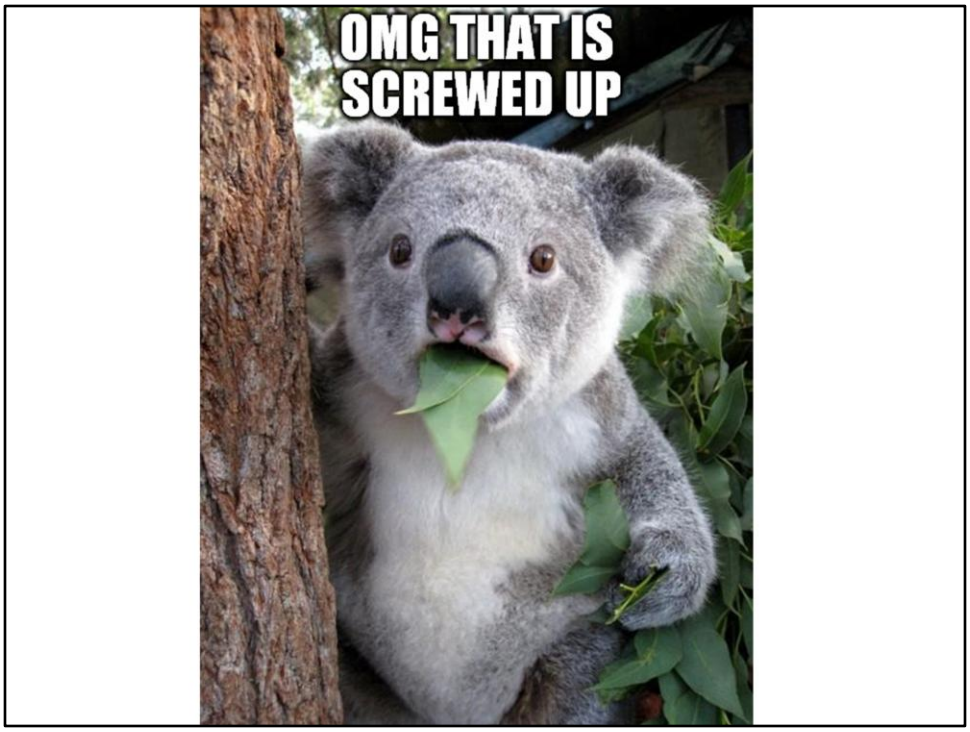
Abusing Build Steps on master

- If a user has the ability to “Add build steps” **on master**, he can effectively remove all security by deleting config.xml from the server and can get admin access on Jenkins.
- This could be achieved only when Jenkins master is on a Windows machine as on *nix machines, the Jenkins service run with non-root privileges.



You need to remove config.xml from $JENKINS_HOME (read the build log for finding out $JENKINS_HOME) to remove all security or replace [useSecurity]true[/useSecurity] with [useSecurity]false[/useSecurity]
The Jenkins service must be restarted after that. During the tests, I was unable to successfully restart the Jenkins service from a build step even on Windows (with SYSTEM privileges). The workaround is to have an interactive reverse shell on the host machine and restart Jenkins service.

Jenkins documentation on Disabling Security
https://wiki.jenkins-ci.org/display/JENKINS/Disable+security




https://imgflip.com/memegenerator/Surprised-Koala

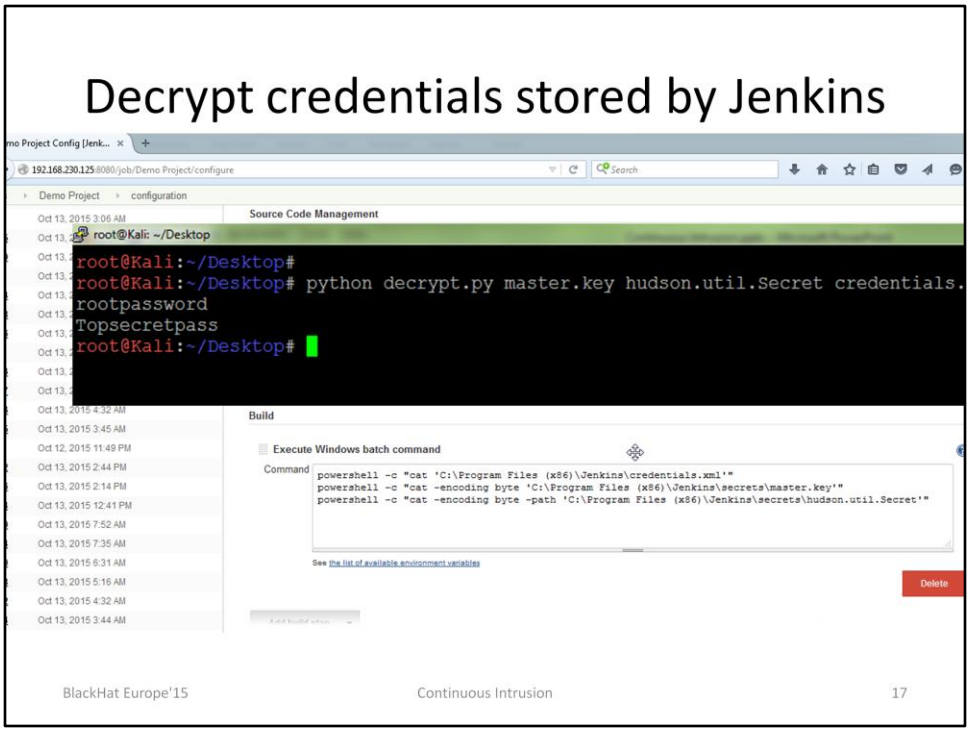
# Jenkins

Stored Credential retrieval in cleartext

- If a user has the ability to “Add build steps” **on master, i**t is possible to retrieve all kind of credentials (passwords, SSH private keys and their passphrases etc) stored in Jenkins in clear text.



