



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

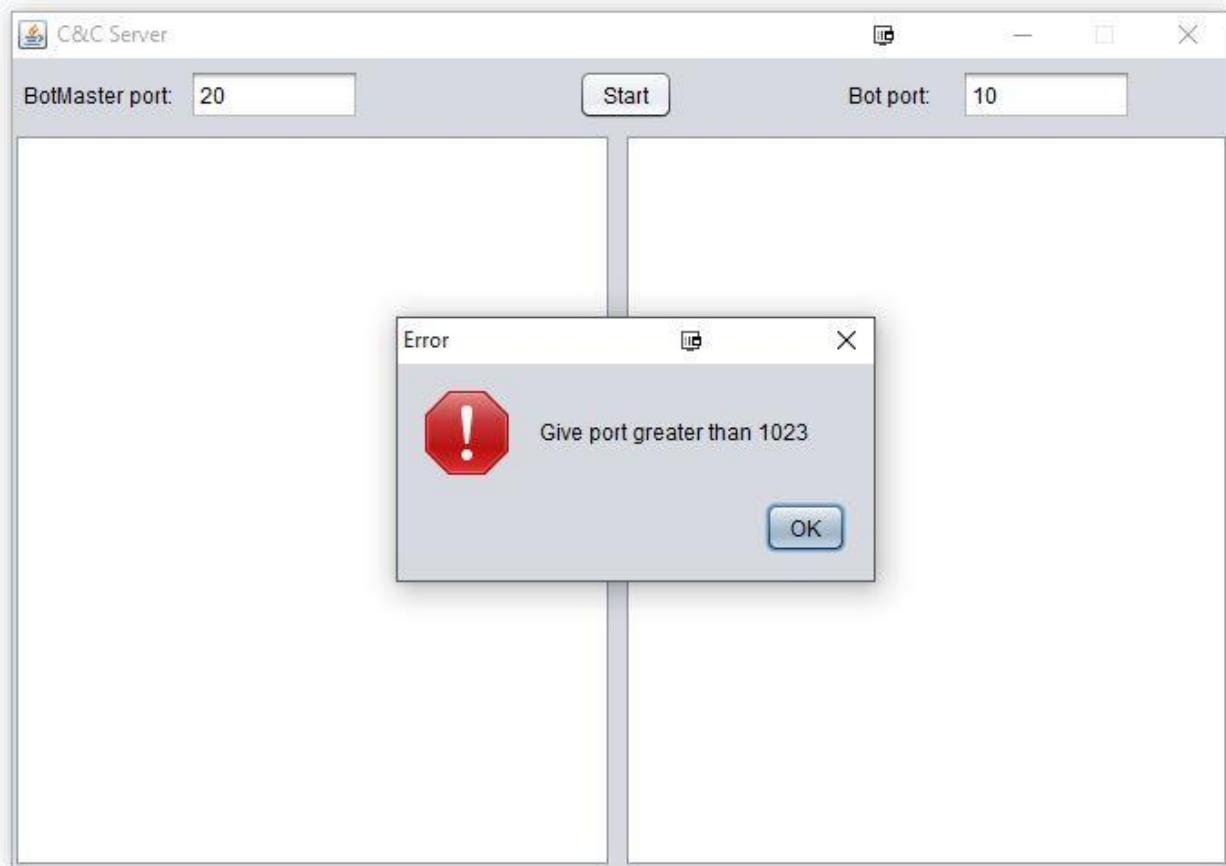
Περιγραφή Έργου

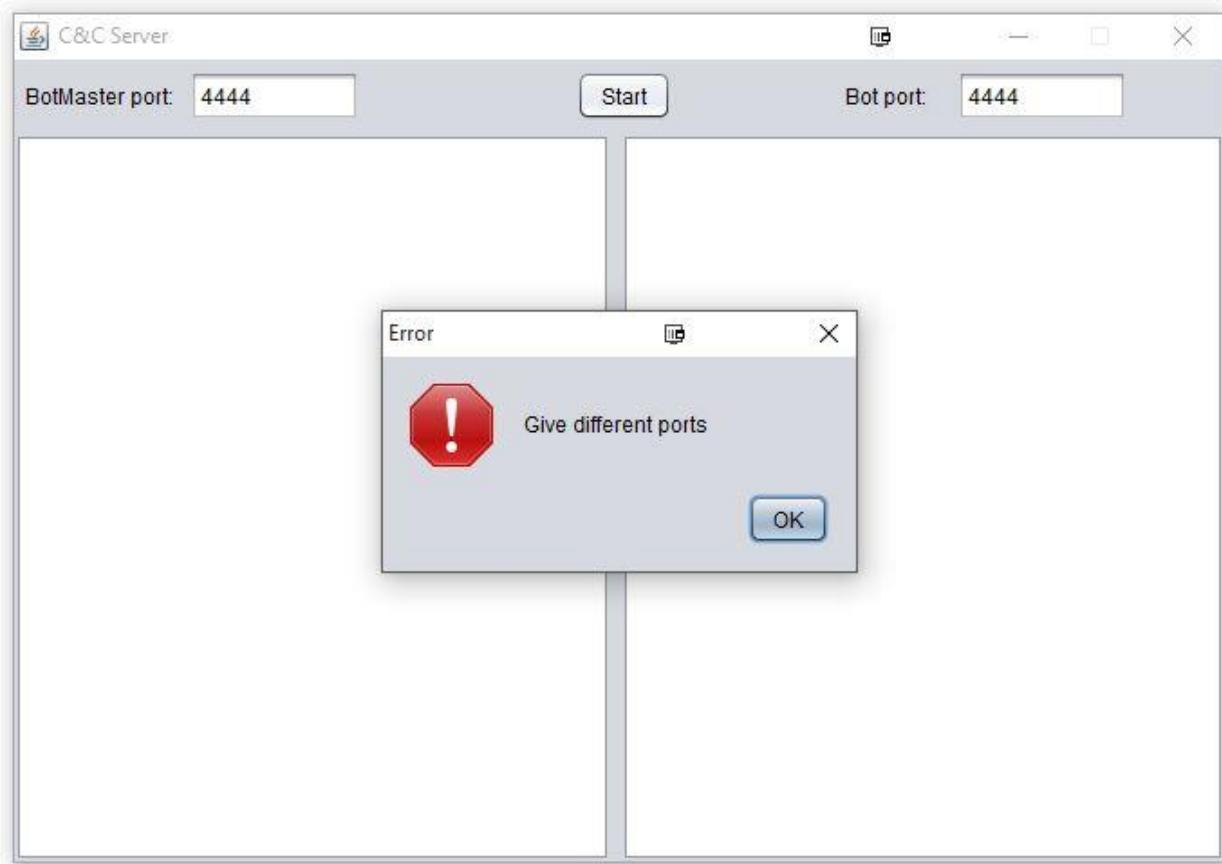
Το project που υλοποιήθηκε στα πλαίσια του μαθήματος αφορά ένα botnet κινητών με λειτουργικό σύστημα Android. Με εντολή του Botmaster, το δίκτυο πραγματοποιεί DTLS DoS. Στο παρακάτω link βρίσκονται όλα τα projects του έργου (servers / clients) καθώς και όλα τα certificates και keystores αυτών:

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

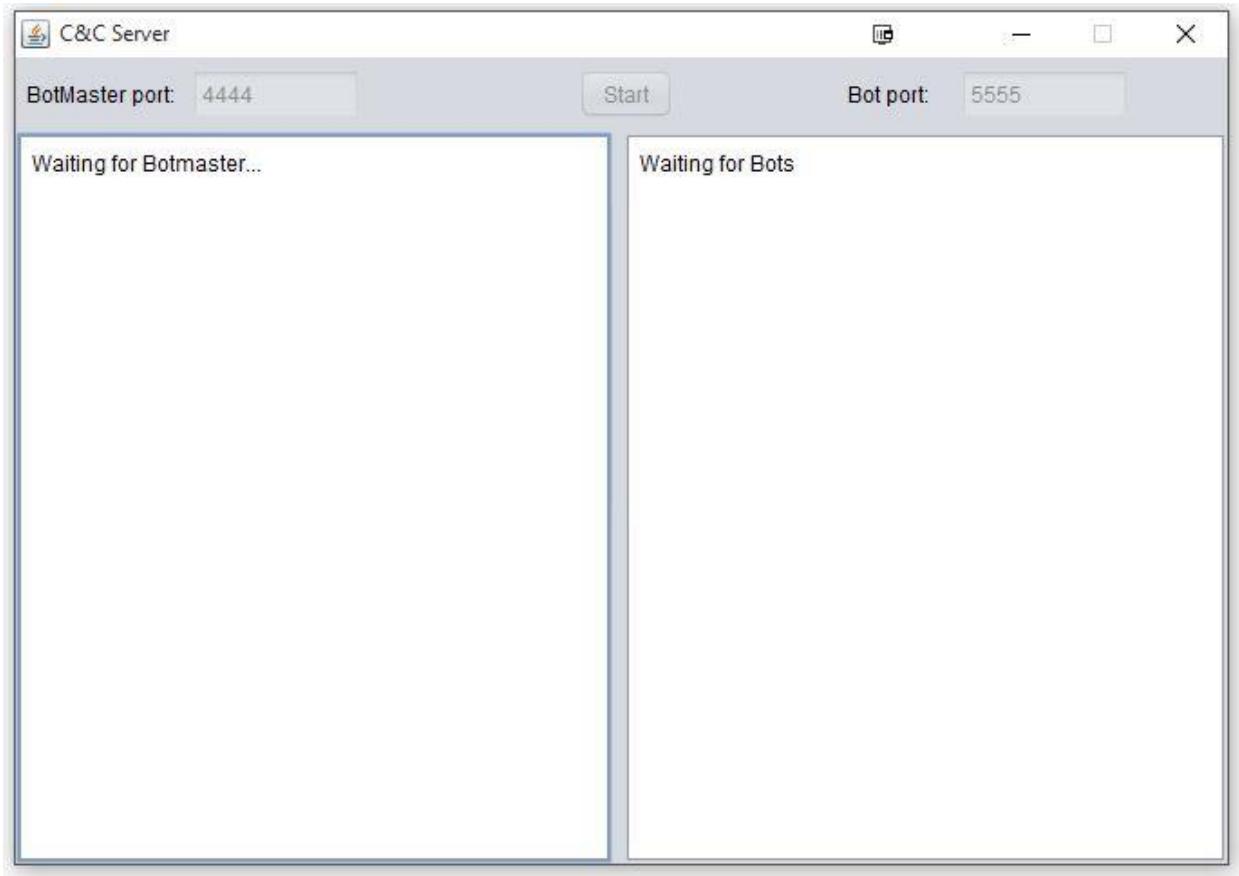
Λειτουργία Έργου

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.

The screenshot displays two terminal windows side-by-side, both titled "Output".

