PROJECT ANALYSIS

06 April 2014

Team C

Jamie Lane, Bradley Norman, Daniel Ross

Project Purpose

This application will generate the type of traffic but not the volume of traffic necessary to initiate an RDoS. It is not meant to be an actual hacking tool, but a proof of concept. Without a proof-of-concept exploit application available, many developers will not patch security vulnerabilities. The vulnerability that this app exploits was patched in 2010.

Project Analysis

A. Identify all outside systems with which this system interfaces.

User, IP Network (LAN, WAN), Open Arena Server

B. Identify all input data and the source(s) of these input data.

User → source IP address, destination IP address, destination port, build directive, transmit directive
Open Arena Server → status response packet

C. Identify all output data and the destinations of these data.

Status request packet → Open Arena Server
Packet size ratio of sent/received packets → User

D. Identify the data processing function of this system:

This system receives source IP address, destination IP address, and destination port input from the user via a GUI. Then it constructs UDP packets and IP packet headers; combines IP packet headers and UDP packet payloads; calculates complete packet size; transmits packets to the Open Arena server at the destination address upon user initiation; receives packets from the Open Arena server; calculates size of received packets; and calculates the ratio of the sent/received packet sizes to be output to the GUI. Figure 1 contains a System Context Diagram depicting data flowing in and out of the system under analysis.

E. Based on the data processing steps, break the system into sub-systems, each dedicated to a single task (specify the task that each subsystem does). These subsystems together show the data processing function that converts the input data into output data.

User – determines the source and destination IP addresses and the destination port
Input Subsystem (GUI) – records the source and destination IP addresses and the destination port
Builder (GUI) – starts packet constructors and combiner, stores resulting packet
UDP Packet Constructor – constructs UDP packet

IP Header Constructor – constructs the IP packet header
IP/UDP Combiner – combines IP header with UDP packet to generate Status Request packet
Packet Size Calculator – calculates the total size of a packet
Outbound – interfaces with Host OS to allow packet transmission
Transmitter (GUI) – sends status request packet to Outbound
Inbound – interfaces with Host OS to allow packet receipt
Receiver – receives Status Response packet from Inbound, stores it
Ratio Calculator – calculates packet size ratio of sent/received packets
Display Subsystem (GUI)

Figure 2, Subsystem Diagram, illustrates the data flow from subsystem to subsystem. Additionally, it illustrates the transfer of data into and out of the system under analysis.