Taken from http://thiébaud.fr/jenkins_credentials.html




We need credentials.xml from $JENKINS_HOME and master.key and hudson.util.secret from $JENKINS_HOME/secrets/

We are reading the keys master.key and hudson.util.secret in bytes and will convert them back to file on our own machine. On a Windows machine the conversion could be done by using TextToExe.ps1 from Nishang.
https://github.com/samratashok/nishang/blob/master/Utility/TexttoExe.ps1

# Jenkins

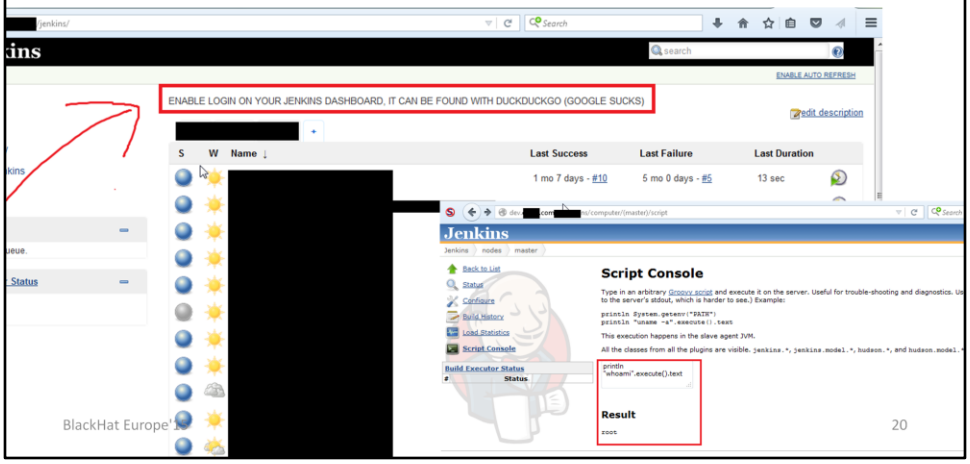
Some bad usage/misconfiguration

- Credentials in clear text found in build parameters and other build configuration.
- Sensitive files shared at [jenkinsurl]/userContent/

# Jenkins

- Google Dorks: intitle:"Dashboard [Jenkins]" for Jenkins instances and intitle:"Dashboard [Jenkins]" intext:"Manage Jenkins“ for instances providing unauthenticated admin access.
- Sensitive data, database credentials, Git credentials, SSH keys and more could be accessed. Ridiculously simple to abuse.

# ENABLE LOGIN ON YOUR JENKINS DASHBOARD, IT CAN BE FOUND WITH DUCKDUCKGO (GOOGLE SUCKS)




EVERYTIME YOU DON'T GIVE A LIVE DEMO
GOD KILLS A KITTEN
memegenerator.net

# TeamCity

- TeamCity is a CI server from JetBRAINS “that is easy to set up and free of charge for small and medium teams.”
- It is widely used and supports distributed builds (agents on different Operating Systems).
- All the demos will be done on TeamCity 9 build 37059.



https://confluence.jetbrains.com/display/TCD9/Getting+Started

# TeamCity

Some Security issues

- Registration of new users is enabled by default. Registered user gets Project Developer privileges.
- No password complexity/policy for user passwords.
- Runs with SYSTEM or high privilege user on Windows (most configs settle with an admin account).



https://confluence.jetbrains.com/pages/viewpage.action?pageId=54334889#HowTo...-TeamCitySecurityNotes


