



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ
ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

Ασφάλεια Ασύρματων & Κινητών Δικτύων Επικοινωνιών

Attacking DTLS Server via Android Botnet

Πέππας Κωνσταντίνος, icsd11134
Σωτηρέλης Χρήστος, icsd12182
Χαϊκάλης Νικόλαος, icsd12200

22/04/2016

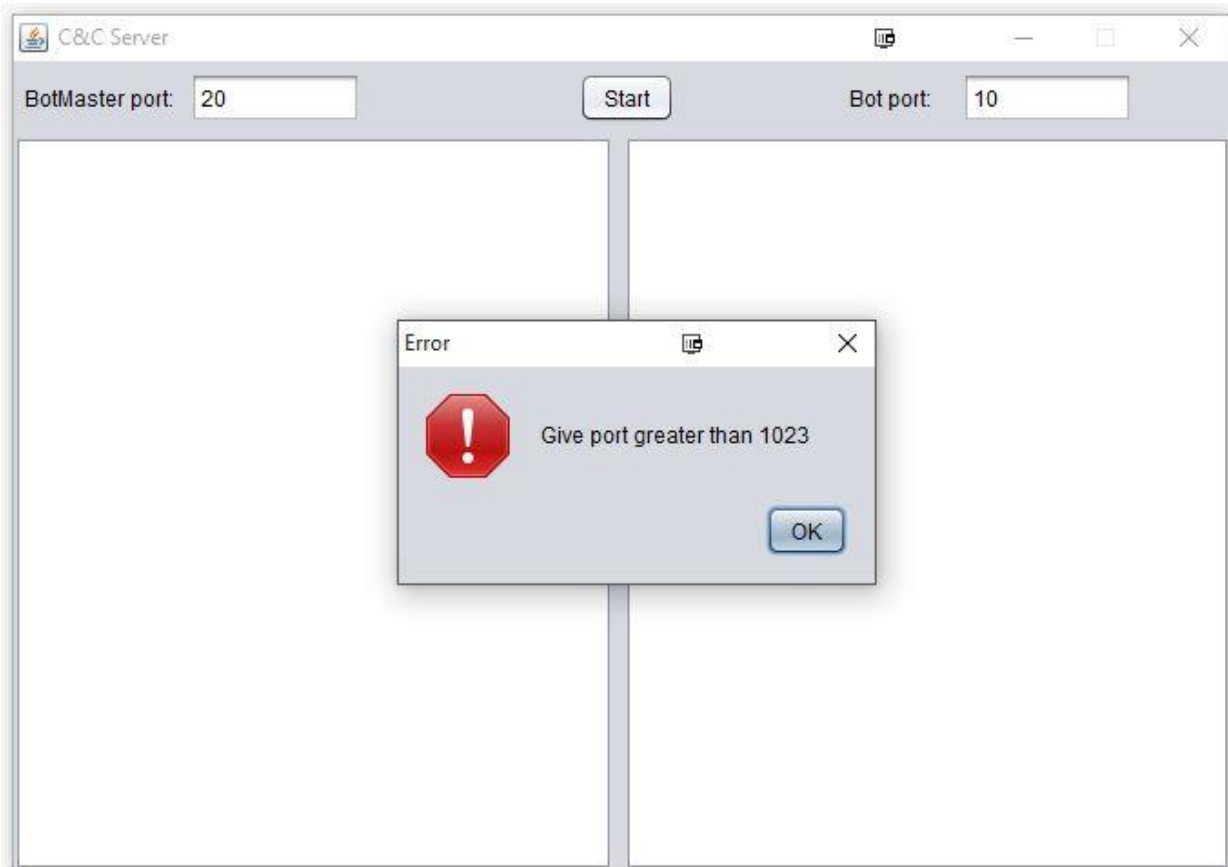
Περιγραφή Έργου

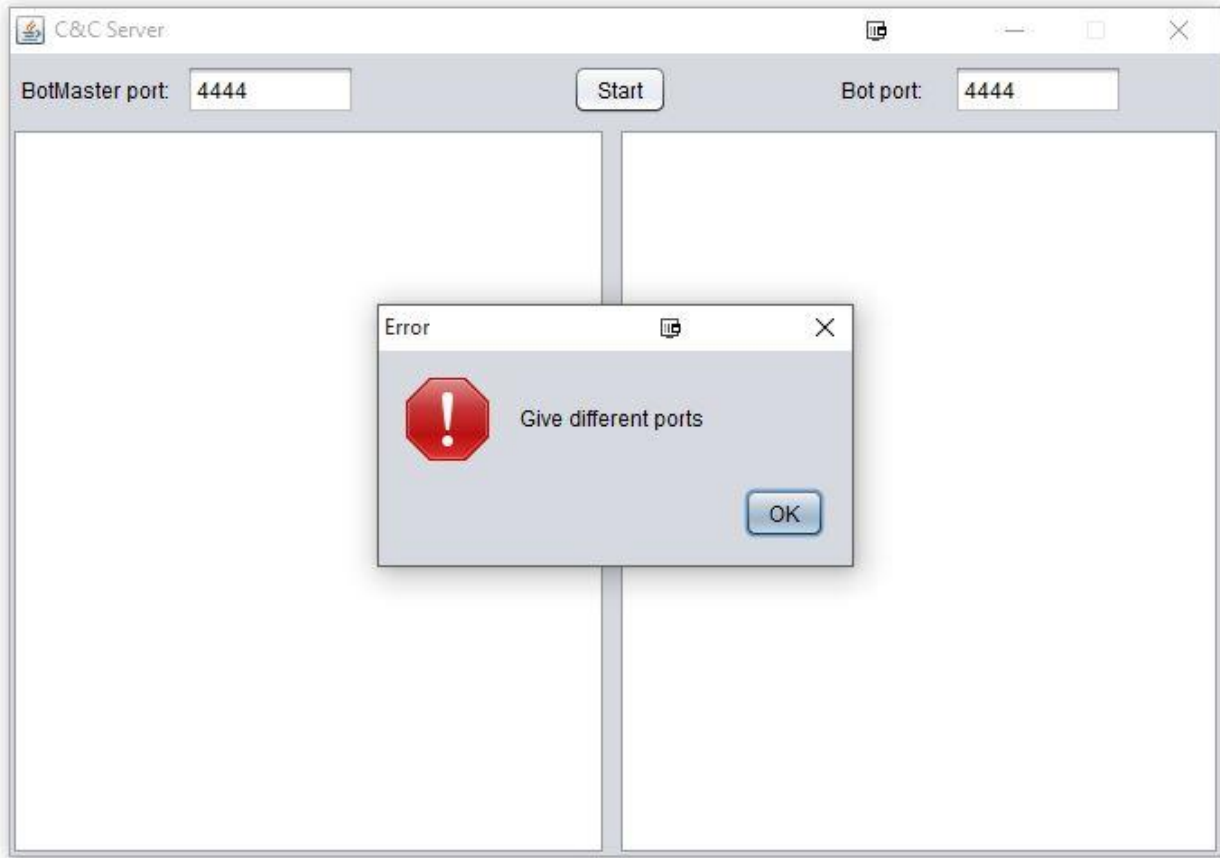
Το project που υλοποιήθηκε στα πλαίσια του μαθήματος αφορά ένα botnet κινητών με λειτουργικό σύστημα Android. Με εντολή του Botmaster, το δίκτυο πραγματοποιεί DTLS DoS. Στο παρακάτω link βρίσκονται όλα τα projects του έργου (servers / clients) καθώς και όλα τα certificates και keystores αυτών:

<https://pithos.oceanos.grnet.gr/public/OjLQiUHZ9hqVK7C4lWhwF1>

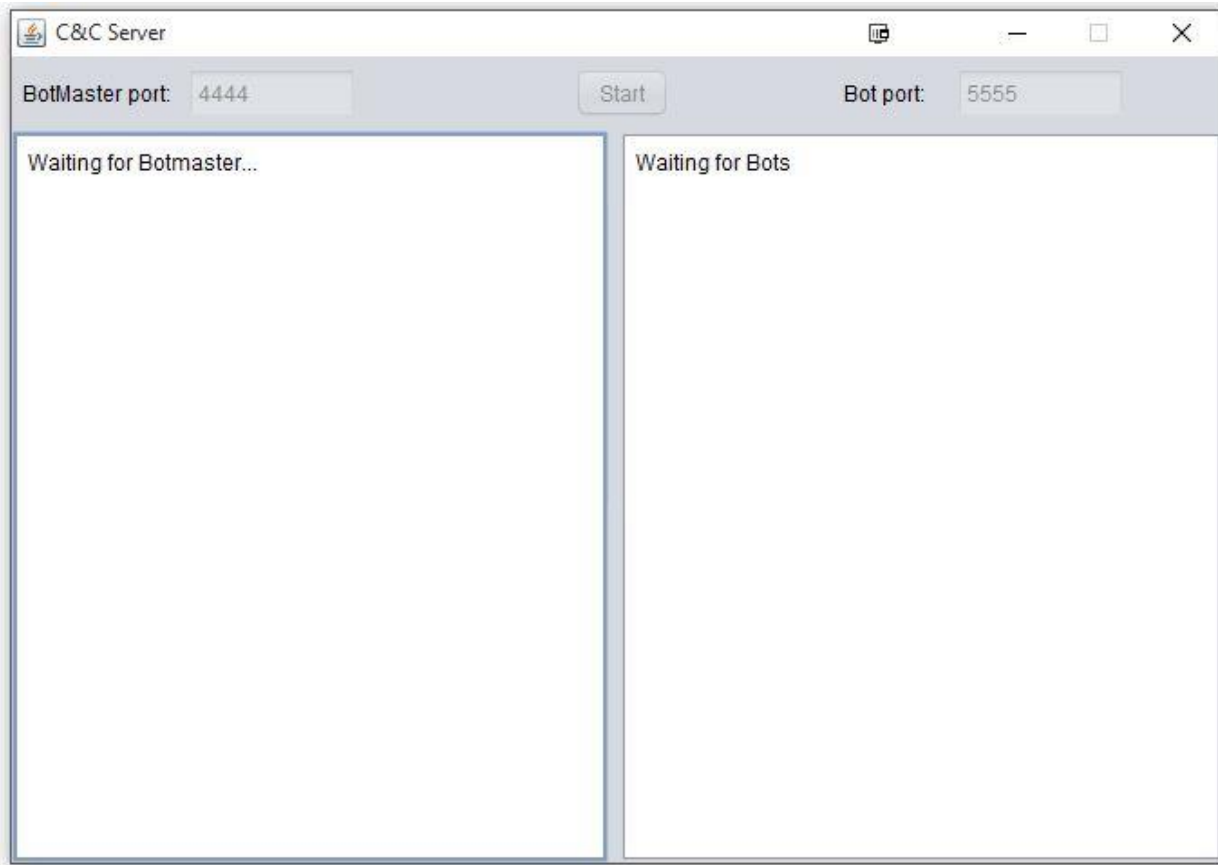
Λειτουργία Έργου

Πιο συγκεκριμένα, έχουμε τον C&C Server στον οποίο ελέγχουμε κάποιους περιορισμούς στα ports που δέχεται.





Αφού θέσουμε έγκυρα ports, ο C&C Server είναι έτοιμος να δεχθεί συνδέσεις από τον Botmaster και τα bots. Ο Botmaster είναι client του C&C.



Για να συνδεθεί ο Botmaster με τον C&C Server μέσω Tor, χρειάζονται ένα RP το οποίο περιλαμβάνεται σε ένα .onion που ο Botmaster ανεβάζει στο Dropbox. Έπειτα, ο C&C Server κατεβάζει το .onion ώστε να συνδεθούν μεταξύ τους. Σε ένα ρεαλιστικό σενάριο, η μεταφορά του .onion θα γινόταν μέσω δικού μας secure καναλιού (πχ. web server) αλλά για τους σκοπούς της συγκεκριμένης εργασίας χρησιμοποιήθηκε η cloud υπηρεσία, Dropbox.