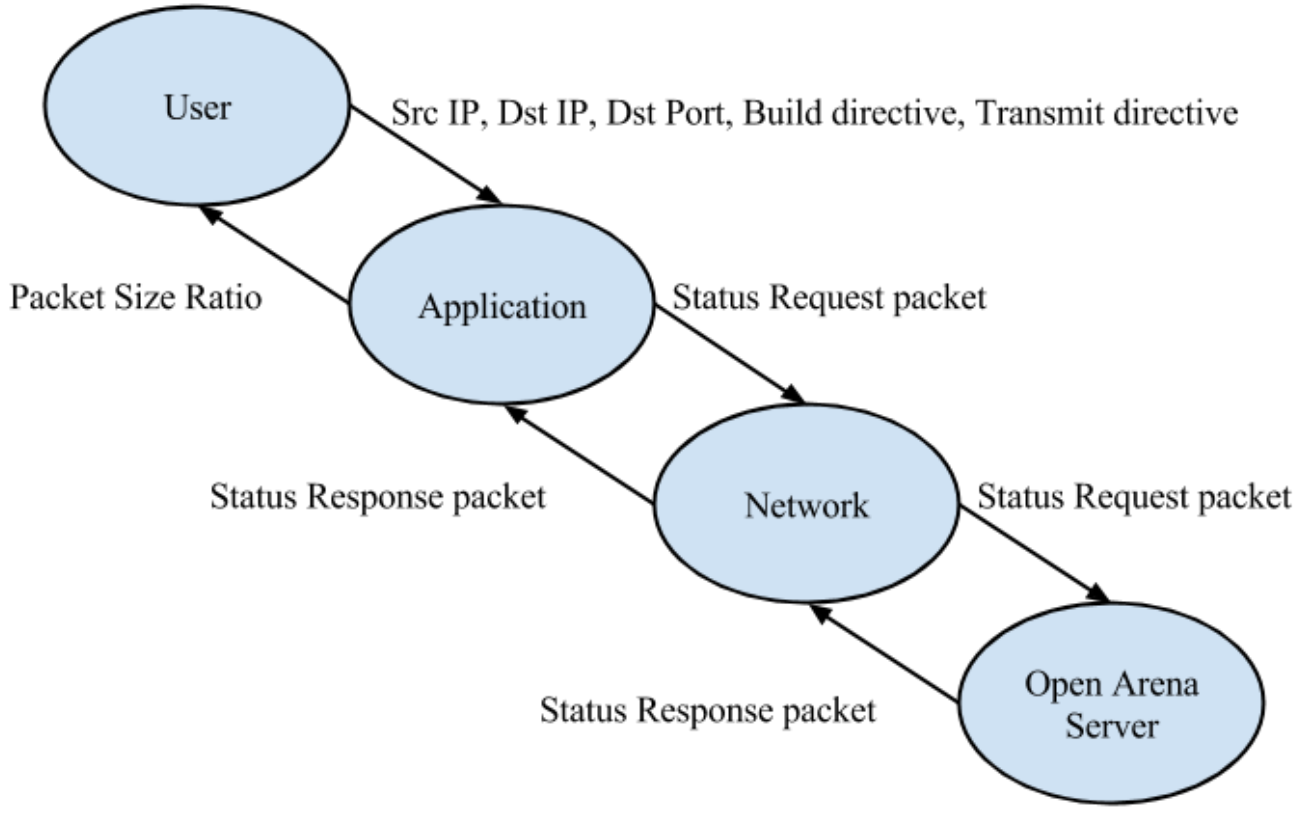

Figure 1. System Context Diagram. Each oval represents a system. Each arrow represents a data pathway.