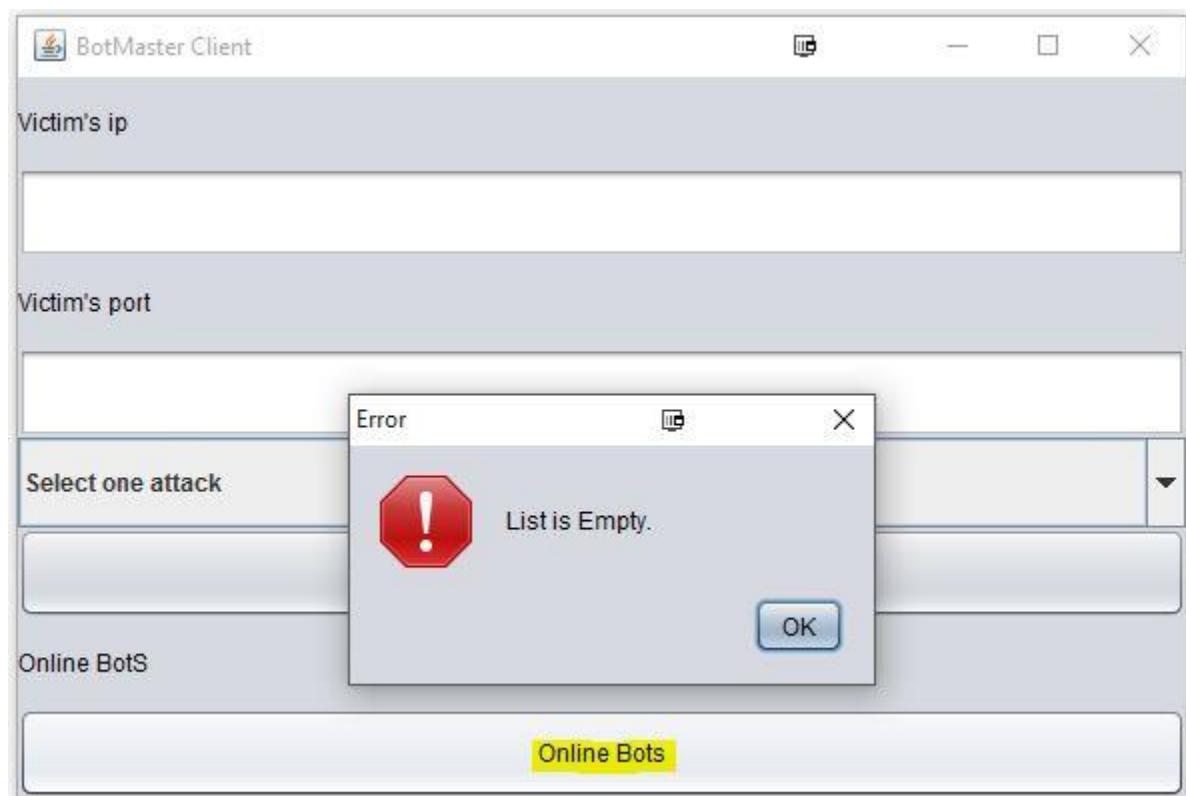
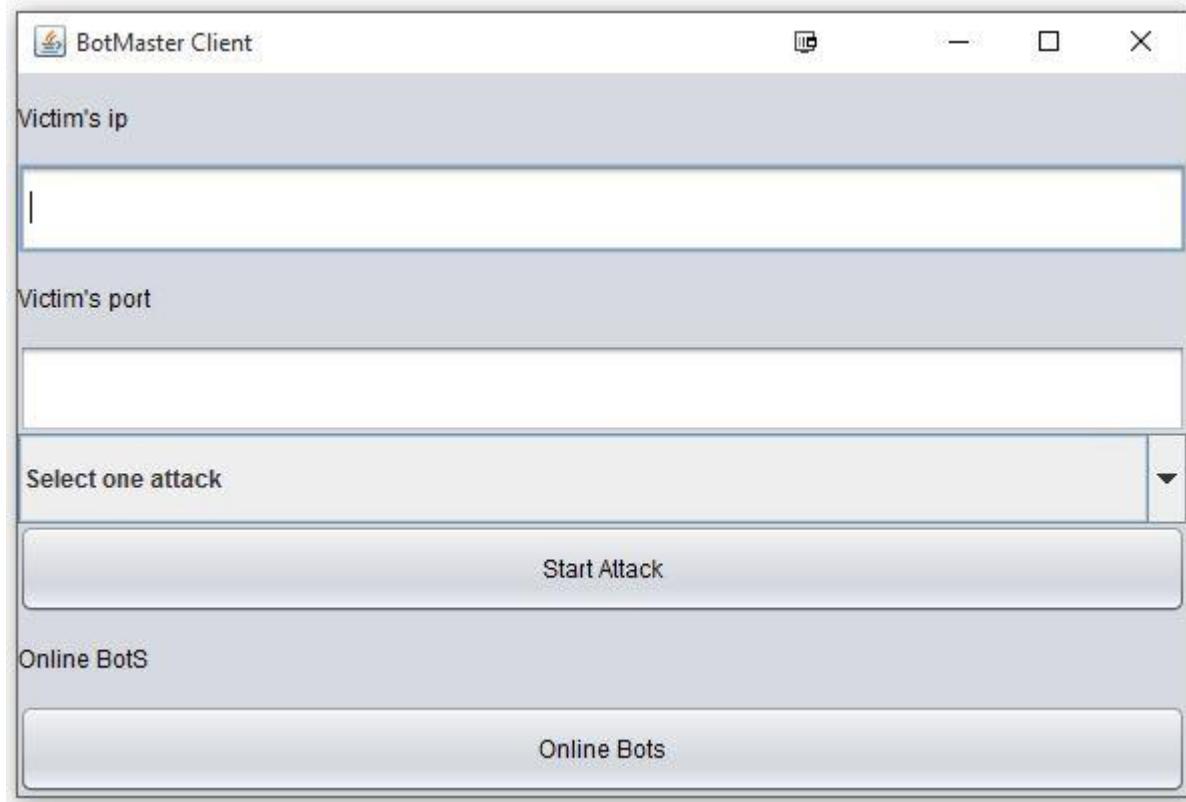
Top Terminal (BotMaster_Android):

```
run:  
Host from dropbox: qjimil3w8pj38k17.onion  
Host from dropbox: qjimil3w8pj38k17.onion  
Host from dropbox: p7ubidotnys34u7s.onion  
CnC Server> Sun RSA public key, 1024 bits  
    modulus: 93483602101015443199734824339292543091516150291646021790179281659451701621317713471904860901312863969746328030054348631215425217180405486864406820872859609146028973639763966837401066880:  
    public exponent: 65537  
CnC Server [B@178c9108  
CnC Server [B@3156e099  
readByte[B@3f9148d3  
in: java.io.ByteArrayInputStream@3baedbf9  
Obj: Hello proxy :D !!!  
CnC Server> [B@7afa774c
```

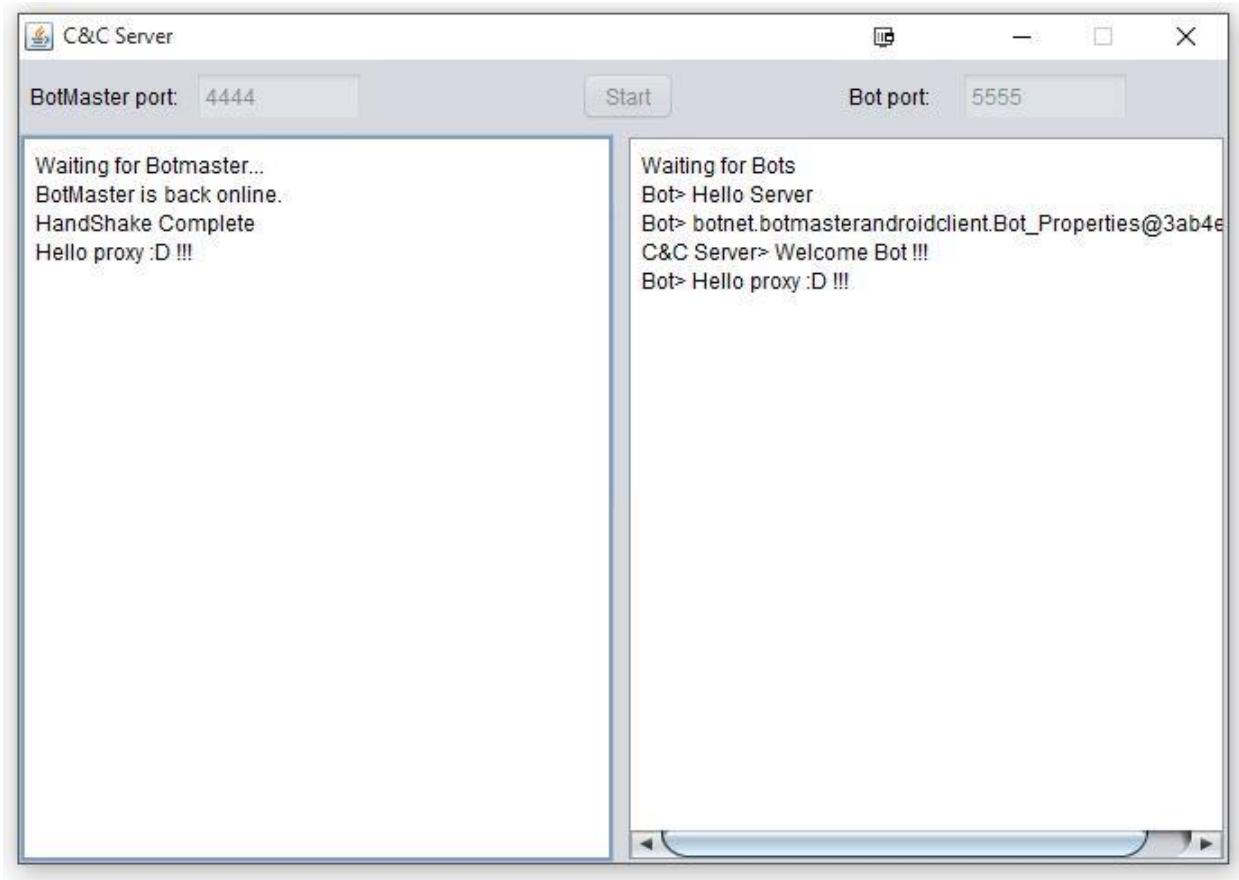
Bottom Terminal (CnC Server):

```
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.
```