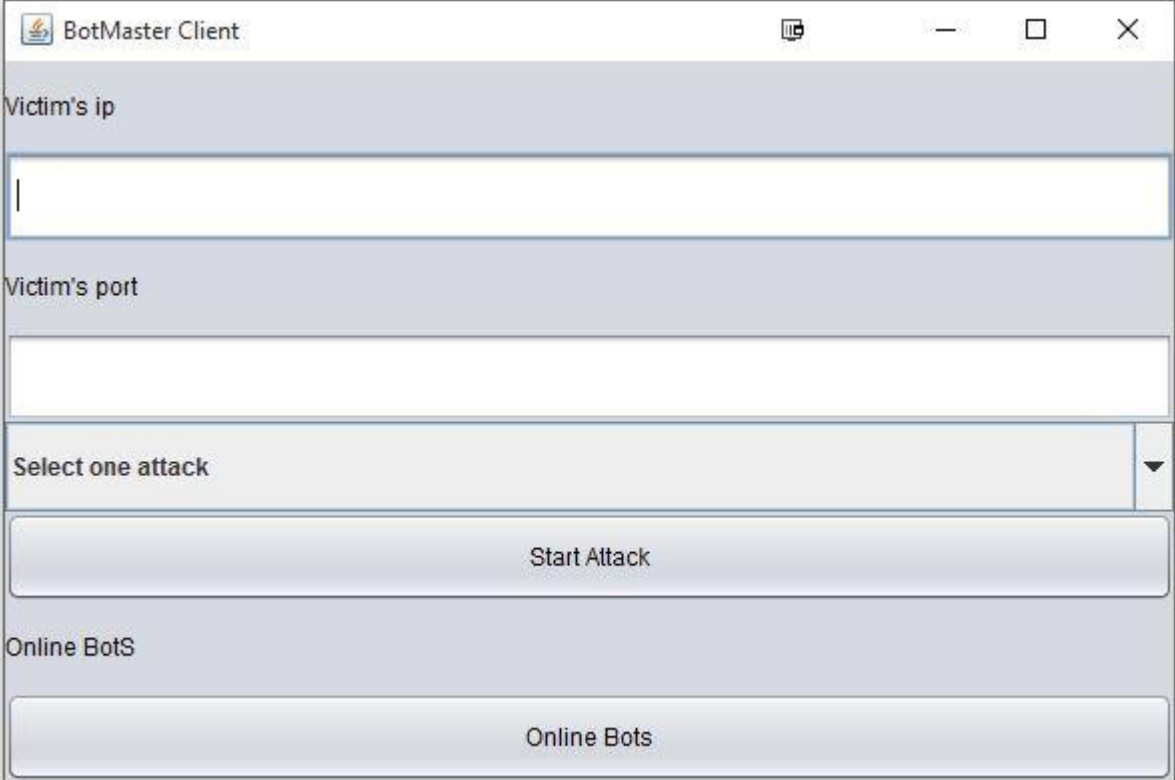
Στη συνέχεια, ο C&C δημιουργεί ένα ζευγάρι Private/Public Keys και στέλνει το Public Key στον Botmaster. Με τη σειρά του, ο Botmaster δημιουργεί ένα Secret Key, το κρυπτογραφεί με το Public Key που έλαβε και στέλνει το κρυπτογράφημα πίσω στον C&C. Ο τελευταίος αποκρυπτογραφεί το κρυπτογράφημα με το δικό του Private Key και, πλέον, έχει στη διάθεσή του το Secret Key, το οποίο θα χρησιμοποιηθεί για να κρυπτογραφεί κάθε μήνυμα που ανταλλάσσεται. Στο σημείο αυτό, η επικοινωνία μεταξύ των δύο οντοτήτων πραγματοποιείται μέσω Tor και κάθε μήνυμα πριν και μετά το Tor είναι κρυπτογραφημένο.

```
Output x
CnC (Server) (run) x BotMaster_Android (run) x
run:
Host from dropbox: qjimi3v9p33kl7.onion
Host from dropbox: qjimi3v9p33kl7.onion
Host from dropbox: p7ubidotnys34u7s.onion
CnC Server> Sun RSA public key, 1024 bits
modulus: 9348360210101544319973482433929254309151615029164602179017928165945170162131771347190486099013128639697463280305643348631215425217180405486864406820872859609146028973639763966837401066830:
public exponent: 65537
CnC Server> [B@178c9109
CnC Server> [B@156b089
readByte[B@3f9148d3
in: java.io.ByteArrayInputStream@3baedbf9
Obj: Hello proxy :D !!!
CnC Server> [B@7afa774c
```

```
Output x
CnC (Server) (run) x BotMaster_Android (run) x
run:
uploading:: p7ubidotnys34u7s.onion

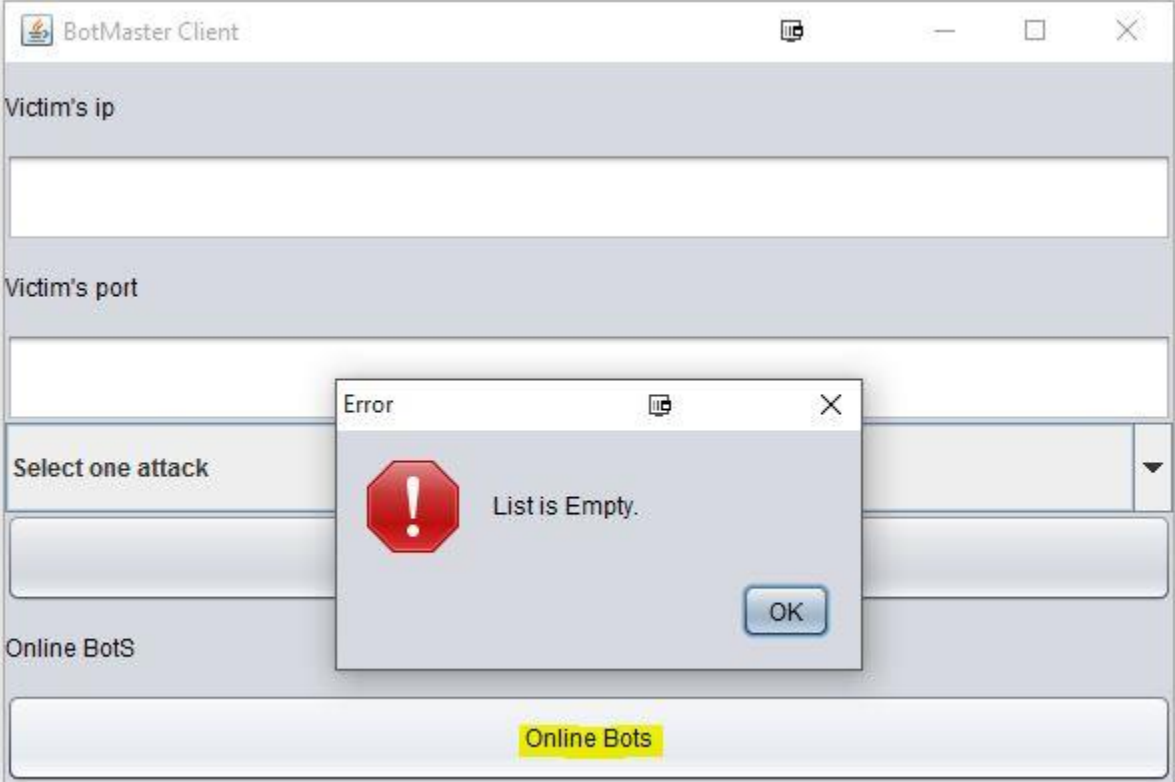
TOR connection is ready.
BotMaster> Sun RSA public key, 1024 bits
modulus: 934836021010154431997348243392925430915161502916460217901792816594517016213177134719048609:
public exponent: 65537
[B@20e2cbe0
[B@65cdee3b
readByte[B@20cf6ce6
in: java.io.ByteArrayInputStream@6393aa69
readObj: Welcome my Master.
|
```

To GUI του Botmaster:



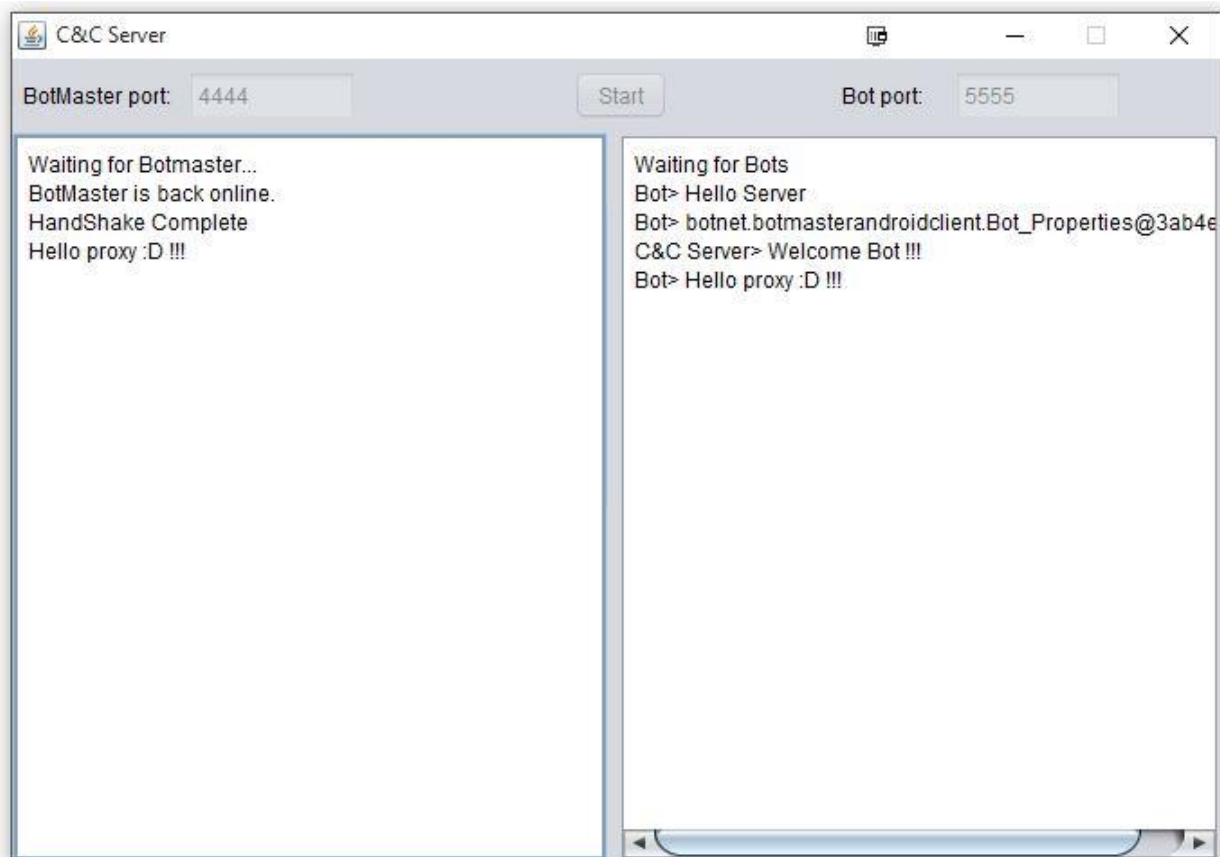
The screenshot shows the 'BotMaster Client' window. It contains the following elements:

