Techniques

Network Defense Applications using IP Sinkholes

Victor Oppleman

Difficulty

A little-talked-about network security technique has proven one of the most effective means of defense against Denial-of-Service attacks and a successful means of threat data collection. In this article we will explore advanced network defense applications using stationary and event-driven IP sinkholes.

It has been deployed by Internet service providers globally as a way to protect their downstream customers. As this article will explain, the technique, known as sinkholing, may also be used to provide valuable intelligence regarding the threats your network is facing. By implementing sinkholes, you'll gain yet another means of defending your network and gleaning valuable information regarding both threats and significant misconfigurations throughout your network.

Meant for network-savvy users, this article will provide the following:

- Sinkhole Background and Function – A brief explanation of IP sinkholes and how a number of organizations have successfully implemented them,
- Decoy Network Deployments – How sinkhole techniques applied using darknets and honeynets may be used to trap and analyze malicious scanning, infiltration attempts, and other events in conjunction with your network monitoring elements such as intrusion detection,
- Denial-of-Service Protection – How organizations and their upstream Internet service providers have developed a means of protection against denial-of-service through extensive, event-driven sinkhole deployments,
- Backscatter and Tracebacks – a brief explanation of backscatter and how tracebacks can be used to identify the ingress point of a Denial-of-Service attack in a large network.

Background and Function

In this text, the term sinkhole may be defined as a generalized means of redirecting specific IP network traffic for different security-related

What you will learn...

- you will learn how to use sinkholing techniques and how to protect from Denial-of-Service attacks.

What you should know...

- you should have basic knowlegde about Denial -of-Service attacks,
- you should know the network traffic issues on ISP side.

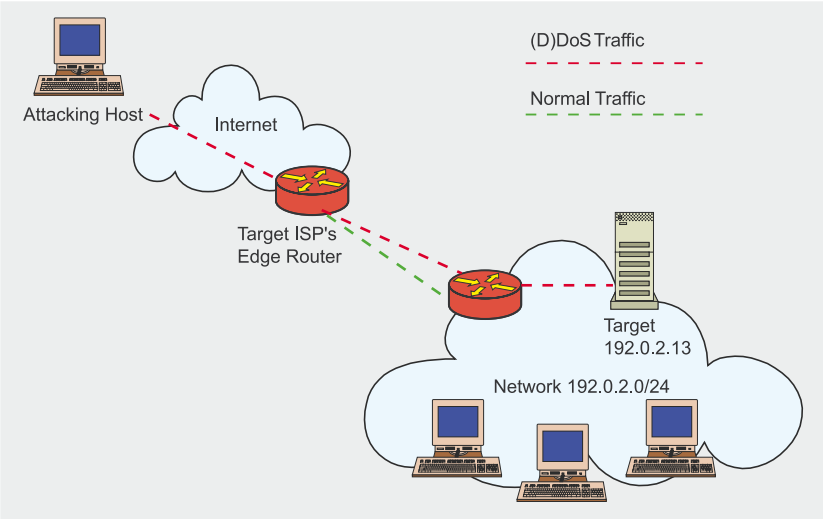

Figure 1. *An attack on IP address 192.0.2.13 (before sinkholing)*

purposes including analysis and forensics, diversion of attacks, and detection of anomalous activities. Tier-1 ISPs were the first to implement these tactics, usually to protect their downstream customers. Since then, the techniques have been adapted to collect interesting threat-related information for security analysis purposes. To visualize the simplest form of a sinkhole, consider the following:

Malicious, disruptive traffic sourced from various networks is destined for network 192.0.2.13, as shown in Figure 1. The organization being targeted by this traffic utilizes 192.0.2.0/24 as its network address block that is routed by its upstream ISP. The attack becomes debilitating, disrupting business operations of the target organization and potentially increasing its costs because of increasing bandwidth utilization, and necessitating action by the ISP because the overwhelming amount of traffic generated by the attack is disrupting adjacent customers as a form of collateral damage.

The ISP reacts and temporarily initiates a blackhole-type sinkhole by injecting a more specific route for the target (192.0.2.13/32) inside their backbone, whose next-hop is the discard interface on their edge router (also known as *null0* or the *bit bucket*), as shown in Figure 2.