SAME OLD SHIT
WHO CARES?
memegenerator.net

# TeamCity

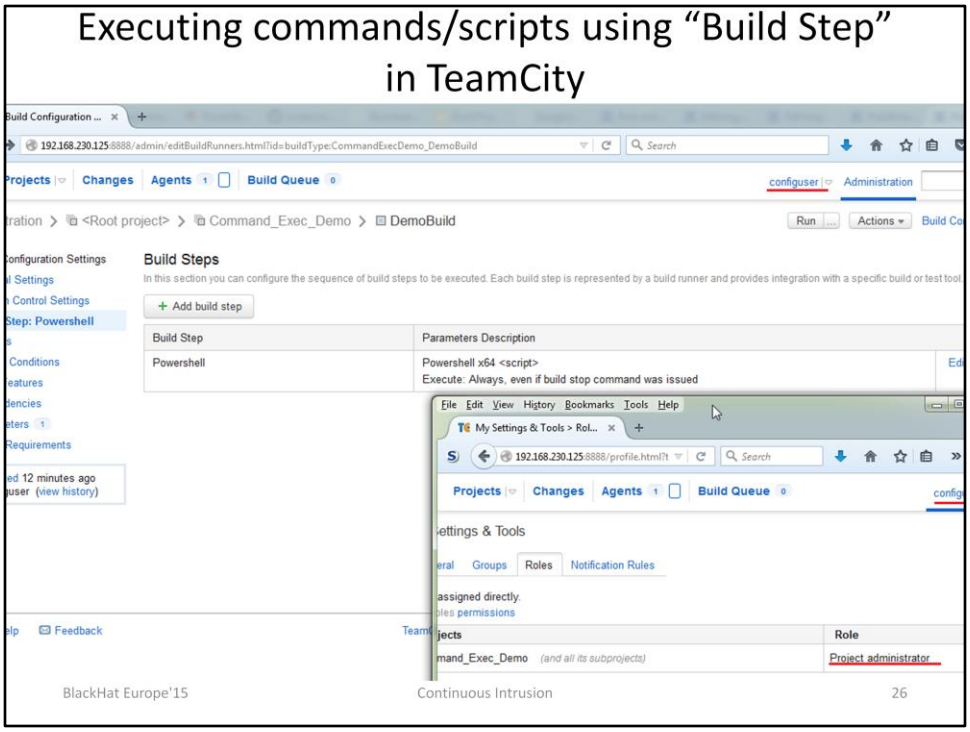
Abusing Build Steps

- If a user has the ability to add build steps (a Project Administrator, unless configured otherwise), he can execute commands and scripts on the host operating system of master and all the agents by adding build steps.
- Due to the distributed nature of TeamCity (like Jenkins), it is possible to execute commands and scripts on large number of slaves/agents.



https://confluence.jetbrains.com/display/TCD9/Role+and+Permission

Teamcity documentation recommends not to have build agent on master but looks like only few care about that.




A Build Step could be added with the Project Administrator or even lower privileges (if configured that way). PowerShell commands and scripts could be executed using the PowerShell runner.
https://confluence.jetbrains.com/display/TCD9/PowerShell

On *nix machines, shell commands and scripts could be executed.

# TeamCity

Abusing agent on master

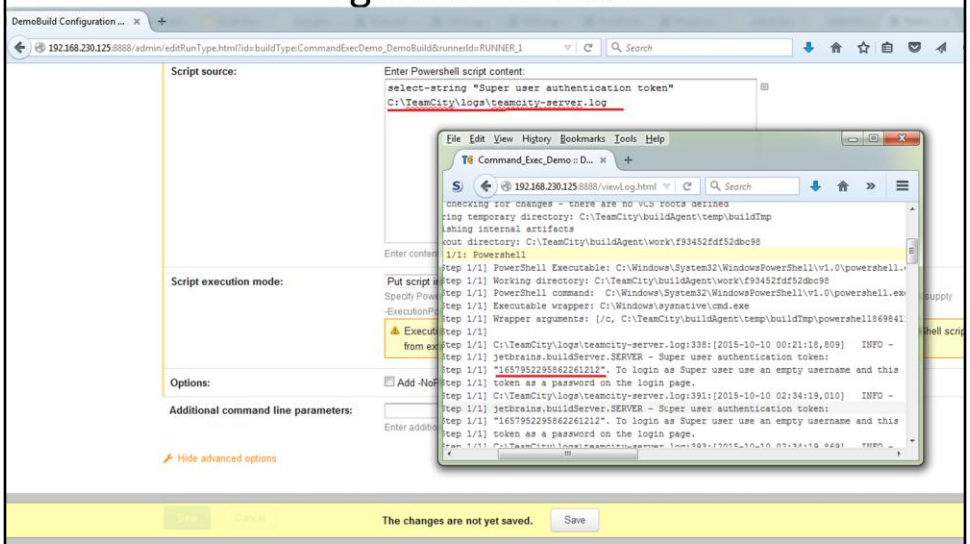
- If a user has the ability to configure builds **on master** (a Project Administrator, unless configured otherwise), he can read the token of the SuperUser from the teamcity-server.log (TeamCity 9) file.
- The token could be used to login as SuperUser.