- Victim's ip:** A text input field that is currently empty.
- Victim's port:** A text input field that is currently empty.
- Select one attack:** A dropdown menu with a downward arrow on the right side.
- Start Attack:** A button located below the dropdown menu.
- Online BotS:** A label above a list box.
- Online Bots:** A button located at the bottom of the list box.

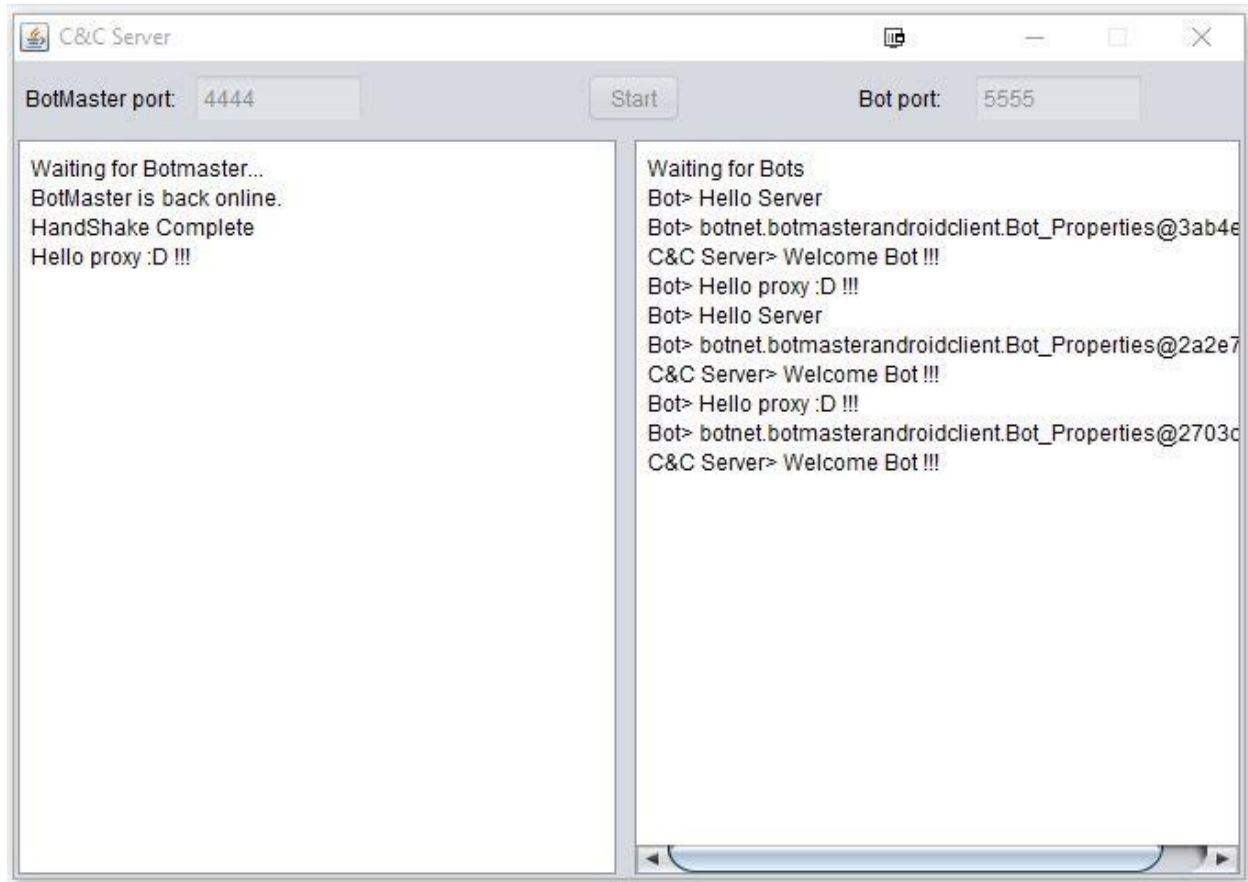


This screenshot shows the same 'BotMaster Client' window, but with an error dialog box overlaid in the center. The dialog box has the title 'Error' and contains a red octagonal warning icon with a white exclamation mark. To the right of the icon, the text 'List is Empty.' is displayed. An 'OK' button is located at the bottom right of the dialog box. In the background, the 'Online Bots' button in the main window is highlighted with a yellow background.

Παρακάτω βλέπουμε ότι στον C&C Server έχουν συνδεθεί ο Botmaster και το 1^ο Android Bot.



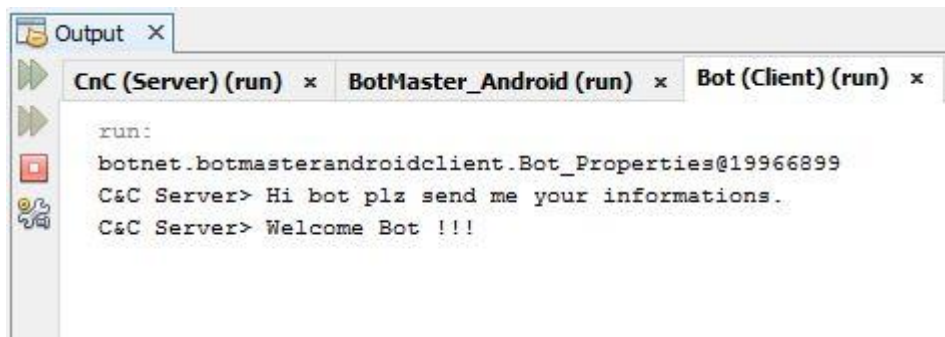
Εν συνεχεία, συνδέουμε ακόμα 2 Android Bots καθώς και 1 Desktop Bot, αφού έχουν τη δυνατότητα να λειτουργήσουν αρμονικά.



Δίνεται, επιπρόσθετα, η λειτουργία να βλέπουμε όλα τα συνδεδεμένα Bots στο Botnet μας.

