

Forum:[Hack Forums](#)

Thread: [Injecting/ Messing with game](#)

Lexxi. - September 24, 2016, 3:42 am

hi, so there is a game that is coded in java, very easy to get around with modifying with cheat engine etc.

Due to the game being coded in java, would i be able to inject scripts etc into this game?

the game is server-sided mainly, can be played on FB, IOS and android.

Viagr4 - September 24, 2016, 1:34 pm

Sure you can. I can think of two examples of what you could do based on what you said:

Static Analysis:

- Open the game
- View the HTML source, and look for the binary location. If it's Java then the file is probably a .jar file.
- Download the JAR to your computer.
- Decompile it with a tool like [JD-Gui](#) Most developers do not obfuscate their Java code, and thus the source code can be read in plain text
- Save the decompiled source code
- Analyze the Java source code. Sometimes you might even find server's credentials ?
- Modify the source code at your will. Like modding the inner class's variable of your player object to initialize it with 999999999999999999 life.
- Re-compile the modified client. Many tutorials out there on how to re-compile a jar file. After all a .jar is nothing more than a fancy ZIP file
- Run the client

Dynamic Analysis:

- Download Wireshark (or tcpdump if you are in linux) and Burp
- Run Wireshark
- Open your game
- In Wireshark, filter the network traffic and check out what protocol is using to communicate to the server endpoint
- Many FB games use a REST API to communicate the server with the client. Therefore i'm going to assume that [Burp](#) will help you in this case, as the client will be probably using HTTP as protocol.
- Open burp, and proxify the traffic. Depending if you are doing this in linux or Windows you could go with iptables or [proxifier](#).
- Ideally, by now you will be able to see the network traffic and tamper the HTTP requests sent by the game. For example, imagine that in the game you shoot someone and win one point, so the game would tell the server to add you one point and send an HTTP request like:

Quote:POST /api/game/setScore HTTP/1.1

Host: somegame.com

score=1

Or if it's using JSON:

Host: somegame.com

If you tamper the traffic, this will allow you to modify them and do things like:

Host: somegame.com

Optionally, you could do all sort of actions in the game and after a while have a general idea of all that requests that could be send. This applies with other protocols as well.

The two approaches above combined. Having the source code at hand and watching what is being passed through the connection between the game and the game server is the way to go. You would have a much bigger understanding of what's happening.

Also the JAR could have a sanity check for each resource, it's a hash verification check. This is to avoid custom modifications ensuring that it's the standard client running, and applies even for the keystore in some case.

This can vary widely but everything described above is based on most common scenarios.

(Sep 24, 2016 01:34 PM)Viagr4 Wrote: [\[CITE\]](#) Sure you can. I can think of two examples of what you could do based on what you said:

- Open the game
- View the HTML source, and look for the binary location. If it's Java then the file is probably a .jar file.
- Download the JAR to your computer.
- Decompile it with a tool like [JD-Gui](#) Most developers do not obfuscate their Java code, and thus the source code can be read in plain text
- Save the decompiled source code
- Analyze the Java source code. Sometimes you might even find server's credentials ?
- Modify the source code at your will. Like modding the inner class's variable of your player object to initialize it with 999999999999999999 life.

- Re-compile the modified client. Many tutorials out there on how to re-compile a jar file. After all a .jar is nothing more than a fancy ZIP file
- Run the client

Dynamic Analysis:

- Download Wireshark (or tcpdump if you are in linux) and Burp
- Run Wireshark
- Open your game
- In Wireshark, filter the network traffic and check out what protocol is using to communicate to the server endpoint
- Many FB games use a REST API to communicate the server with the client. Therefore i'm going to assume that [Burp](#) will help you in this case, as the client will be probably using HTTP as protocol.
- Open burp, and proxy the traffic. Depending if you are doing this in linux or Windows you could go with iptables or [proxifier](#).
- Ideally, by now you will be able to see the network traffic and tamper the HTTP requests sent by the game. For example, imagine that in the game you shoot someone and win one point, so the game would tell the server to add you one point and send an HTTP request like:

Or if it's using JSON:

If you tamper the traffic, this will allow you to modify them and do things like:

Optionally, you could do all sort of actions in the game and after a while have a general idea of all that requests that could be send. This applies with other protocols as well.

Hybrid Analysis:

The two approachs above combined. Having the source code at hand and watching what is being passed through the connection between the game and the game server is the way to go. You would have a much bigger understanding of what's happening.