This tactic redirects the offensive traffic toward the ISP's sinkhole instead of allowing it to flow downstream to the original target. The benefit is that from the time the sinkhole goes into effect, the adjacent ISP customers are likely (as long as the ISP thoughtfully designed their sinkhole defenses) free of collateral damage and the target of the attack has regained use of their Internet connection and local access to the specifically targeted device. Unfortunately, the specific IP address (device) being attacked cannot converse with remote systems across the Internet until the sinkhole is removed (presumably after the attack has subsided). Obviously, the services originally provided by the target device may be migrated to an alternative device at a different IP address, but many other considerations would have to be made in terms of DNS TTL expiry, and so on.

This example is merely one type of sinkhole, normally referred to as an ISP-induced blackhole route, but this should familiarize you with the concept so that we can explain various other uses of sinkholes.

Using Sinkholes to Deploy Decoy Networks

A more novel use of sinkholes is in the deployment of various kinds of decoy networks for entrapment, exposure, and intelligence-gathering purposes.

Decoy \De*coy"\, *noun*, anything intended to lead into a snare; a lure that deceives and misleads into danger, or into the power of an enemy; a bait.

The two types of decoy networks we'll discuss in detail are the darknet and the honeynet. Both may be used to glean security intelligence, but one is particularly useful in the realm of secure network engineering.

Deploying Darknets

Generally, a darknet is a portion of routed, allocated IP space in which no responsive services reside. Such networks are classified as *dark* because there is seemingly nothing *lit up* inside these networks. However,

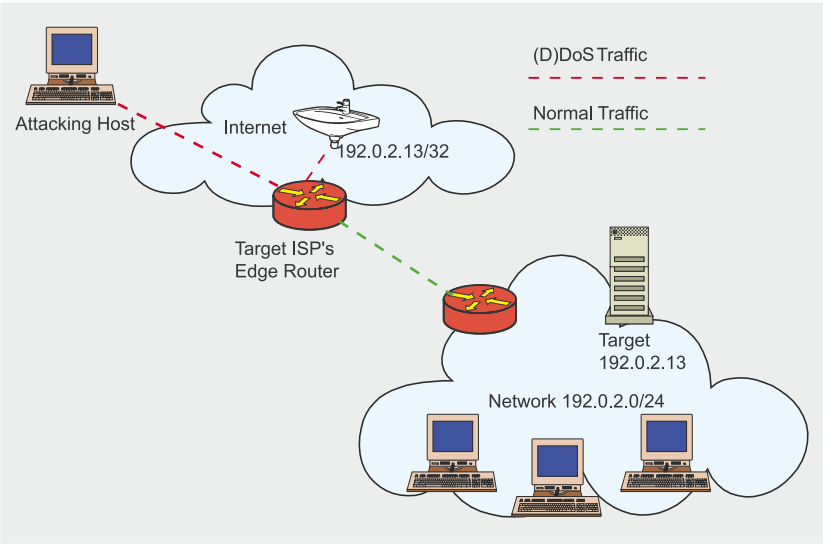

Figure 2. *An attack on IP address 192.0.2.13 (while sinkholing)*

Listing1. *BGB sample configuration*

```
router bgp XXX
redistribute static route-map static-to-bgp
# Route-map is a policy mechanism to
# allow modification of prefix attributes, or special
# filtering policies
route-map static-to-bgp permit 10
match tag 199
set ip next-hop 192.0.2.1
set local-preference 50
set community no-export
set origin igp
```

Listing 2. *The basic configuration on the ISP side*

```
router bgp XXX
# Route-map is simply a policy mechanism
# to massage routing information such
# as setting the next hop
neighbor < customer-ip > route-map customer-in in
# prefix-list is a static list of customer prefixes and mask length that
# are allowed.  Customer should be allowed to
# announce down to a single host
# in their prefix(es) such as 172.16.0.1/32
neighbor < customer-ip > prefix-list 10 in
# ebgp-multihop is necessary to prevent
# continuous prefix announcement and
# withdrawal
neighbor < customer-ip > ebgp-multihop 2
# Now we define the route-map for policy match
# and setting the blackhole
# next hop
route-map in-customer permit 5
# the customer sets this community on their side,
# and the ISP matches on its
# side.  XXXX  would likely be the customer ASN,
# and NNNN is an arbitrary number agreed
# on by the ISP and the customer
match ip community XXXX:NNNN
set ip next-hop < blackhole-ip>
set community additive no-export
```

a darknet does in fact include at least one server, designed to act as a packet vacuum. This server gathers and organizes the packets that enter the darknet, useful for real-time analysis or post-event network forensics.