Host Name	Host Address
Android: localhost	127.0.0.1
Android: localhost	127.0.0.1
MarvinTheMartian	169.254.80.80

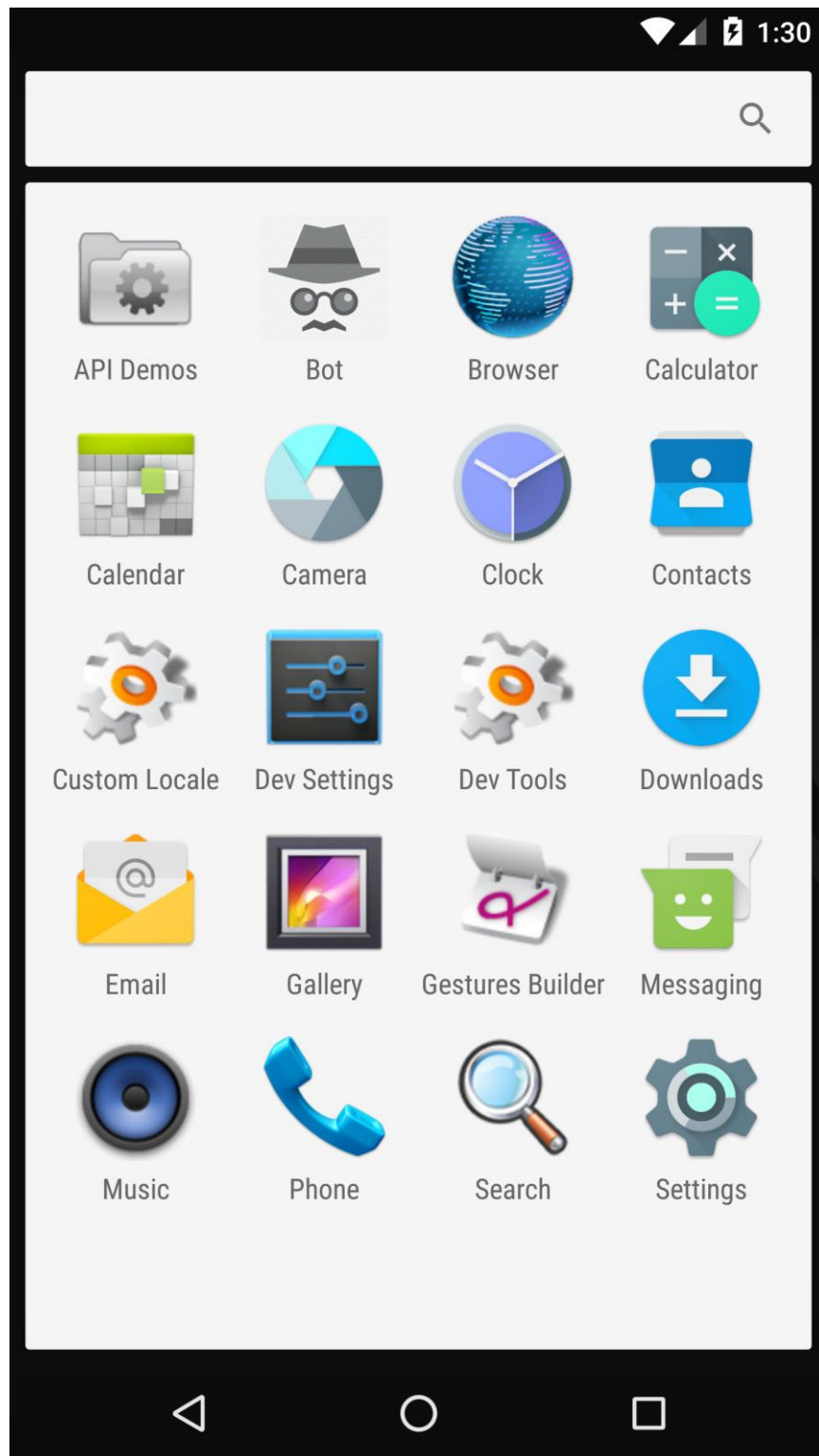
Ένδειξη ότι το Desktop Bot έχει συνδεθεί και επικοινωνεί με τον C&C Server.



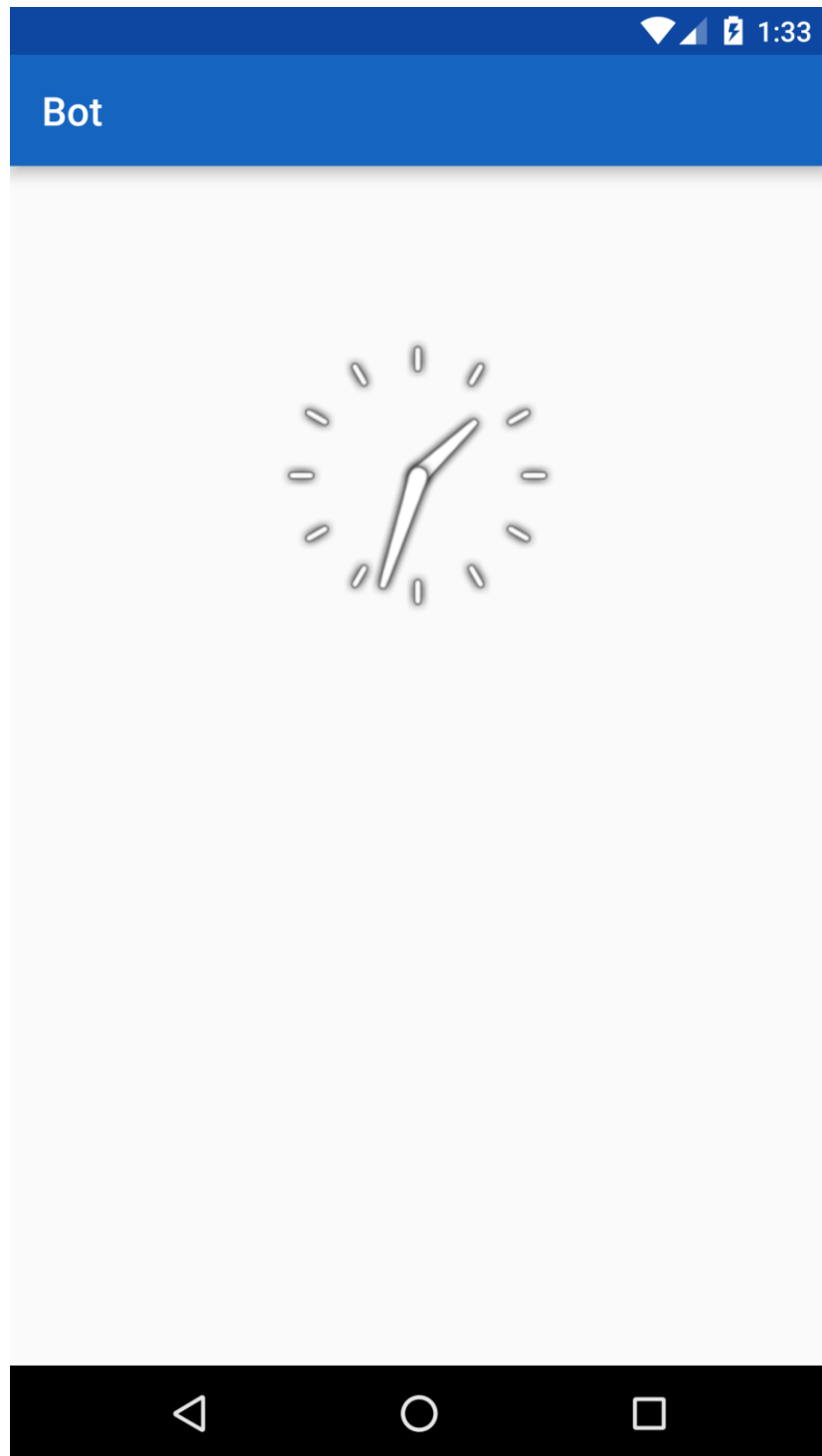
The screenshot shows an IDE's output window with three tabs: 'CnC (Server) (run)', 'BotMaster_Android (run)', and 'Bot (Client) (run)'. The 'Bot (Client) (run)' tab is active, displaying the following text:

```
run:
botnet.botmasterandroidclient.Bot_Properties@19966899
C&C Server> Hi bot plz send me your informations.
C&C Server> Welcome Bot !!!
```

Android Bot:



Παρακάτω φαίνεται το γραφικό περιβάλλον του Android Bot ενώ τρέχει και συνδέεται στον C&C Server.