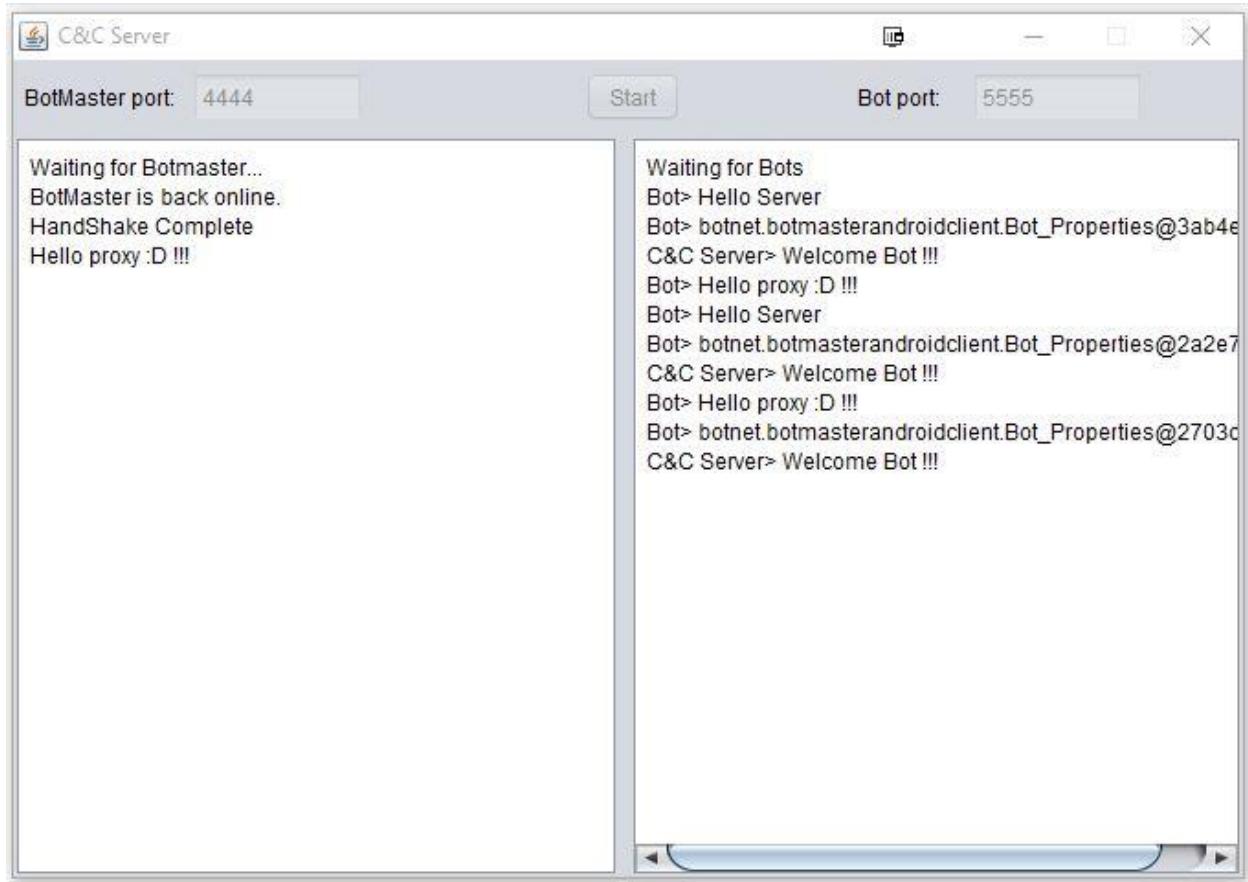
To GUI του Botmaster:



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



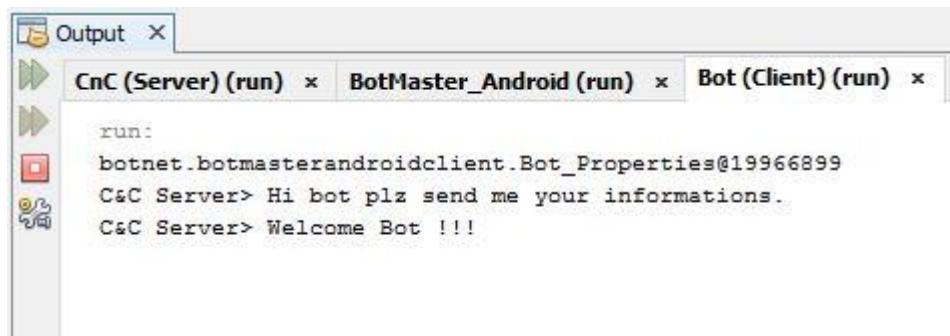
Εν συνεχεία, συνδέουμε ακόμα 2 Android Bots καθώς και 1 Desktop Bot, αφού έχουν τη δυνατότητα να λειτουργήσουν αρμονικά.



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

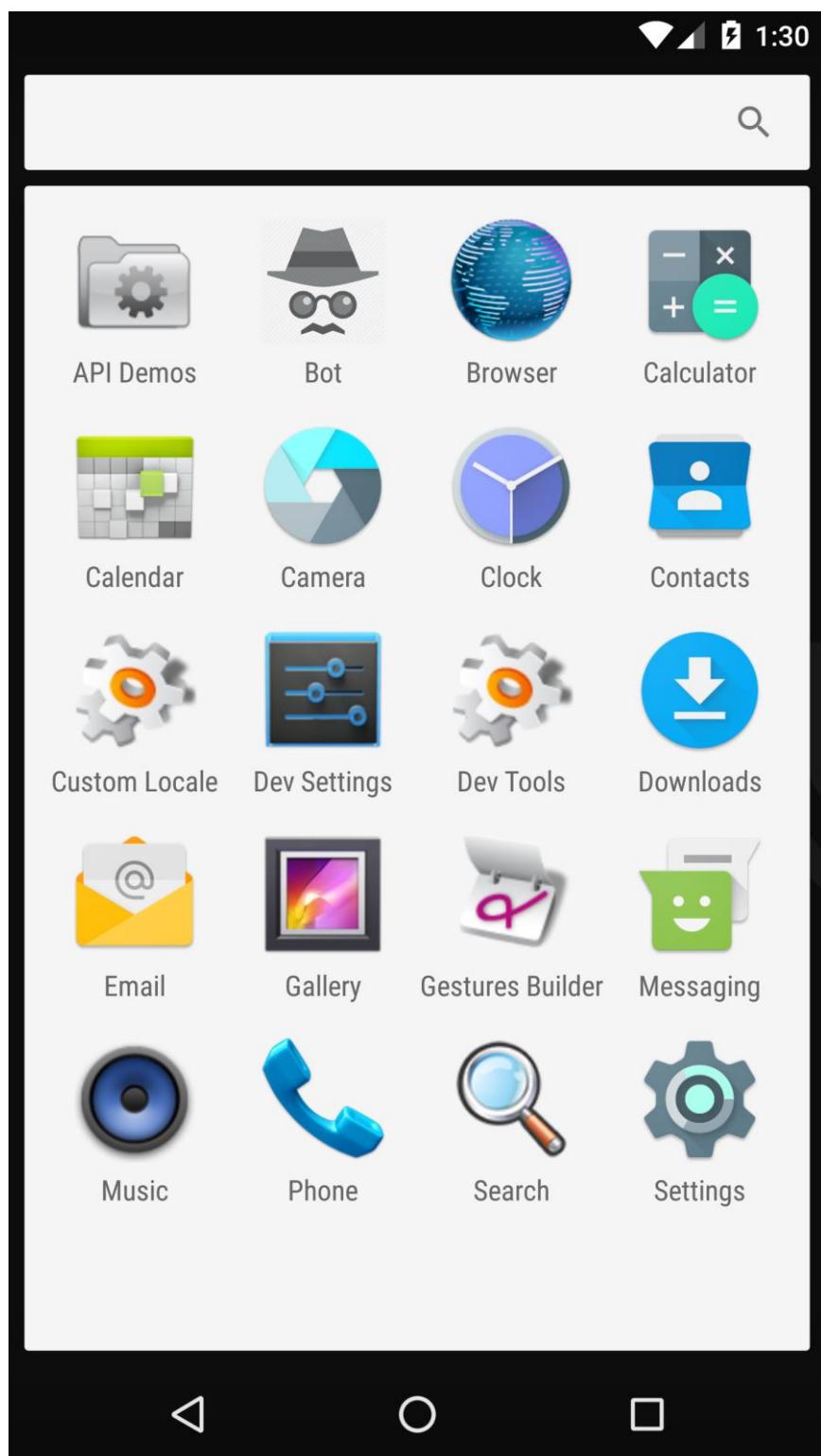
Host Name	Host Address
Android: localhost	127.0.0.1
Android: localhost	127.0.0.1
MarvinTheMartian	169.254.80.80