Any packet that enters a darknet is unexpected. Because no legitimate packets should ever appear inside a darknet, those that do appear have either arrived by misconfiguration or by the more frequent scenario, having been sent by malware. This malware, scanning for vulnerable devices, will send packets into the darknet, thereby exposing itself to administrative security review. There is a slant of genius in this approach for finding worms and other propagating malware. Without false positives, and without signatures or complicated statistical analysis gear, a security administrator with properly deployed darknets can spot scanning (attempts made by malware to discover adjacent hosts suitable for propagation) in any size network. That's a powerful security tool. Further, packets arriving in the darknet expose innocuous network misconfigurations that network administrators will appreciate ironing out. Of course, darknets have multiple uses in the realm of security. They can be used to host flow collectors, backscatter detectors, packet sniffers, and intrusion detection systems. The elegance of the darknet is that it cuts down considerably on the false positives for any device or technology through simple traffic reduction.

Implementing a darknet is relatively simple. In fact, here are five easy steps.

Select one or more unused regions of IP address space from your network that you'll route into your darknet. This could be a /16 prefix of addresses or larger, or all the way down to a single (/32) address. More addresses result in a more statistically accurate perception of unsolicited network activity. I recommend selecting several address segments, such as a /29 from each of several internal networks, and a /25 from your public (external) network allocation, for example. There's no reason you can't darknet a region of your internal private address space (for example, RFC 1918 space, 10.0.0.0/8). In fact, by selecting regions of your internal network to darknet, you'll be able to see internal scanning that you may miss if you only darknet external (public) network segments. Another strategy that can be considered by organizations utilizing specific routing for their internal networks is to rely upon the *most specific route wins* rule of routing (usually distributed through some kind of interior gateway protocol). Meaning, if I use the 10.1.1.0/24 and the 10.2.1.0/24 networks internally, I can just route the entire 10.0.0.0/8 network into my darknet. I know that if my network is properly configured, the darknet will receive all 10.0.0.0/8 traffic except for the networks within it that I'm specifically using/routing (these likely have static routing entries in my network infrastructure).

Next, you'll configure your physical topology. You'll need a router or (layer-3) switch that will forward traffic into the your darknet, a server with ample storage to serve as your data collector, and an Ethernet switch you'll use to connect