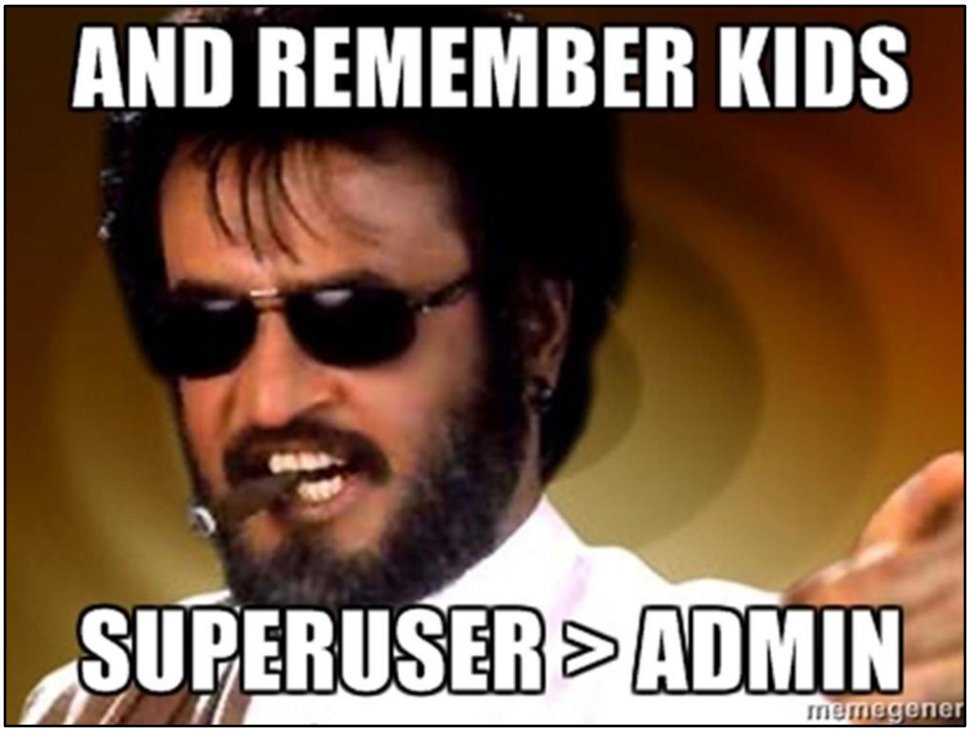


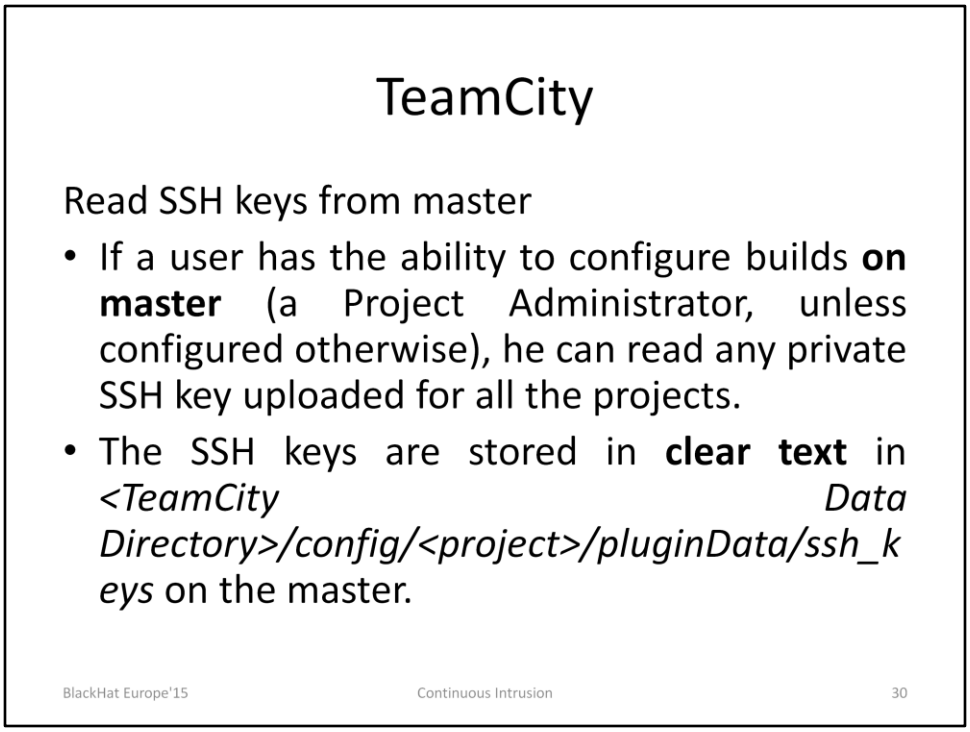
https://confluence.jetbrains.com/display/TCD9/Super+User

Fun Fact: You can lock out SuperUser for one minute by clicking Log in button five times without entering a Username and Password. This makes it easy to block it by repeating login requests indefinitely.