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

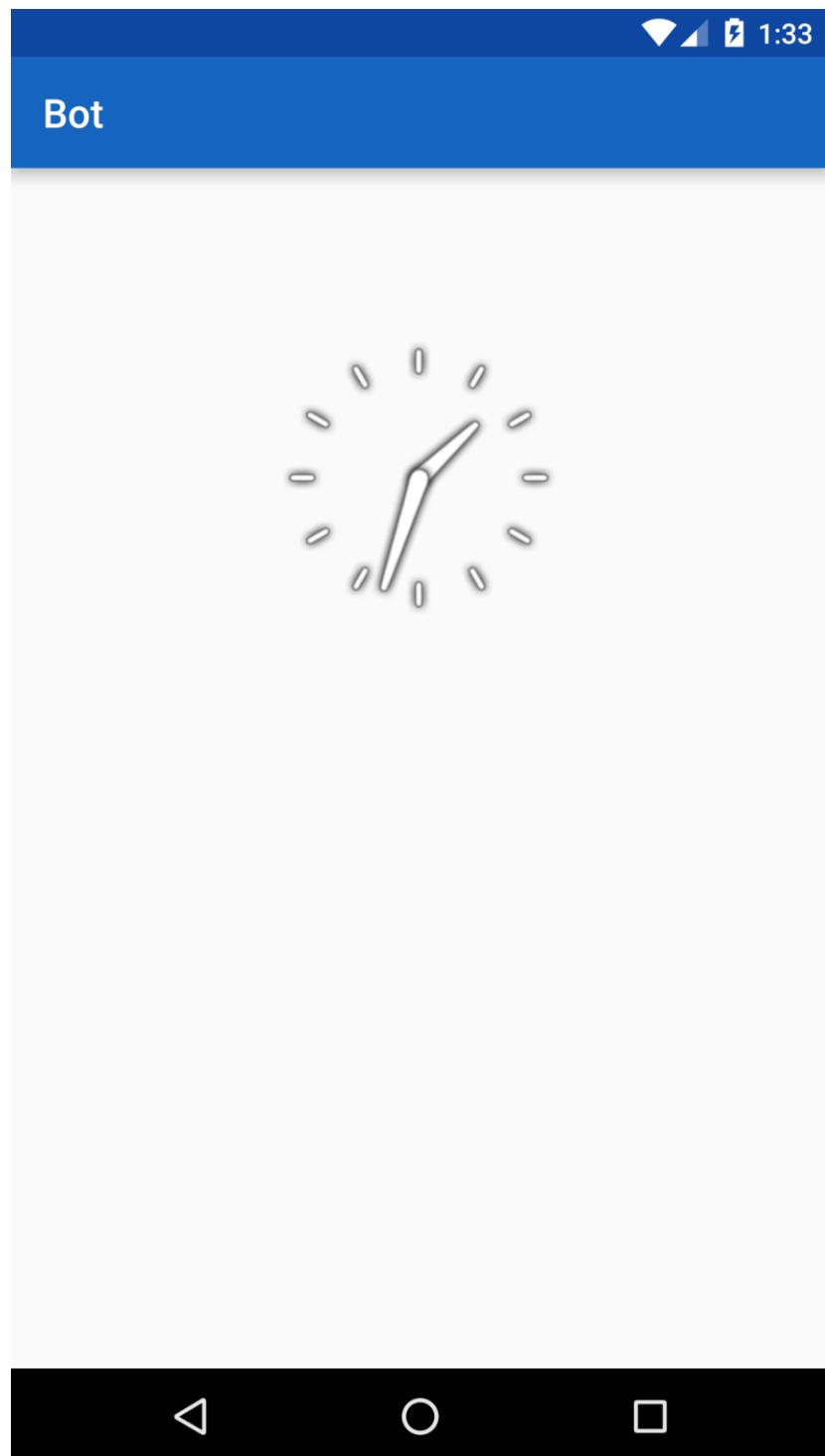


```
Output X
CnC (Server) (run) x BotMaster_Android (run) x Bot (Client) (run) x
run:
botnet.botmasterandroidclient.Bot_Properties@19966899
C&C Server> Hi bot plz send me your informations.
C&C Server> Welcome Bot !!!
```

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.



```
Output X  
CnC (Server) (run) x BotMaster_Android (run) x Bot (Client) (run) x DTLS_Server (run) x Client_DTLS (run) x  
  
run:  
Server started  
Cookie: javax.servlet.http.Cookie@279fc6f9  
SendMsg: java.net DatagramPacket@37b33194  
w8ing for cookie  
Cookie from Client: DTLSserverCookie_1 169.254.80.80  
HandShake Method:  
SendMsg: java.net DatagramPacket@2b2fb14e  
Sec Key: javax.crypto.spec.SecretKeySpec@ffffe8669  
HMAC is rdy for use.  
Chat is rdy.  
first: [B@1e5358d2  
SendMsg: java.net DatagramPacket@38613434
```