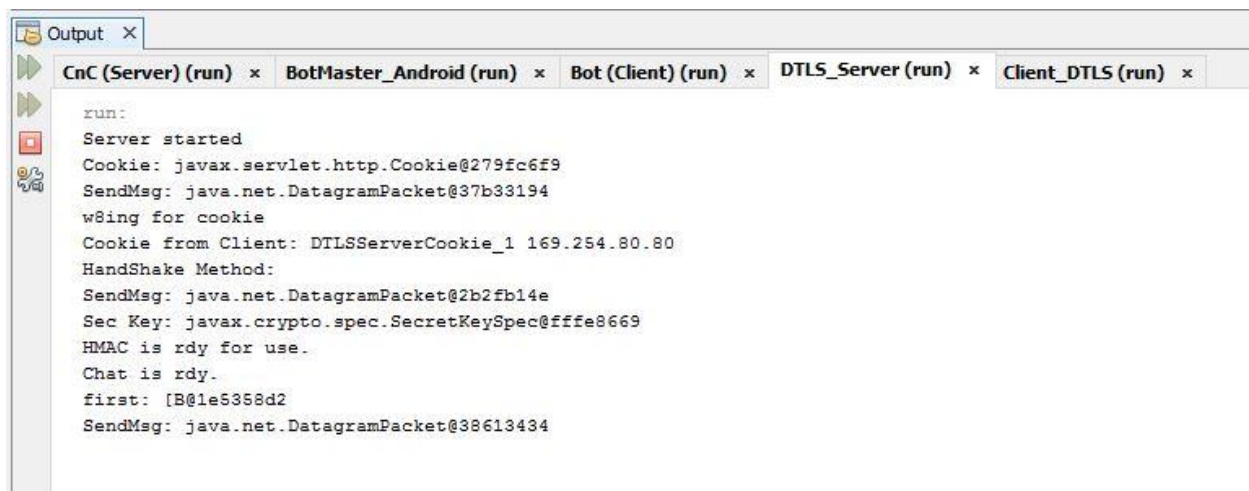
Στα logs του Android Studio φαίνεται η επιτυχής σύνδεση του Bot με τον C&C Server.

```
04-17 13:02:41.572 2095-2108/? D/msg from server:: Hi bot plz send me your informations.
04-17 13:02:41.573 2095-2108/? D/IP:: 127.0.0.1
04-17 13:02:41.573 2095-2108/? D/name:: Android: localhost
04-17 13:02:41.576 2095-2114/? I/System.out: botnet.botmasterandroidclient.Bot_Properties@fc22b69
04-17 13:02:41.576 2095-2114/? I/System.out: Hello proxy :D !!!
04-17 13:02:41.861 2095-2114/? I/System.out: C&C Server> Welcome Bot !!!
04-17 13:02:41.861 2095-2114/? D/Readobj:: Welcome Bot !!!
```

Για να πραγματοποιήσουμε επίθεση θα χρησιμοποιήσουμε έναν DTLS Server που έχουμε γράψει σε Java. Κατά την εκκίνησή του, ο server αναμένει συνδέσεις από clients. Κάθε φορά που κάποιος client στέλνει “hello”, ο server δημιουργεί ένα cookie και το στέλνει πίσω σε αυτόν. Στη συνέχεια, ο server περιμένει να λάβει πίσω το ίδιο cookie από τον client ώστε να πραγματοποιηθεί η διαδικασία ταυτοποίησης του με certificates. Αν λάβει το σωστό cookie, τότε ξεκινάει η επικοινωνία μέσω SSL/TLS.

Αξίζει να σημειωθεί πως επειδή δεν βρέθηκε έτοιμο SSL/TLS over UDP Socket, χρησιμοποιούμε δική μας custom υλοποίηση.

Στο σημείο αυτό, αφού έχει πραγματοποιηθεί ασφαλής αποστολή ενός μηνύματος (client hello), ο server δεν πραγματοποιεί καμία άλλη ενέργεια με τον client, παρά μόνο ένα loop ώστε να μην διακοπεί η επικοινωνία μεταξύ τους. Στο έργο δεν εξετάζουμε την λειτουργία του DTLS Server αλλά μόνο την επίθεση DoS/DDoS σε αυτόν.



The screenshot shows the Android Studio Output window with five tabs: CnC (Server) (run), BotMaster_Android (run), Bot (Client) (run), DTLS_Server (run), and Client_DTLS (run). The logs for the DTLS_Server (run) tab are visible, showing the server's startup and the handshake process with a client.

```
run:
Server started
Cookie: javax.servlet.http.Cookie@279fc6f9
SendMsg: java.net.DatagramPacket@37b33194
waiting for cookie
Cookie from Client: DTLS_ServerCookie_1 169.254.80.80
HandShake Method:
SendMsg: java.net.DatagramPacket@2b2fb14e
Sec Key: javax.crypto.spec.SecretKeySpec@fffe8669
HMAC is rdy for use.
Chat is rdy.
first: [B@1e5358d2
SendMsg: java.net.DatagramPacket@38613434
```

```
Output x
CnC (Server) (run) x BotMaster_Android (run) x Bot (Client) (run) x DTLS_Server (run) x Client_DTLS (run) x

run:
SendMsgs: java.net.DatagramPacket@eed1f14
169.254.80.80
Cookie: DTLS_ServerCookie_1
SendMsgs: java.net.DatagramPacket@58372a00
000000
000000
0000000000
`000X000MFOf0f0(000o\00000,LD_u000uDe0z>2000n
0P0)%00+0!Y070A0000;W00y-a000"0`0{g000000000000dQK0?0'0000000;0000{<0M0080000Qx,~N0qh<0~k0Im#0e0Y000y#'00'00010
000h~
R0{0b00|0bKn0Hj%0020A,0zQ04Y00Q0'000') (0z0d%0~0000_00
!'00T000>;0000T000w0W00 0f00?0)pEa#00_)0m_000)i!00 00 M0vB8U000R00z00'000020h0 q0000000703K000w0p000000P00000+0
++This certificate is VALID++
Sec Key: javax.crypto.spec.SecretKeySpec@fffe8669
SendMsgs: java.net.DatagramPacket@3ecf72fd
handshake complete. HMAC is ready.
chat is rdy.
ReadByte: 000ur0{B0000T00xp 0x00h0000$00K0000xwc00A00{000!F
Decrypted msg: hello client
```

Παρακάτω, ορίζουμε στον Botmaster την IP και την port του θύματος και την εκκινούμε με το “Start Attack”.