# Accessing Super User token by abusing agent on master




AND REMEMBER KIDS
SUPERUSER > ADMIN
memegener

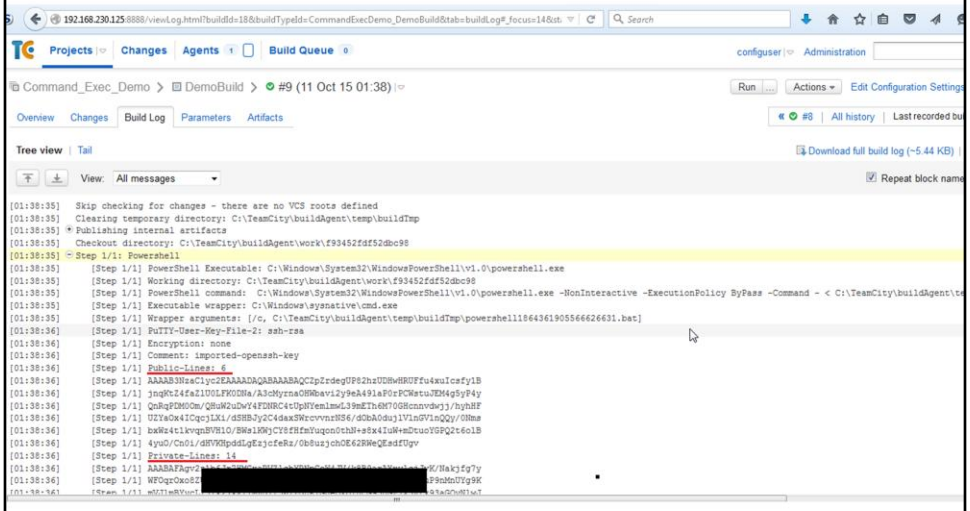



https://confluence.jetbrains.com/display/TCD9/SSH+Keys+Management

Here is how it could be done:

1. To know the data directory of the master, look for “Data Directory” in the teamcity-server.log. Use a PowerShell runner.
2. Use “cat <TeamCity Data Directory>/config/<project>/pluginData/ssh_keys/ *” to list contents of all the keys.

# Accessing SSH Keys by abusing agent on master



# TeamCity

Some interesting points/bad usage

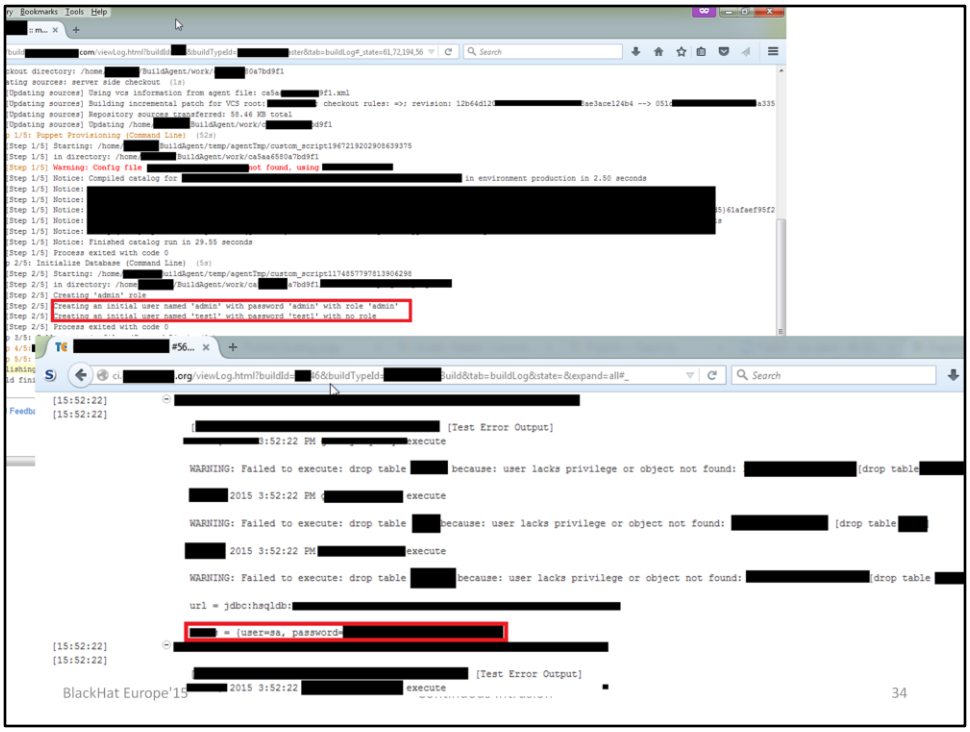
- Guest login could be enabled. In fact, so many public instances has it enabled.
- Users tend to use Passwords in Build Parameters. If the Type of parameter is “Text”, the Password is visible even with a Project Developer privilege.



Teamcity supports using the Password type Parameter for passwords but I have seen so many users using Text Parameter for passwords.
See: https://confluence.jetbrains.com/display/TCD9/Typed+Parameters

# TeamCity

- Google Dork(s) intitle:"Projects - TeamCity“, intitle:Register a new account – TeamCity and more ;)
- The information that can be collected from public instances of TeamCity with Guest privileges is mind boggling! I saw web portal credentials, database credentials, hidden services and code repositories, employee data and much more on some public instances.




Couple of examples of credentials in Build Logs. In both the above screenshots, access to public instances and the Build Logs is with Guest privileges.


DEMO TIME
IS THIS NOT WHY YOU ARE HERE?