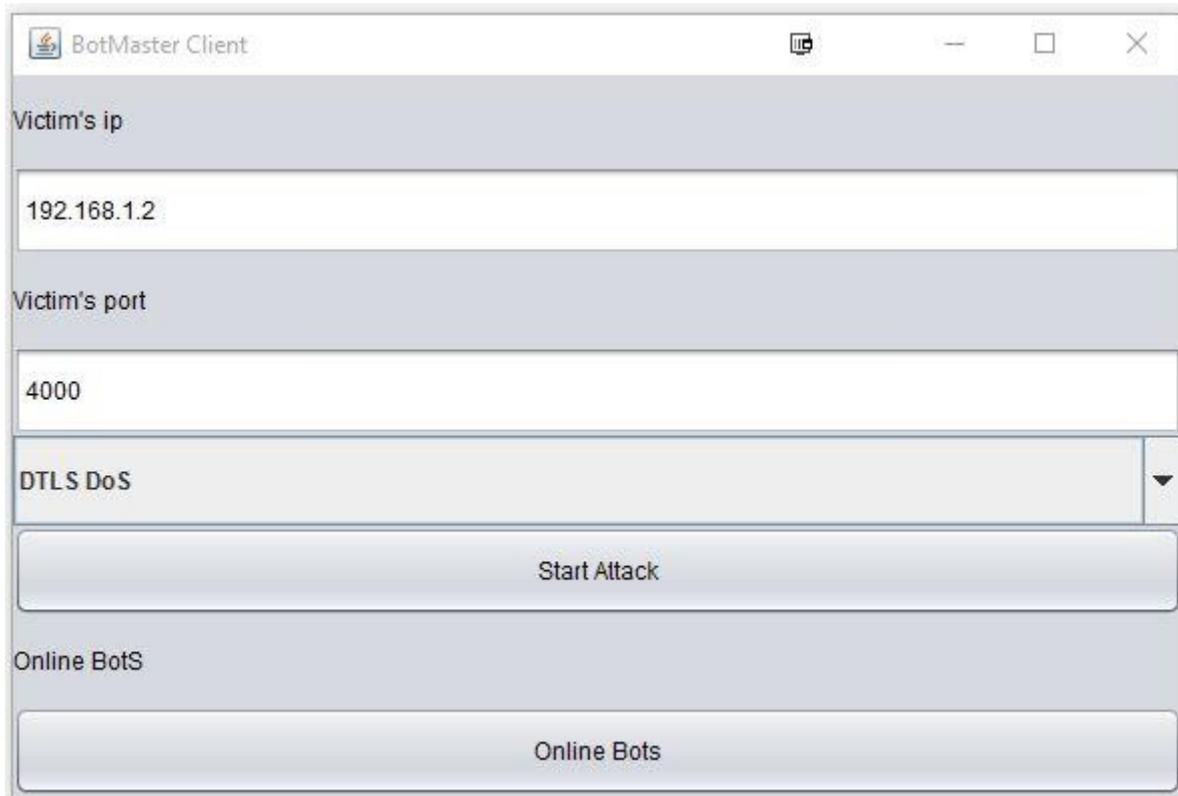
```

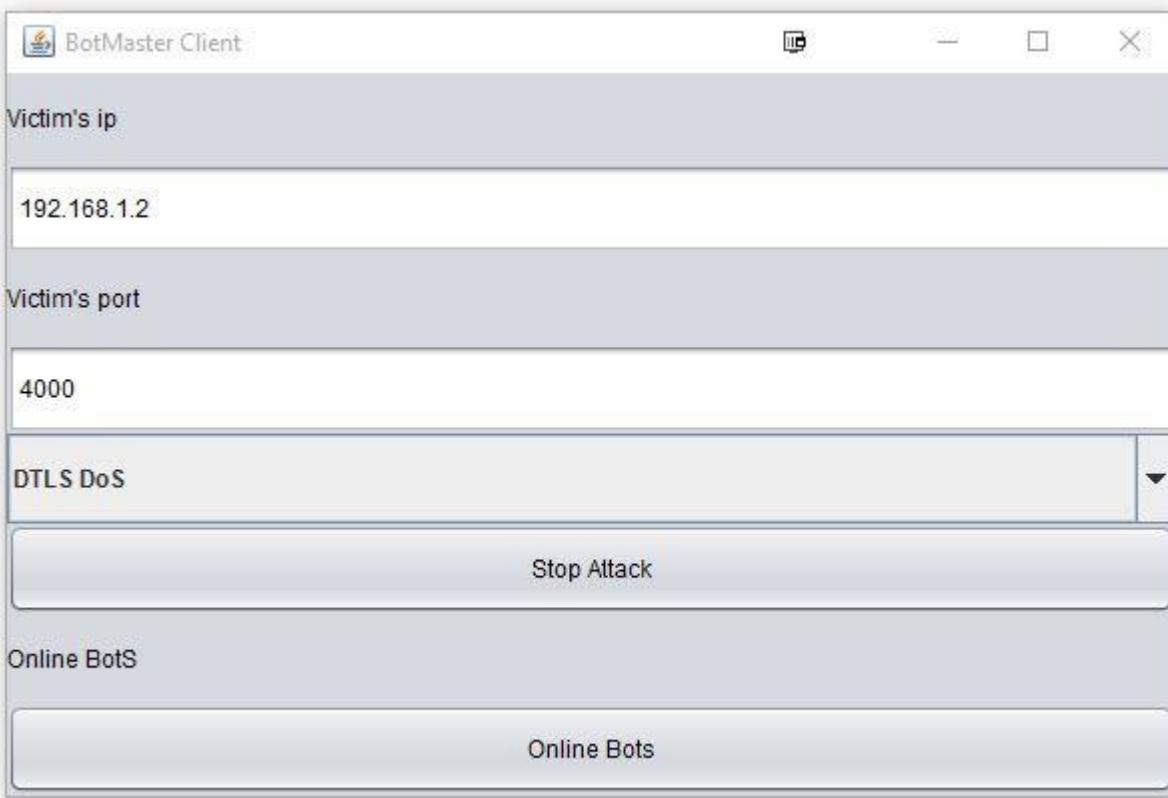
Output X
CnC (Server) (run) x BotMaster_Anyroid (run) x Bot (Client) (run) x DTLS_Server (run) x Client_DTLS (run) x

run:
SendMsgs: java.net DatagramPacket@eed1f14
169.254.80.80
Cookie: DTLS ServerCookie_1
SendMsgs: java.net DatagramPacket@58372a00
000000
000000
0000000000
`000X000MFOfOf(000o\0000,10_u0000u0e0z>2000n
0P0) $00+0! Y0?0A0000;W00y-a000"0`0(j000000000dQK0?0'0000000;0000(<0M00080000Qx,&~N0qh<0~k0Im#0e0Y000y#`00'00010
000h^
R0{0b00|0bKn0Hj$0020A,0zQ04Y000Q0'000`)(0Z0d40^00000_00
!`00T000>,00D0T000w0W00 Of00?0)pEa$00_)0m0_000)i!00 00 M0vBZU000R00z00'0000Z0h0 q0000000703K000w0p000000PE00000+C
++This certificate is VALID++
Sec Key: javax.crypto.spec.SecretKeySpec@ffffe8669
SendMsgs: java.net DatagramPacket@3ecf72fd
handshake complete. HMAC is ready.
chat is rdy.
ReadByte: 000ur0[B0000T00xp 0300h0000$00K0000xwc00A00{000!F
Decrypted msg: hello client

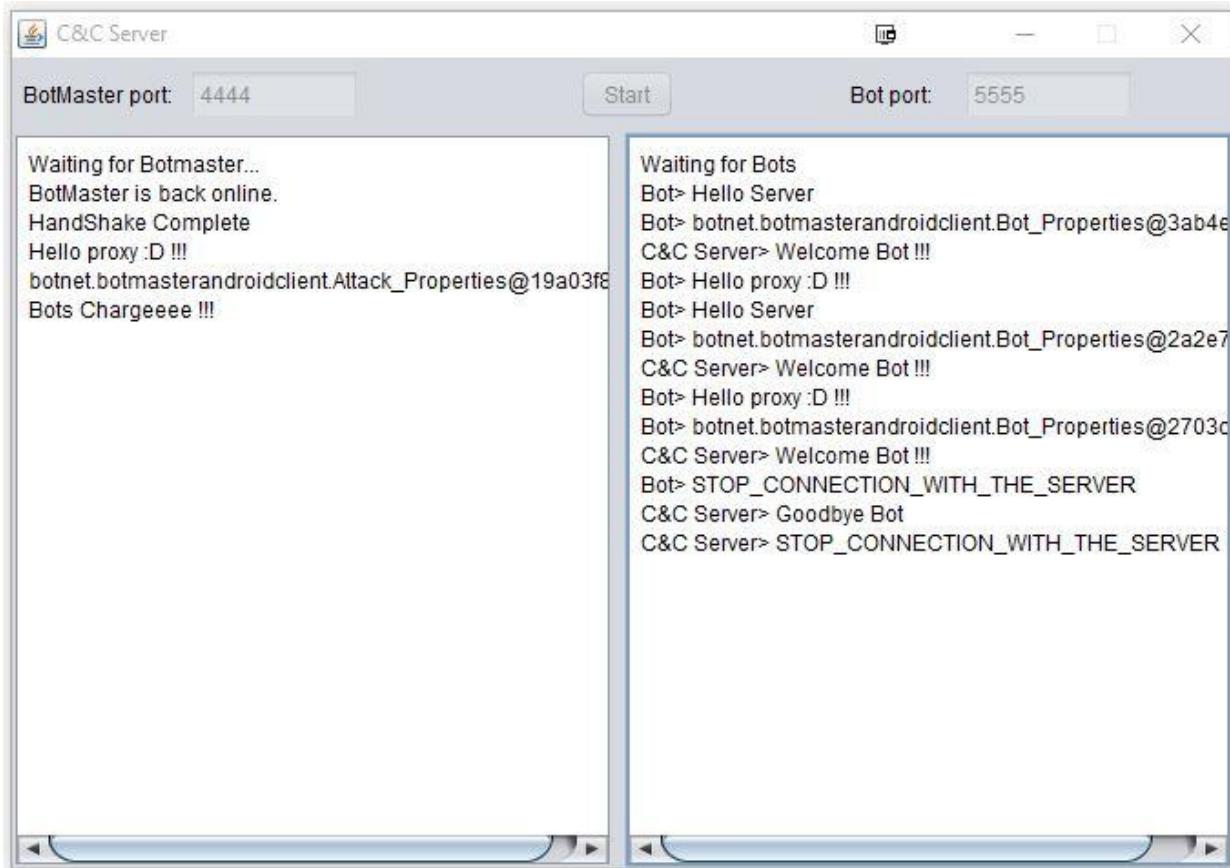
```