BotMaster Client

Victim's ip

192.168.1.2

Victim's port

4000

DTLS DoS

Start Attack

Online BotS

Online Bots

BotMaster Client

Victim's ip

192.168.1.2

Victim's port

4000

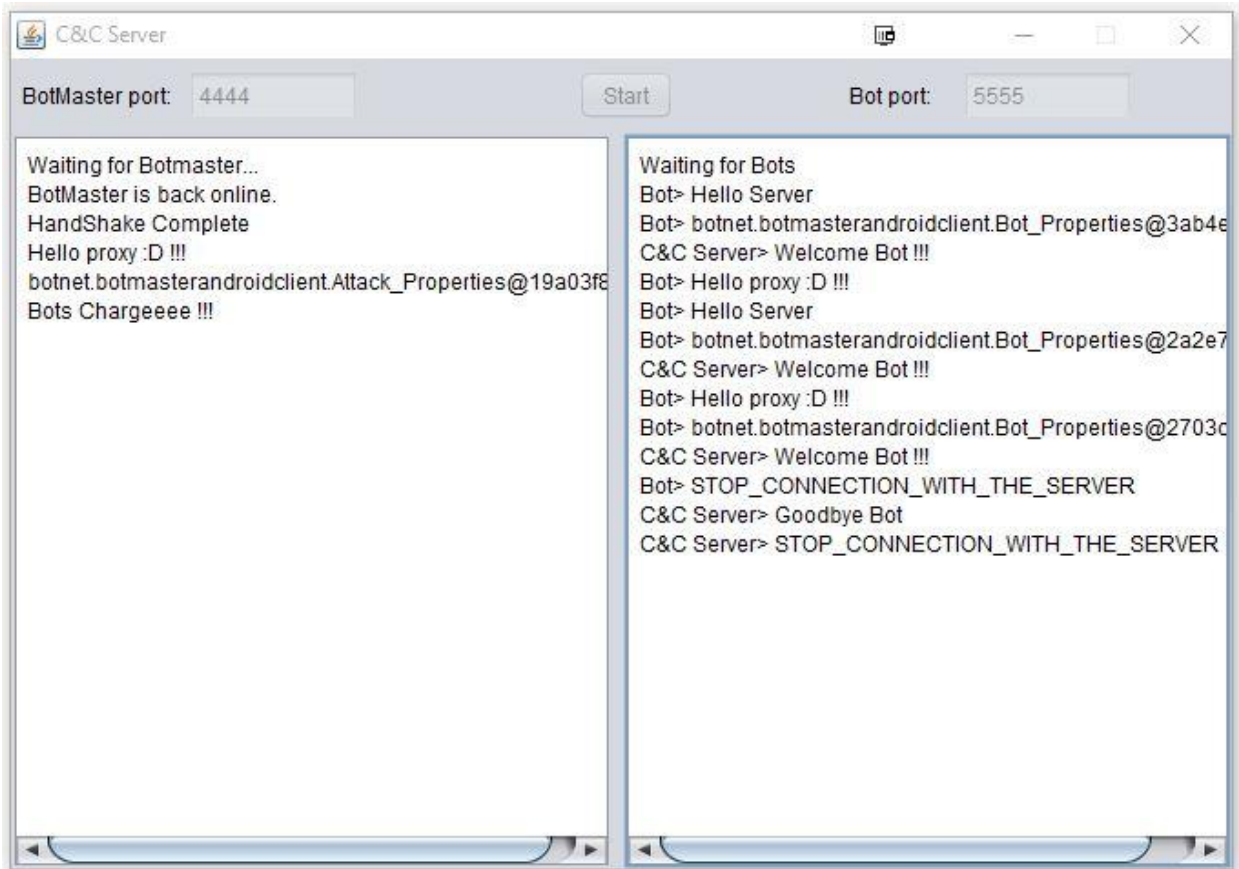
DTLS DoS

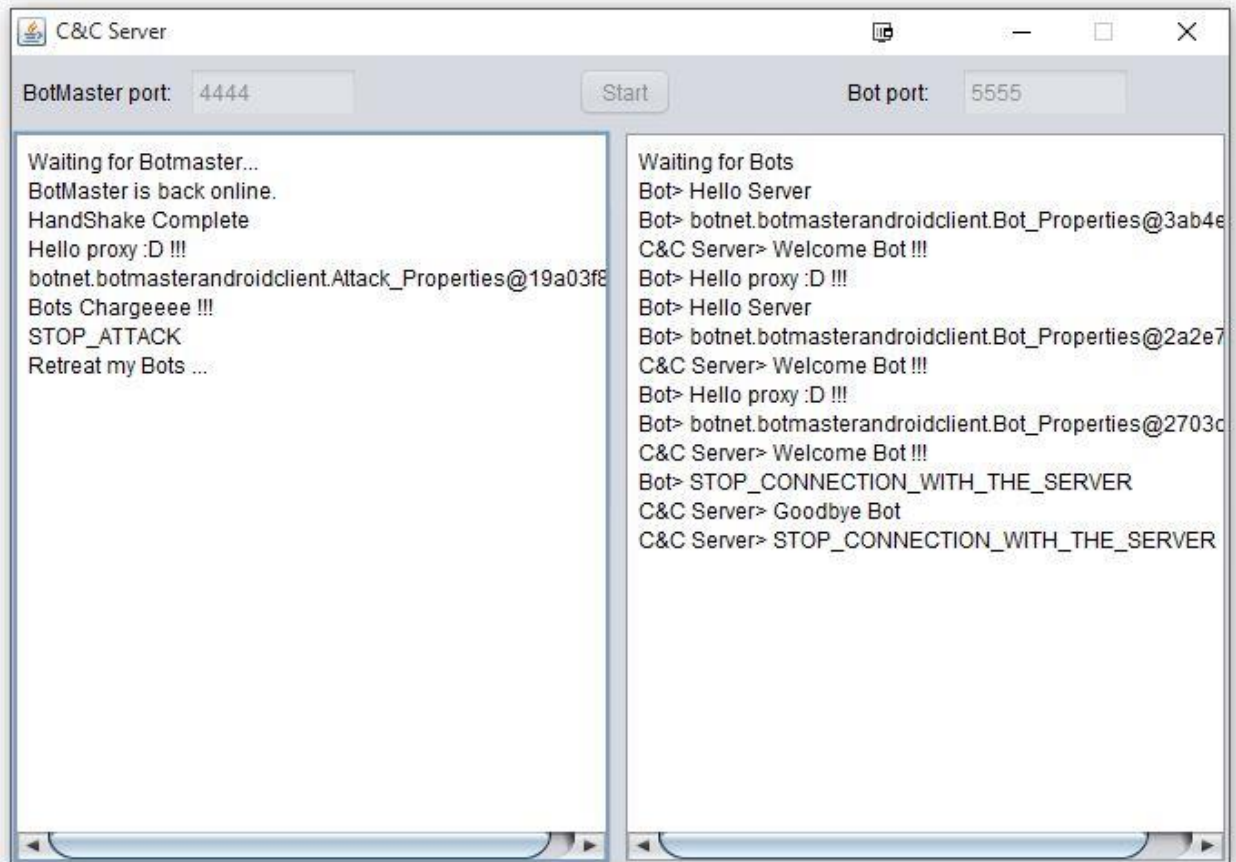
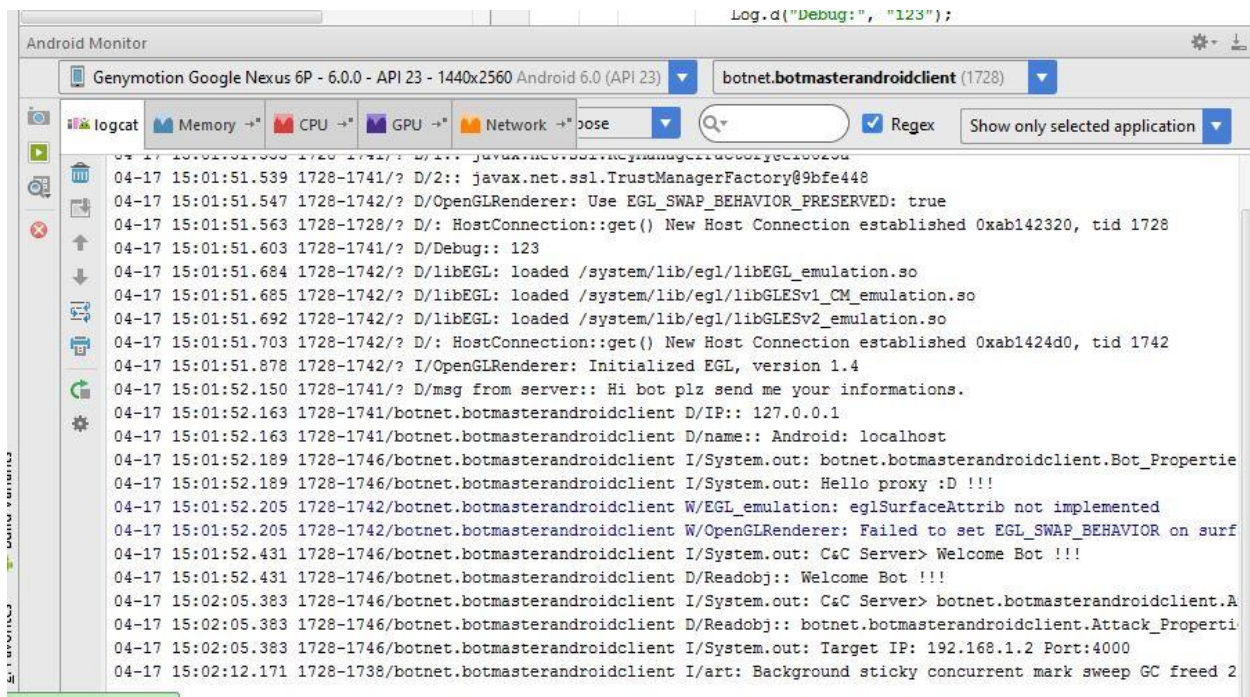
Stop Attack

