Testing Intrusion Detection Systems and Firewalls:

A Holistic Approach

Introduction

- My name is Matt, my handle is syke.
- Member of NewHackCity, a hacker collective based in San Francisco
- Worked for 2 years at a vendor of “security” software
- Selling out: t-shirts are available from website
- Check out jeru’s paper on IDS Evasion

Rundown

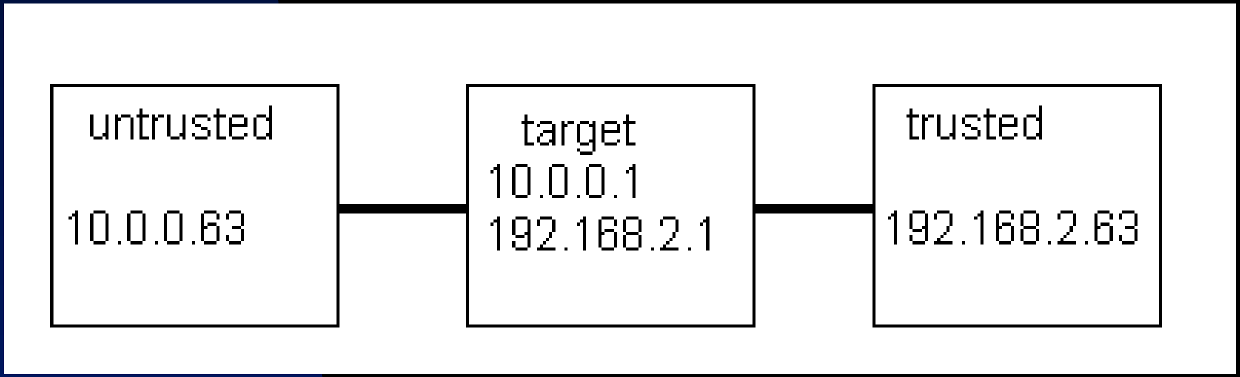
- User Expectations
- Testing for those Expectations
- Common coding mistakes testing reveals
- What designers can do to avoid this nonsense

Users: Expectations

- Users don’t care about proxy vs. packet filter or network-based vs. host-based
- Most users are aware IDSes don’t reduce risk, only exposure
- Firewalls actually can limit risk
- But, you should have both anyways

Testers: Summary

- nmap, whisker, isic
- Don’t just use these tools blindly, understand them
- Configuration can make a difference
- Test from trusted to untrusted and vice versa
- When reporting bugs, be detailed



Testers: nmap

- Good for testing state table code in packet filter, NAT, and IPsec implementations
- From untrusted “nmap –sS –p1-65535 trusted”
- Try different scan types (FIN, ACK, etc)
- Tiny IP frags (nmap –f)
- “`nmap –sS –p 1-65535 –D 192.168.2.0,192.168.2.255,10.0.0.0,10.0.0.255,ME,255.255.255.255,0.0.0.0 trusted`”
- http://insecure.org/nmap

Testers: whisker

- whisker is already widely known as a great tool to test IDS' susceptibility to evasion
- "`./whisker –h target –I 1`"
- whisker is also good for testing HTTP proxies that block based on URL content
- http://www.wiretrip.net/rfp

Testers: isic

- isic is a suite of utilities: isic, tcpsic, udpsic, icmpsic, esic
- great for testing protocol implementations, IDSes, packet filters, IPsec tunnels, NAT
- "`tcpsic –s rand –d trusted –r 31337 –m 600`"
- Test IKE: "`udpsic –s rand –d trusted,500 –r 31337 –m 700`"
- Linux can send packets faster, but mangles them; BSD is slower
- http://expert.cc.purdue.edu/~frantzen

Testers: isic (cont’d)

- Real world example: icmpsic testing against NetBSD
- “`icmpsic –s attacking_ip –d victim_ip –r 31337 –m 600`”
- Caused a crash around packet 225,000
- Binary search and destroy: “`icmpsic –s attacking_ip –d victim_ip –r 31337 –m 600 –k 112500 –p 225000`”

Coders: Summary

- define your interfaces and data structures before you start coding, maintain consistency
- writing clean, well documented code will save time and aneurysms later
- compilers give warnings for a reason
- source and runtime analysis tools can help
  - `PC-Lint: http://www.gimpel.com`
  - `Purify: http://www.rational.com`
  - `Insure++: http://www.parasoft.com`
  - `BoundsChecker: http://www.numega.com`

Coders: Example

```
void icmp_filter(struct icmp_header_struct *icmp_header, int
   icmp_header_length, int ICMP_TYPE)
{
   struct icmp_header *icmp_header_munged;
   icmp_header_munged = kmalloc( sizeof (*icmp_header) );
   memcpy(*icmp_header, *icmp_header_munged, sizeof (*icmp_header) );
   switch (ICMP_TYPE)
   {
   case ICMP_ECHO_REQUEST:
   kfree(icmp_header_munged);
   return;
   case ICMP_REDIRECT:
   kfree(icmp_header_munged);
   break;
   case ICMP_REPLY:
   kfree(icmp_header_munged);
   }
}
```

Coders: Example (cont’d)

- declared void
- pointer from malloc is used without checking for NULL
- flow from switch flows into program
- switch has no default
- because switch has no default, condition exists where malloc'd memory is never free'd
- function ends without a return
- No comments

Designers

- host-based IDSes can use the TDI interfaces on Windows (9x/NT/2k), or the netfilter interfaces on Linux 2.4 kernel for post-reassembly detection
- NDIS (or NIC driver integration) can get low level things
- don’t re-implement the world, especially when it comes to protocol decoding
- UML can help with maintaining consistent interfaces between design and development

Q&A

- http://www.newhackcity.net
- mailto:syke@newhackcity.net