# Go

- Go is an open source CI server from ThoughtWorks. They also created CruiseControl.
- All the demos will be done on Go 15.2.0.



http://www.go.cd/
http://www.thoughtworks.com/products/go-continuous-delivery

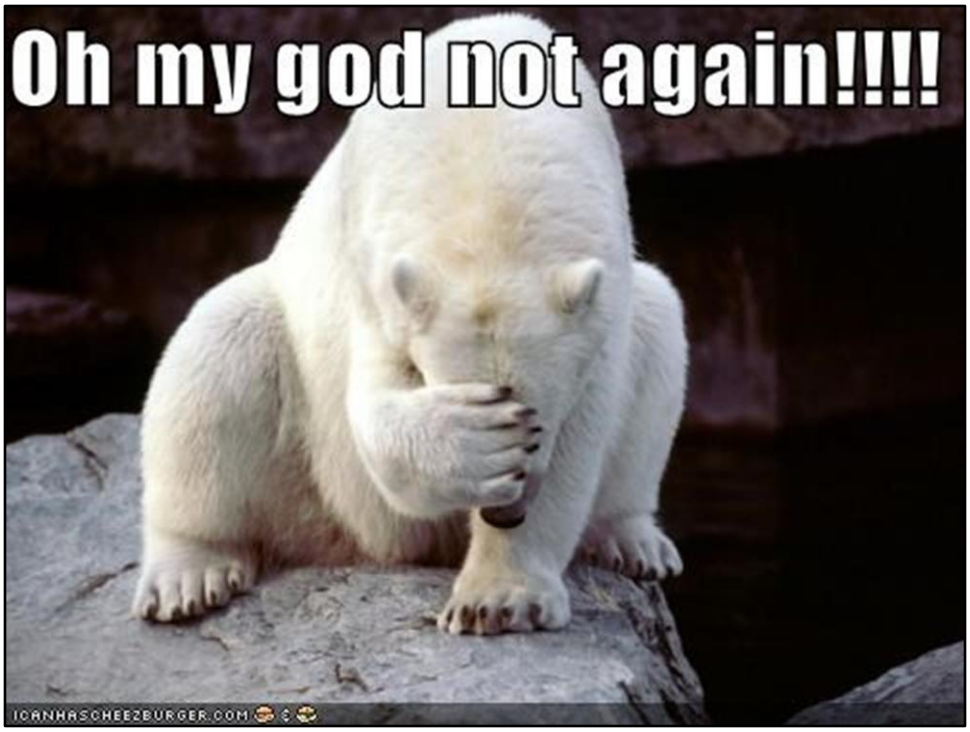
# Go

Some Security issues

- No authentication in the default installation.
- No protection against brute force attacks.
- No password complexity/policy for user passwords.
- Runs with SYSTEM or high privilege user on Windows (most configs settle with an admin account).



http://support.thoughtworks.com/entries/22299328-Go-Security-Questions


Oh my god not again!!!!
ICANHASCHEEZBURGER.COM

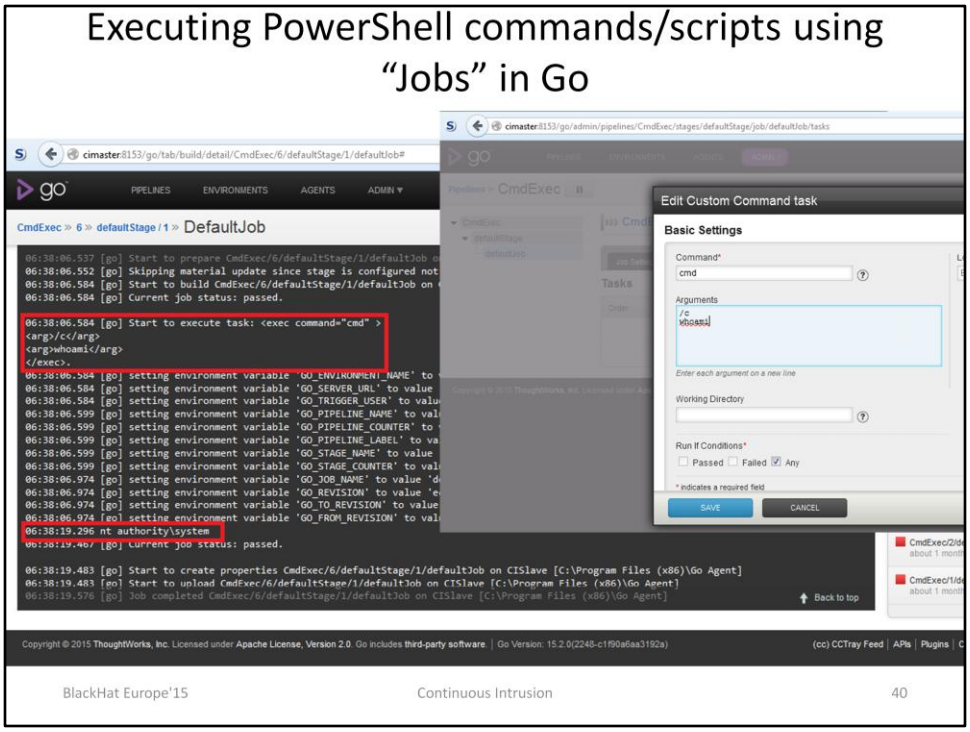
# Go

Abusing Build Steps

- If a user has the ability to add/edit Jobs (Pipeline group administrators can do that), he can execute commands and scripts on the host Operating System.
- Due to the distributed nature of Go, it is possible to execute commands and scripts on large number of slaves/agents.



http://support.thoughtworks.com/entries/22873043-go-s-custom-command
http://www.go.cd/documentation/user/current/advanced_usage/command_repository.html