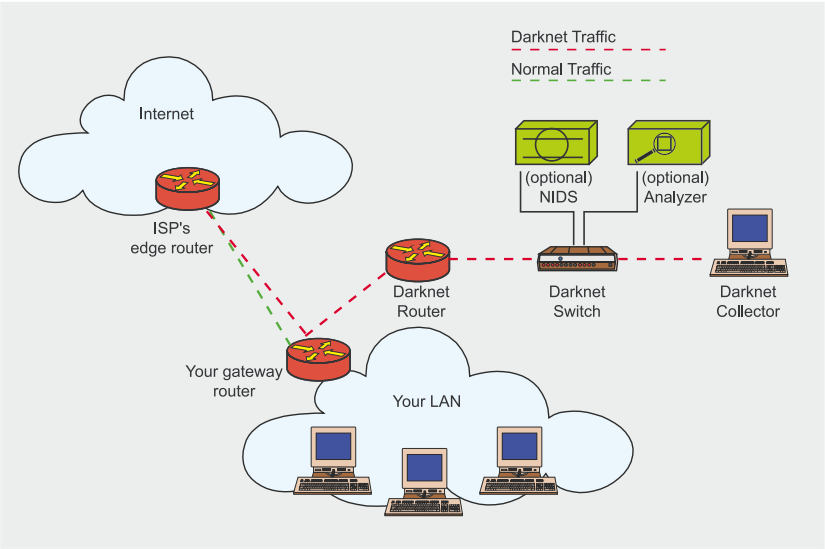

Figure 3. *A reference physical topology for darknets*

these components and optional components in the future such as an IDS sensor or protocol analyzer. For the router, you may elect to use an existing internal or external (or both, though it is not recommended) gateway device – most *enterprise* darknets (as opposed to those of telecom carriers) are located inside one of the organization's DMZs and segregated from the rest of the network. Therefore, you may consider using a firewall to do this job in lieu of one of your routers. We recommend, however, that you use your external gateway router for external darknets, and an internal layer-3 switch for your internal darknets. Either way, the key item to consider is that you'll configure this routing device to forward the darknet-destined traffic it receives out of a dedicated darknet ethernet interface (through the switch) to the collector server that you'll configure to accept such packets. The collector server must also have a dedicated darknet interface that will receive those packets. For management, the collector server will also require at least one additional Ethernet interface (to be placed on a separate management LAN). Make sure you follow your own best practices for network device security as you can be guaranteed that all sorts of nasties will be flowing through this network segment very soon. Fight the urge to quickly utilize an existing DMZ switch for the purpose of connecting these components unless you're comfortable configuring the VLAN so that no broadcast packets will make their way into the darknet – remember, the darknet is for illegitimate traffic only so we don't want legitimate broadcasts from your other LANs encroaching on darknet turf. Figure 3 depicts an example of this configuration. In our examples, we're using a router or switch running Cisco IOS with a layer-3 software license, a FreeBSD-based server, and a commodity unmanaged layer-2 switch to connect devices.

In order for our collector server to avoid having to ARP (address resolution protocol) for every address in the darknet space, we'll configure the router to forward the darknet-destined traffic to a unique endpoint IP address on the server's darknet Ethernet interface. In order to accomplish this, we suggest dedicating a /30 network for your point-to-point between your router and the darknet interface, such as 192.0.2.0/30. This would make your router's Ethernet interface 192.0.2.1/30 and the collector server could be reached via 192.0.2.2/30. Interface configuration depends largely on the platforms you've selected so we'll assume you're comfortable setting that up on your own. In our examples, we're using Cisco IOS with a layer-3 software license. Once that's done, you'll simply enter the appropriate routing statements to the switch to forward all your darknet traffic to 192.0.2.2 on the collector server, and you're home free:

```
router#conf t
router(config)# ip route 10.0.0.0 ←
255.0.0.0 192.0.2.2
router(config)# ^Z
router# wr
```

You should now be receiving darknet traffic. An example logical topology is shown in Figure 4.

What to do with the traffic once it gets there is another story. The server should be configured not to respond to any data it receives on its darknet interface. Of course, it will ARP for its configured address (192.0.2.2/30 only) in order to establish communications with the router, however all other packets should be discarded by some sort of host-based firewall. As mentioned earlier, no management whatsoever should occur on the darknet interface – you'll need to configure another Ethernet interface on which to perform management and administration. The default route for the system should be the management interface's gateway. For the necessary firewall, your platform selection of the server will impact your firewall selection, but we recommend using a BSD-based system and pf or ipfw2 as your firewall. Whether or not firewall logging should be enabled largely depends on what you'd do with it. We use logfile analysis tools that require logging to be turned on (so that the logs can be parsed and alerts generated); however, depending on several hardware and software choices and the size of your darknet, this logging may severely degrade darknet performance. As an additional safety measure (firewalls can crash or be accidentally turned off), it is a good idea to null-route the darknet traffic should it accidentally go unfiltered.

An example null-route under FreeBSD might look like this:

```
route add -net 10.0.0.0/8 ←↩
127.0.0.1 -blackhole
```

Now that your darknet is humming and you've protected your darknet collector server, you need to store the data in a format useful to your analysis and forensics tools. The most obvious choice would be pcap-formatted binary files as they are nearly ubiquitous in that most network analysis applications can operate on them. The easiest way to do this on an ongoing basis is to use the tcpdump program's built-in rotation feature. The tcpdump program is provided by the Network Research Group of the Lawrence Berkeley National Laboratory. An example tcpdump command line to accomplish the log rotation for us is

```
tcpdump -i en0 -n -w darknet_dump -C125
```

In this example, tcpdump is told to listen on the en0 interface, number-to-name (DNS) resolution is disabled, and a file named darknet_dumpN is written for every 125 million bytes committed, where N increments to make the filenames unique. Again, this will provide a pcap-formatted binary file containing the network traffic. You may then use this file as input to your favorite network analyzer software. The idea here is to keep a copy of the data and use a plethora of different tools to replay the files later to look for interesting characteristics of the traffic. In a normal scenario, you'll be using a program like tcpdump with a specific BPF (Berkeley packet filter) expression to look for things inside these files. While this can be done at run-time (capture-time), by keeping a recording of all traffic, you can use different tools later without the risk of losing anything important.