It's possible that you won't be able to intercept the traffic at first if the game uses a secure connection. This because Burp uses it's own self-signed certificate to intercept the traffic. For circumventing this barrier you would need to open up the jar's keystore, export the Burp CA and add it there.

Also the JAR could have a sanity check for each resource, it's a hash verification check. This is to avoid custom modifications ensuring that it's the standard client running, and applies even for the keystore in some case.

This barrier can be bypassed in different ways but this depends on how that resource checksum is implemented (if any). Note: the default password of the Java keystore is usually: 'changeit' and no one changes it ?

This can vary widely but everything described above is based on most common scenarios.

Good luck ?

Wow! Awesome guide! Thanks a lot

This Viagr4 guy just knows his shit... im impressed

?Criminal - September 24, 2016, 3:57 pm

Have you messed with cheat engines ASM injection feature? it's pretty powerful if you know what ASM registers to use.

Lexxi. - September 25, 2016, 1:20 am

(Sep 24, 2016 03:57 PM)?Criminal Wrote: [\[CITE\]](#) Have you messed with cheat engines ASM injection feature? it's pretty powerful if you know what ASM registers to use.

no clue, can just do basic x, y adding etc, i cant replace names as serversided images etc, they are updated when changed. no idea for ASM injection, sounds interesting

(Sep 24, 2016 01:34 PM)Viagr4 Wrote: [\[CITE\]](#) Sure you can. I can think of two examples of what you could do based on what you said:

Static Analysis:

- Open the game
- View the HTML source, and look for the binary location. If it's Java then the file is probably a .jar file.
- Download the JAR to your computer.
- Decompile it with a tool like [JD-Gui](#) Most developers do not obfuscate their Java code, and thus the source code can be read in plain text
- Save the decompiled source code
- Analyze the Java source code. Sometimes you might even find server's credentials ?
- Modify the source code at your will. Like modding the inner class's variable of your player object to initialize it with 99999999999999999999 life.
- Re-compile the modified client. Many tutorials out there on how to re-compile a jar file. After all a .jar is nothing more than a fancy ZIP file
- Run the client

Dynamic Analysis:

- Download Wireshark (or tcpdump if you are in linux) and Burp
- Run Wireshark
- Open your game
- In Wireshark, filter the network traffic and check out what protocol is using to communicate to the server endpoint
- Many FB games use a REST API to communicate the server with the client. Therefore i'm going to assume that [Burp](#) will help you in this case, as the client will be probably using HTTP as protocol.
- Open burp, and proxyify the traffic. Depending if you are doing this in linux or Windows you could go with iptables or [proxifier](#).
- Ideally, by now you will be able to see the network traffic and tamper the HTTP requests sent by the game. For example, imagine that in the game you shoot someone and win one point, so the game would tell the server to add you one point and send an HTTP request like:

Or if it's using JSON:

If you tamper the traffic, this will allow you to modify them and do things like:

Optionally, you could do all sort of actions in the game and after a while have a general idea of all that requests that could be send. This applies with other protocols as well.

Hybrid Analysis:

The two approaches above combined. Having the source code at hand and watching what is being passed through the connection between the game and the game server is the way to go. You would have a much bigger understanding of what's happening.

It's possible that you won't be able to intercept the traffic at first if the game uses a secure connection. This because Burp uses it's own self-signed certificate to intercept the traffic. For circumventing this barrier you would need to open up the jar's keystore, export the Burp CA and add it there.

Also the JAR could have a sanity check for each resource, it's a hash verification check. This is to avoid custom modifications ensuring that it's the standard client running, and applies even for the keystore in some case.

This barrier can be bypassed in different ways but this depends on how that resource checksum is implemented (if any). Note: the default password of the Java keystore is usually: 'changeit' and no one changes it ?

This can vary widely but everything described above is based on most common scenarios.

Good luck ?

amazing, definantly deserved a rep!~ would i just inject it? by client its run via website as you know, first method is changing variables etc, would i just inject as its not something i can replace like copy paste since its a file?

Lexxi. - September 28, 2016, 11:48 am

UPDATE:

i figured

https://iappsbeats.com/bbuilder/facebook/index_ios.php?

so index_ios shows device as well.. ios, changing the "ios" to android makes you seem like android device.

its a .php so easy to mess around with.

using: https://iappsbeats.com/bbuilder/facebook/index_android.php?signed_request=

pretty much logs into facebook acc youre identified with the signed_request=

overall its no different than the apps and fb connecting to a link like a webbrowser just fullscreen

this link is the page source, commands etc.. java code: just paste the hole line in google chrome

view-source:https://iappsbeats.com/bbuilder/facebook/index_android.php?