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





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



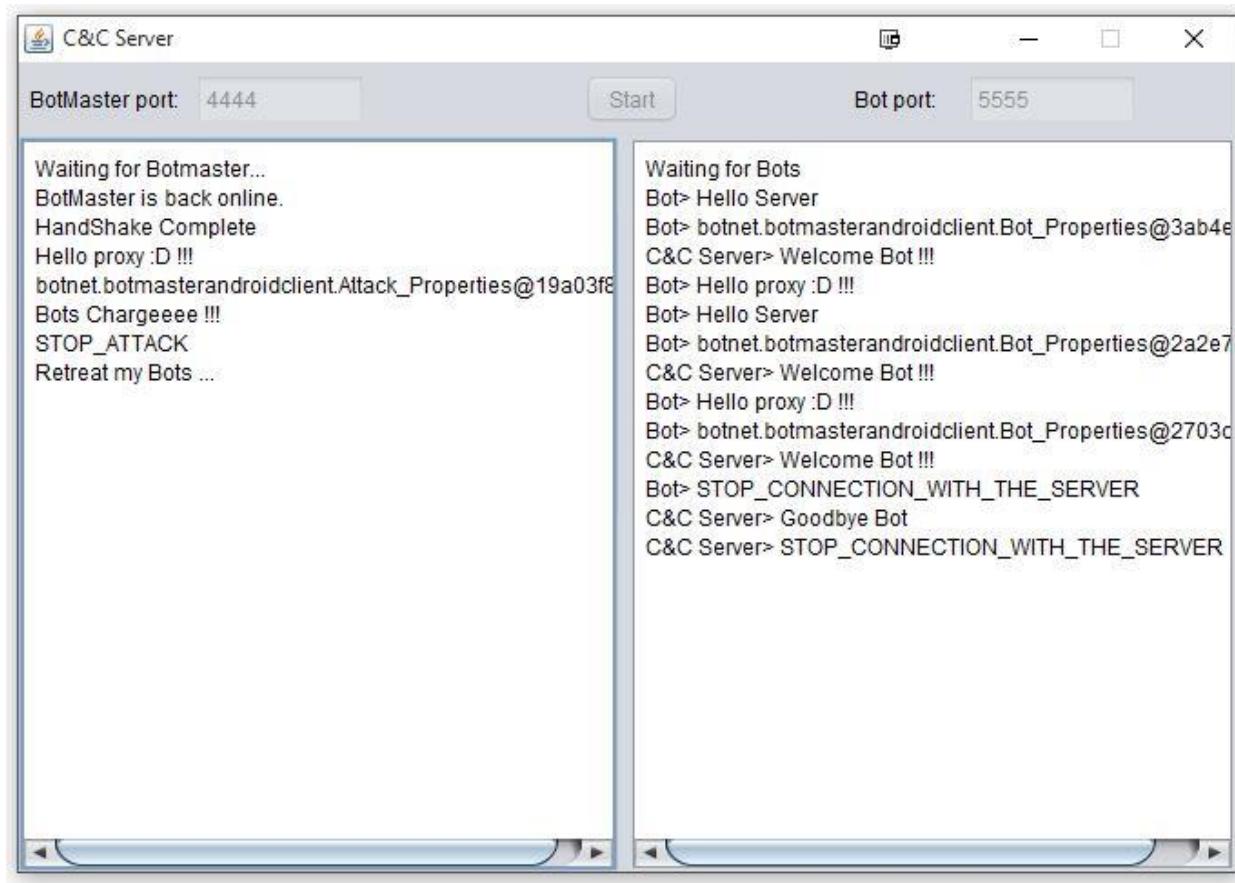
Log.d("Debug:", "123");