Another helpful tool that makes it easy to visualize flows of traffic is argus, the network Audit Record Generation and Utilization System developed by QoSient. Although its configuration is too involved to detail here, we utilize argus regularly to watch for interesting flows in our darknets. Argus provides a keen flow-based summary interface that should help you understand exactly what's going on in terms of malicious traffic flows.

In order to visualize the volume of traffic entering your darknet, interface counter-based tools such as MRTG (see *http://www.mrtg.org/*) by Tobias Oetiker should do the trick. MRTG can help you produce beautiful graphs of your not-so-beautiful darknet traffic. There are also dozens of tools out there to parse firewall logs that can be a quick and easy alternative to the more complicated pcap-based analysis tools or argus. Keep in mind the performance problems you'll have with text-based logging of the packet filter and subsequent parsing of those files.

Listing 3. *The basic customer configuration*

```
router bgp XXXX (customer's ASN)
# the customer will install a static route,
# which is redistributed into BGP
# hereredistribute static route-map static-to-bgp
# just like the ISP, use a route-map to set
# and match specific prefix
# attributes
route-map static-to-bgp permit 5
# match the arbitrary tag,
# agreed on by the customer and the ISP
match tag NNNN
set community additive XXX:NNNN
# NNNN is the tag, agreed on by the customer and the ISP
ip route 192.168.0.1 255.255.255.255 Null0 tag NNNN
```

There are literally dozens of tools that can be used within your darknet. To get you started, here's what you'd find in some of ours:

- an IDS sensor (Bro, Snort, et al.),
- a packet sniffer (tcpdump as described earlier),
- a flow analyzer (argus, netflow export from router, SiLK, flowtools),
- a firewall log-file parser that populates RRD databases for graphing,
- MRTG to graph traffic counters,
- p0f (by Michal Zalewski) to categorize platforms of infected/scanning devices.

Deploying Honeynets

Like a darknet, a honeynet is generally a portion of routed, allocated IP space. However, instead of providing a destination where packets go to die, the destination mimics an actual service (or many services), thereby allowing the connection (handshake) to take place, and establishing a complete two-way dialogue. A *honeypot*, or the system mimicking an actual service, is meant to be a tightly held and constantly monitored resource that is intended to lure attackers to probe it and/or infiltrate it. While there are a few different types of honeypots, they all have the same goal: learn the tactics and garner as much information as possible about the attacker.

Physical Honeypots

Physical honeypots are whole machines inside the honeynet with their own IP address, operating system, and service-mimicking tools.

Virtual Honeypots

Virtual honeypots are *software-simulated* complete honeypot systems within the honeynet that mimic environmental conditions such as the operating system, network stack, and services provided as decoys. One physical server may provide a network of thousands of virtual honeypots.