Online Bots

Online Bots

Βλέπουμε ότι ο C&C Server έχει λάβει εντολή για επίθεση και επικοινωνεί με τα συνδεδεμένα Android/Desktop Bots.








Στα logs του DTLS Server φαίνονται να δημιουργούνται συνέχεια cookies προς τους clients (bots) χωρίς αυτά να το στέλνουν πίσω (επιτυχής DDoS επίθεση).

```
run:
Server started
Cookie: javax.servlet.http.Cookie@279fc6f9
SendMsg: java.net.DatagramPacket@37b33194
w8ing for cookie
Cookie from Client: DTLSSTServerCookie_1 169.254.80.80
HandShake Method:
SendMsg: java.net.DatagramPacket@2b2fb14e
Sec Key: javax.crypto.spec.SecretKeySpec@fffe8669
HMAC is rdy for use.
Chat is rdy.
first: [B@1e5358d2
SendMsg: java.net.DatagramPacket@38613434
Cookie: javax.servlet.http.Cookie@1c90f3d0
SendMsg: java.net.DatagramPacket@536a0409
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@3974aa1d
SendMsg: java.net.DatagramPacket@768e5978
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@685158c1
SendMsg: java.net.DatagramPacket@27aba180
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@63d19174
SendMsg: java.net.DatagramPacket@2c2b7945
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@50d55efe
SendMsg: java.net.DatagramPacket@1f2fa0af
w8ing for cookie
Cookie validation failed
```

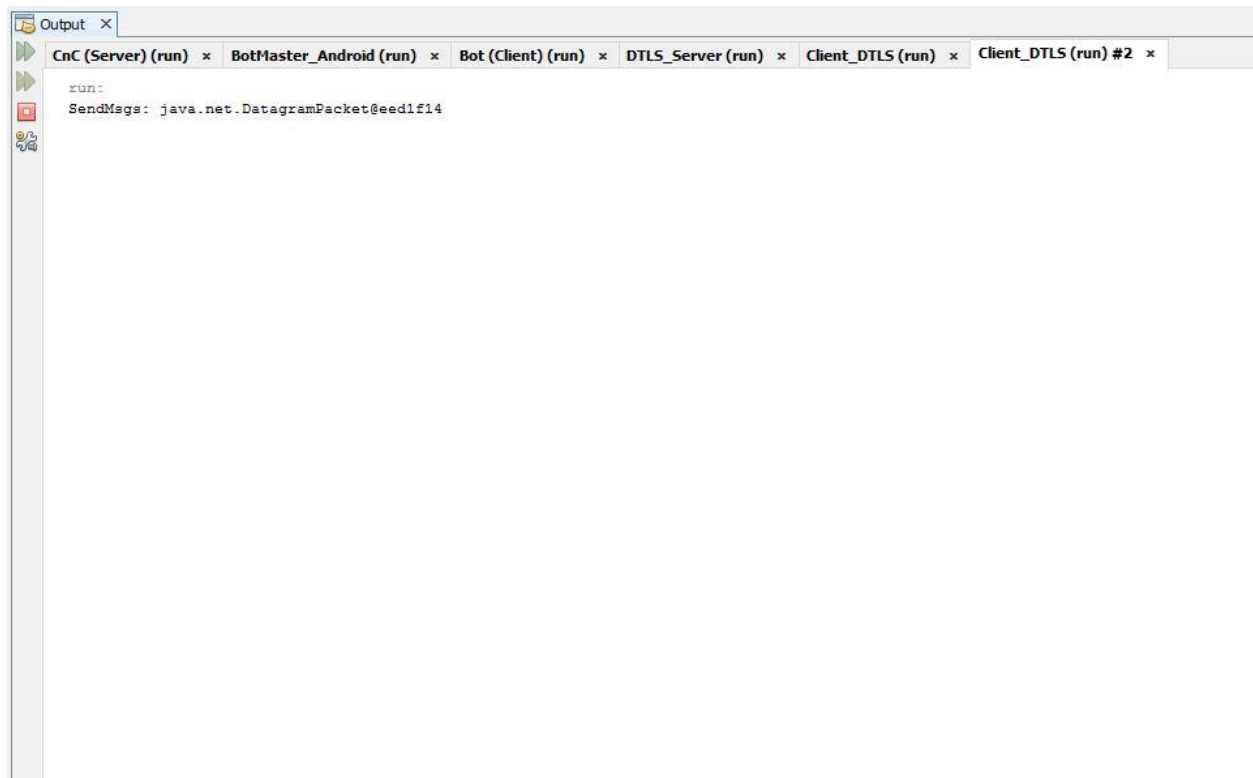
Output x

CnC (Server) (run) x BotMaster_Android (run) x Bot (Client) (run) x DTLS_Server (run) x Client_DTLS (run) x Client_DTLS (run) #2 x



```
Cookie: javax.servlet.http.Cookie@24d0c701
SendMsg: java.net.DatagramPacket@624f667c
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@29c4f418
SendMsg: java.net.DatagramPacket@663e156b
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@4d22f00e
SendMsg: java.net.DatagramPacket@2d80de1
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@43d9754b
SendMsg: java.net.DatagramPacket@c23997b
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@75ad3e87
SendMsg: java.net.DatagramPacket@3a57d479
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@506d0b61
SendMsg: java.net.DatagramPacket@1f97e527
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@1133ea76
SendMsg: java.net.DatagramPacket@5ec90c1b
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@6a0fc530
SendMsg: java.net.DatagramPacket@25f52c3a
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@4d0b6d80
SendMsg: java.net.DatagramPacket@2f1906a
w8ing for cookie
Cookie validation failed
Cookie: javax.servlet.http.Cookie@35dd8ea8
```

Δοκιμή σύνδεσης νέου client στον DTLS Server (αδυναμία λόγω επίθεσης).



Πηγές:

On Improving Resistance to Denial of Service and Key Provisioning Scalability of the DTLS Handshake - *Marco Tiloca · Christian Gehrman · Ludwig Seitz*

“Hackers. We inherently trust no one, including each other.” - Mr. Robot