Android Monitor

Genymotion Google Nexus 6P - 6.0.0 - API 23 - 1440x2560 Android 6.0 (API 23) | botnet.botmasterandroidclient (1728)

logcat Memory CPU GPU Network pose Regex Show only selected application

```
04-17 15:01:51.539 1728-1741/? D/2:: javax.net.ssl.TrustManagerFactory@9bfe448
04-17 15:01:51.547 1728-1742/? D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
04-17 15:01:51.563 1728-1728/? D/: HostConnection::get() New Host Connection established 0xab142320, tid 1728
04-17 15:01:51.603 1728-1741/? D/Debug: 123
04-17 15:01:51.684 1728-1742/? D/libEGL: loaded /system/lib/egl/libEGL_emulation.so
04-17 15:01:51.685 1728-1742/? D/libEGL: loaded /system/lib/egl/libGLESv1_CM_emulation.so
04-17 15:01:51.692 1728-1742/? D/libEGL: loaded /system/lib/egl/libGLESv2_emulation.so
04-17 15:01:51.703 1728-1742/? D/: HostConnection::get() New Host Connection established 0xab1424d0, tid 1742
04-17 15:01:51.878 1728-1742/? I/OpenGLRenderer: Initialized EGL, version 1.4
04-17 15:01:52.150 1728-1741/? D/msg from server:: Hi bot plz send me your informations.
04-17 15:01:52.163 1728-1741/botnet.botmasterandroidclient D/IP:: 127.0.0.1
04-17 15:01:52.163 1728-1741/botnet.botmasterandroidclient D/name:: Android: localhost
04-17 15:01:52.189 1728-1746/botnet.botmasterandroidclient I/System.out: botnet.botmasterandroidclient.Bot_Properties
04-17 15:01:52.189 1728-1746/botnet.botmasterandroidclient I/System.out: Hello proxy :D !!!
04-17 15:01:52.205 1728-1742/botnet.botmasterandroidclient W/EGL_emulation: eglSurfaceAttrib not implemented
04-17 15:01:52.205 1728-1742/botnet.botmasterandroidclient W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surf
04-17 15:01:52.431 1728-1746/botnet.botmasterandroidclient I/System.out: C&C Server> Welcome Bot !!!
04-17 15:01:52.431 1728-1746/botnet.botmasterandroidclient D/Readobj:: Welcome Bot !!!
04-17 15:02:05.383 1728-1746/botnet.botmasterandroidclient I/System.out: C&C Server> botnet.botmasterandroidclient.A
04-17 15:02:05.383 1728-1746/botnet.botmasterandroidclient D/Readobj:: botnet.botmasterandroidclient.Attack_Properti
04-17 15:02:05.383 1728-1746/botnet.botmasterandroidclient I/System.out: Target IP: 192.168.1.2 Port:4000
04-17 15:02:12.171 1728-1738/botnet.botmasterandroidclient I/art: Background sticky concurrent mark sweep GC freed 2
```

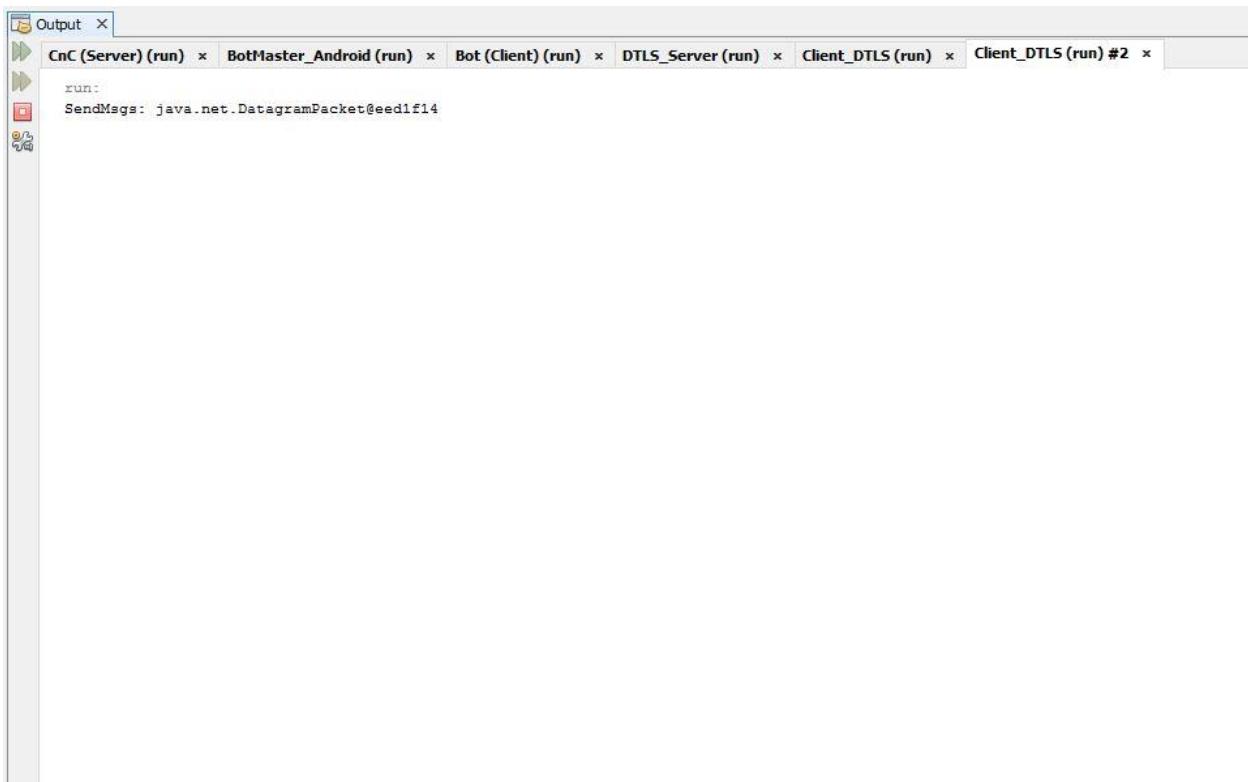


Στα logs του DTLS Server φαίνονται να δημιουργούνται συνέχεια cookies προς τους clients (bots) χωρίς αυτά να το στέλνουν πίσω (επιτυχής DDoS επίθεση).

```
run:  
Server started  
Cookie: javax.servlet.http.Cookie@279fc6f9  
SendMsg: java.net DatagramPacket@37b33194  
w8ing for cookie  
Cookie from Client: DTLS ServerCookie_1 169.254.80.80  
HandShake Method:  
SendMsg: java.net DatagramPacket@2b2fb14e  
Sec Key: javax.crypto.spec.SecretKeySpec@ffffe8669  
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_Anyroid (run) x Bot (Client) (run) x DTLS_Server (run) x Client_DTLS (run) x Client_DTLS (run) #2 x
Cookie: javax.servlet.http.Cookie@2u00c701
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).



The screenshot shows the 'Output' tab of an IDE interface. The tab bar at the top includes 'Output' (selected), 'CnC (Server) (run)', 'BotMaster_Android (run)', 'Bot (Client) (run)', 'DTLS_Server (run)', 'Client_DTLS (run)', and 'Client_DTLS (run) #2'. The main pane displays a single line of text: 'SendMsgs: java.net DatagramPacket@eed1f14'. On the far left, there is a vertical toolbar with icons for file operations like Open, Save, and Print.

```
SendMsgs: java.net DatagramPacket@eed1f14
```

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