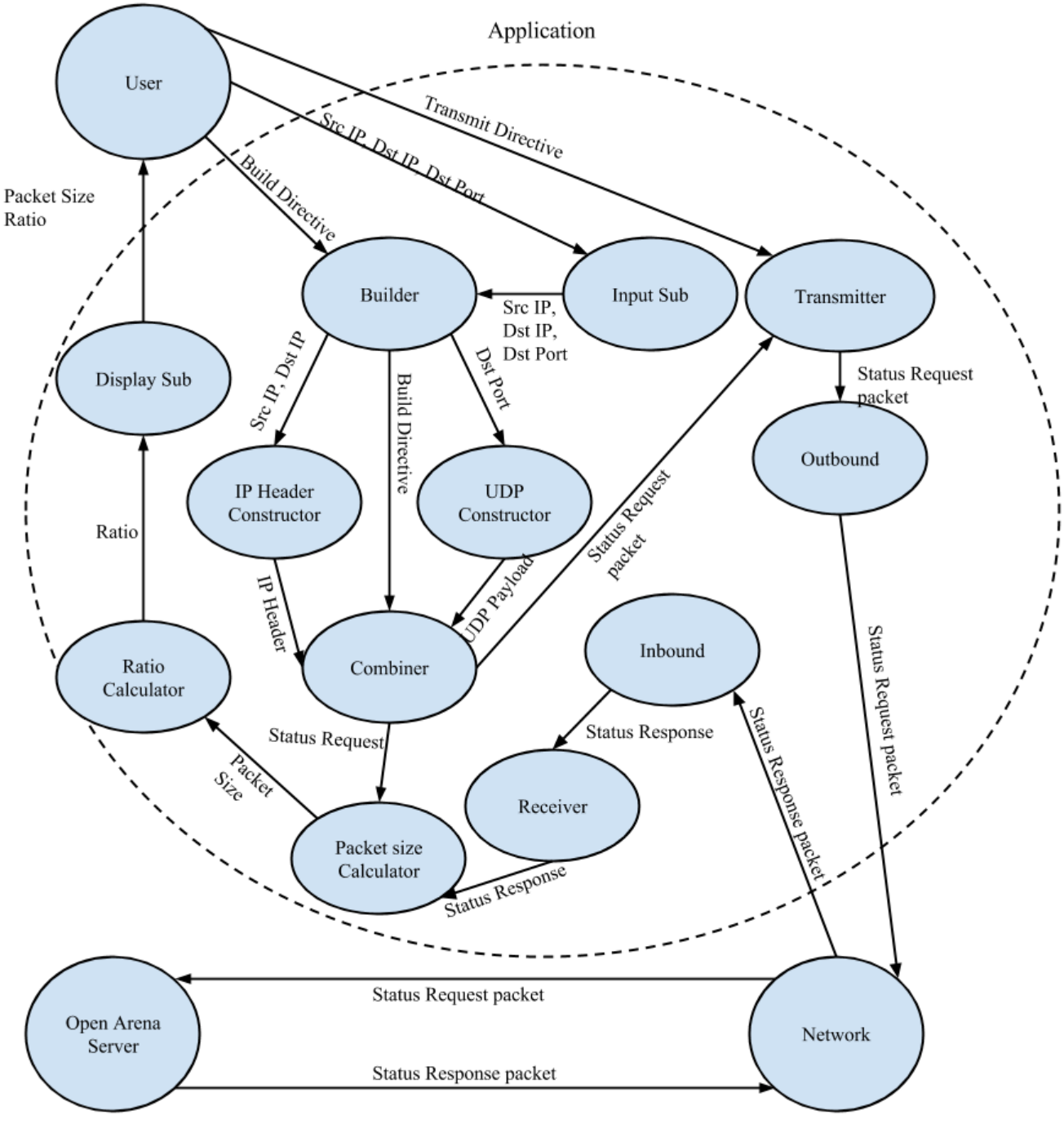

Figure 2. Subsystem Diagram. Each oval represents a system. Each arrow represents a data pathway.

F. Identify interface data between each subsystem (and which subsystem processes the inputs, which subsystem does the output).

User → Input Subsystem
User → Builder
User → Transmitter
Input Subsystem → Builder
Builder → UDP Packet Constructor, IP Header Constructor, IP/UDP Combiner
IP/UDP Combiner → Transmitter, Packet Size Calculator
Transmitter → Outbound
Inbound → Receiver
Receiver → Packet Size Calculator
Packet Size Calculator → Ratio Calculator
Ratio Calculator → Display Subsystem
Display Subsystem → User

G. Check the solution against the requirements.

Table 1, Requirements Matchup illustrates the correlation between each requirement and its implementing subsystem.

Table 1. Requirements Matchup.

Correlates each subsystem to its related requirement.

Table 1	
Requirements Matchup	
Requirement	Description
1	nonfunctional requirement
2	nonfunctional requirement
3	Input Subsystem (GUI)
4	nonfunctional requirement
5	Input Subsystem (GUI)

6	Input Subsystem (GUI)
7	Input Subsystem (GUI)
8	UDP Packet Constructor
9	IP Header Constructor
10	Builder (GUI), IP/UDP Combiner
11	Packet Size Calculator
12	Transmitter (GUI), Outbound
13	Inbound, Receiver
14	Packet Size Calculator
15	Ratio Calculator
16	Display Subsystem (GUI)
17	nonfunctional requirement
18	nonfunctional requirement
19	nonfunctional requirement

H. Identify risks in your design and possible ways for mitigating these risks .

A risk associated with testing or using this application is the possibility of causing a Denial-Of-Service "attack" on the host machine when using the host IP address as the source IP address. This risk will be mitigated by the application sending a single packet per user click. It is highly unlikely that a user would inadvertently generate enough clicks to compromise their own system or network.

An additional testing risk, would be the possibility of impacting the performance of an Open Arena server without the administrator's permission. This risk will be mitigated by only testing with a server hosted by a member of the development team.

The application will be designed to provide proof of a vulnerability, while minimizing the risk of it becoming an actual malware or hacking tool. One mitigating factor is that the patches for the vulnerability the application exploits became available in 2010. The remaining mitigating techniques

will be applied during the application design phase. It will not be designed to be operated remotely. Its design will not include operation by a timer. The interface will not have hidden functionality. The code will not include deliberately opaque or obfuscated functionality.

I. Identify possible enhancements (new features) to your design; this is a way to get future work.

Add the ability to add multiple destination IP addresses to demonstrate a Distributed Denial-of-Service attack.

Add the ability to exploit additional Open Arena server vulnerabilities.

Add the ability for the user to specify a custom UDP payload. With small modifications, the application could be used to demonstrate similar vulnerabilities in other servers.