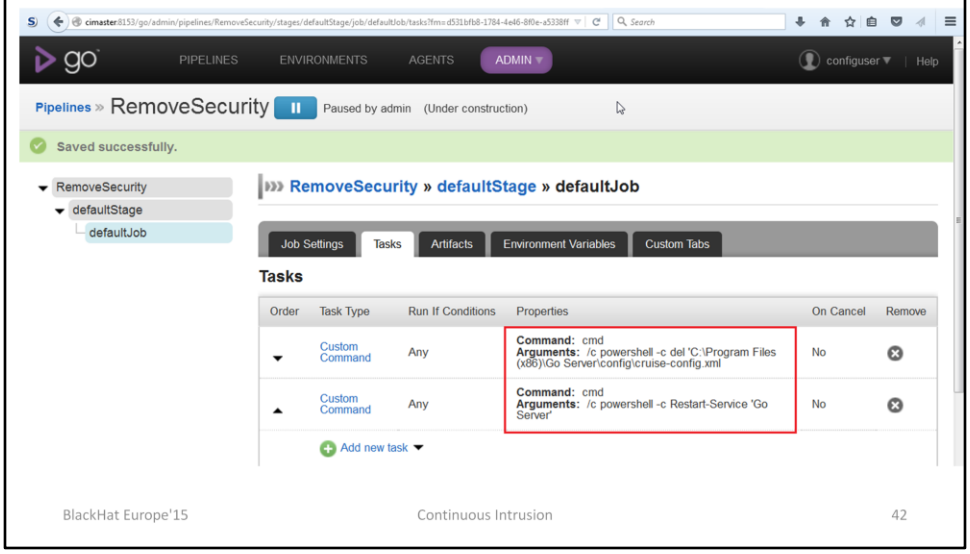

We need Pipeline Group Administrator rights to be able to configure Jobs which can run custom commands.

# Go

Abusing agent on master

- If a user has the ability to add/edit Jobs (Pipeline group administrators can do that) **on master**, he can remove all security and become administrator.
- Cruise-config.xml needs to be removed from the config directory in the Go installation and the Go Server service must be restarted.
- Go has no agent/executor on master by default.

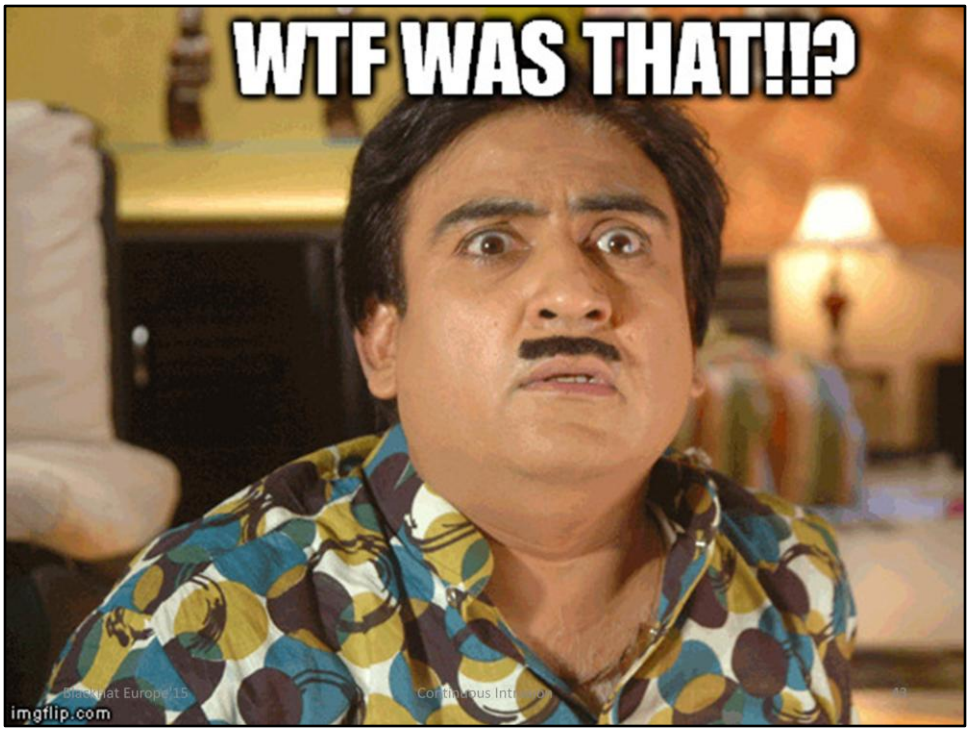
# Removing Security by abusing agent on master




In above, the command cmd /c powershell -c del 'C:\Program Files (x86)\Go Server\config\cruise-config.xml' will remove the configuration file of Go.
The command cmd /c powershell –c Restart-Service 'Go Server' will restart the Go Server service.
After this, all security will be removed from the Go dashboard and anyone who knows the URL will have admin rights.

Instead of removing the cruise-config.xml file, we can also remove only the <security></security> part of it and restart the Go Server service for same effect.

Or we can add the current user to <admins> in the <security> part of cruise-config.xml


WTF WAS THAT!!?
imgflip.com

# Go

Credentials Storage

- The password file option for creating users need the passwords to be Base64 encoded SHA-1 (no salt). These could be successfully computed in plaintext using brute force/rainbow tables.
- SSH keys are stored in cleartext on the disk. A user with ability to configure jobs on master can read the keys.



Documentation on using password files:
http://www.go.cd/documentation/user/current/configuration/dev_authentication.html
https://github.com/gocd/gocd/blob/master/manual-testing/ant_hg/password.properties

Location of SSH keys is:
C:/Program Files (x86)/Go Server/%HOMEDRIVE%%HOMEPATH%/.ssh
/var/go/.ssh on Linux

# Go

Some interesting points/bad usage

- Credentials of the form https://user:password@github.com/me/myrepo are seen in cleartext in Go console to Pipeline group administrators.
- Like other CI tools, build parameters and environment variables are also good places to look at.

# Go

- Google Dork(s) intitle:"Administration - Go" inurl:/go/admin, intitle:"Go - Login" inurl:go/auth/ and more.
- Not many public unauthenticated instances of Go, yet.


ON YOUR FEET MAGGOT
ITS DEMO TIME

# CruiseControl

- CruiseControl is an earlier CI solution from ThoughtWorks.
- It is no more developed actively but it is still used. There are many public instances as well.
- Lets have a look at command exec only.


OLD, BUT NOT OBSOLETE

# CruiseControl

- CruiseControl has no authentication by default.
- The interesting URLs are /dashboard and /cruisecontrol.
- Commands could be executed by adding an “exec” builder in the Schedule category.
- intitle:"CruiseControl - Dashboard“ shows there are still many interesting public instances of it out there.



Documentation for the exec builder:
http://cruisecontrol.sourceforge.net/main/configxml.html#exec

## Demo of a Penetration Test Scenario

- A Jenkins (or any other CI tool) agent running on a Windows machine.
- Execute PowerShell scripts on it to enumerate available tokens.
- Reuse token of a Domain Administrator.
- Too easy? You will be surprised how often it works ;)


GENTLEMEN, WE HAVE THE DOMAN ADMIN PRIVILEGES
memegenerato

# Common Abuse Set

- Minimal security by default. Security controls which are common for other tools are missing.
- Ability to run commands and scripts on OS from non-admin users of the CI tool. In many cases , this provides Administrative privileges on OS across platforms.
- If an agent is running on the master, all security is effectively useless.
- Insecure storage of credentials and SSH keys.
- So many public instances of these tools.

# Defense (Users)

- No build agent/executor should run on the master EVER.
- Restrict privileges which allow configuration of build steps.
- Pool agents together for specific projects.
- Secure the administrators dashboard.
- Expose a CI tool to the internet only if really required.
- Do not provide read privileges to Anonymous users.

# Defense (CI Tools Developers)

- Enforce password policies (complexity, expiry, captcha on login failures etc.)
- A user must install agent/salve explicitly on the master. Do not include it in the install package. Show a warning if an agent/slave is installed on the master.
- Do not store credentials and keys in cleartext on master.
- Assume that your users don’t know a thing about security.

# What others are doing?

- "The Jenkins project has received multiple credible reports indicating that unsecured, publicly accessible instances of Jenkins are being targeted and infected with malware"
  https://wiki.jenkins-ci.org/display/SECURITY/Jenkins+Security+Advisory+2015-10-01
- Facebook used to run an unauthenticated Jenkins instance
  http://blog.dewhurstsecurity.com/2014/12/09/how-i-hacked-facebook.html
- "We got hacked. We were running a (unauthenticated) Jenkins instance in this machine." – plumber.eu
  https://plumbr.eu/blog/plumbr-blog/we-got-hacked

# Previous Work

- Is this your pipe? Hijacking the build pipeline. (Defcon 22): https://goo.gl/9iTHmz
- https://www.pentestgeek.com/penetration-testing/hacking-jenkins-servers-with-no-password/
- http://www.labofapenetrationtester.com/2014/06/hacking-jenkins-servers.html
- http://www.labofapenetrationtester.com/2014/08/script-execution-and-privilege-esc-jenkins.html
- http://zeroknock.blogspot.in/2013/08/protect-your-software-development-web.html

# What was not covered/ Future research

- Master<->agent communication and MITM attacks.
- Compromising integrity of the source code and the build process.
- Web application related vulnerabilities of web consoles.
- Memory corruption bugs.

# Conclusion

- CI tools lack many basic security controls. Still, paying attention to them and configuring them properly will reduce the attack surface.
- Due to the distributed nature of their working, compromise of UI/dashboard even with non-admin privs may result in access to all slave machines.
- Running a poorly configured CI tool is like providing a ready-to-use botnet for anyone to exploit.

As a single withered tree, if set aflame, causes a whole forest to burn, so does a single improperly configured Continuous Integration tool destroys a whole enterprise network. – Chanakya (350-275 BCE)

# Thank you

- Questions?
- Please fill the feedback forms.
- Follow me @nikhil_mitt
- nikhil.uitrgpv@gmail.com
- http://labofapenetrationtester.com/
- https://github.com/samratashok/ContinuousIntrusion