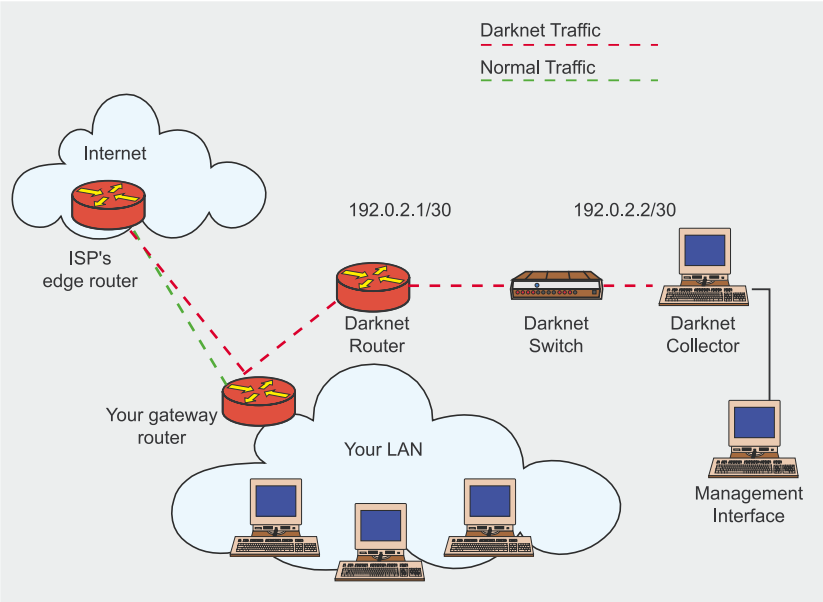

Figure 4. *A reference logical topology for darknets*

Low-Interaction Honeypots

Low-interaction honeypots (the most prevalent type of honeypot in use today) are designed to lure an attacker with one or more presumably exploitable vulnerabilities, establish dialogue, and capture the first few packets of communication with the attacker. Obviously, the attacker or the autonomous malicious software that is conversing with the honeypot will eventually realize the target is unable to be exploited, but before that occurs, valuable information can be exposed, such as the exploitation tactic or the signature of the malicious software. Such low-interaction honeypots are used today to model spammers' tactics (attempting to derive heuristics such as timing characteristics of spammer SMTP transactions, for example).

There are only a few commercial implementations of honeynet technology in general, but the most popular implementation is found in the open source project, honeyd, by Niels Provos. More information on acquiring and setting up honeyd may be found at *http://www.honeyd.org*.

Tip: honeyd is designed to be a virtual honeypot/honeynet that can simulate a number of different operating systems and software components suitable for attracting attackers.

Another low-interaction form of honeypot worth mentioning is a novel concept by Tom Liston called LaBrea. LaBrea (named after the tar pit) is a software daemon (service) that is capable of generating autonomous responses to connection requests across potentially enormous blocks of IP addresses. In short, it creates an environment attractive to scanning/propagating malware, but it has one nasty trick. As soon as the malware attempts to connect, LaBrea slows down the network stack of the sender, sometimes quite significantly. Figuratively speaking, the network stack of the malware-infected system gets stuck in a tar pit. Therefore, there is no interaction at the application layer, but significant interaction at layer 4 when the (TCP) connection handshake attempts take place. LaBrea is even capable of ARPing for all of the virtual IP addresses in its configuration without assigning them to the host system's interfaces, which makes setting it up incredibly easy. More information on LaBrea can be found at *http://labrea.sourceforge.net/labrea-info.html*.

Note: several research bodies have concluded that low-interaction honeypots are a viable tactic against high-performance propagating worms by slowing them down in order to protect network infrastructure. We postulate that the configuration required to realize this benefit is obtuse at best. However, LaBrea and honeyd may both be configured to create such a worm-unfriendly environment.

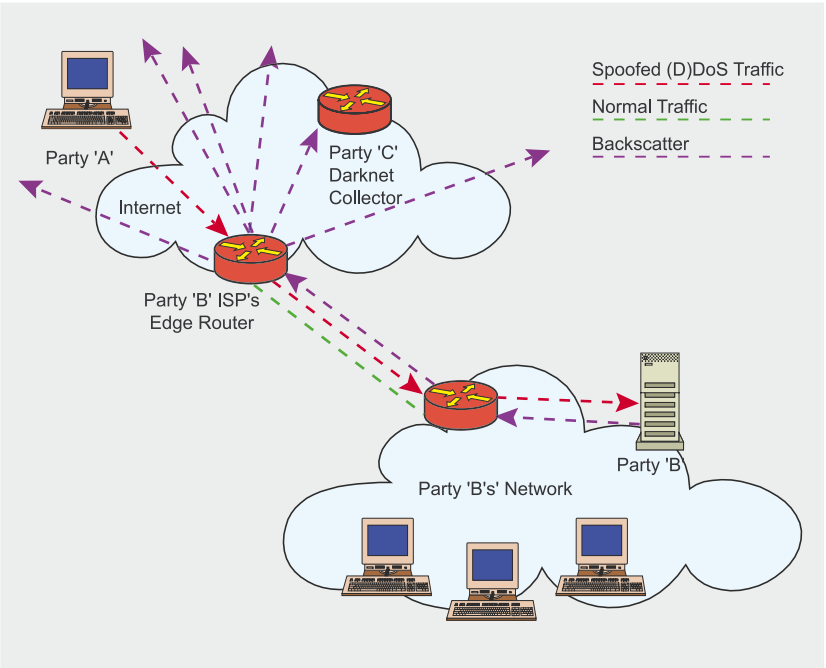

Figure 5. *An example of backscatter during a DDoS attack*

High-Interaction Honeypots

High-interaction honeypots are less used, but exceedingly valuable. As opposed to simply capturing the first few transactions in a dialogue between an attacker and the honeypot, a high-interaction honeypot is designed to let an attack completely infiltrate the system on which it resides. In this scenario, useful information captured will not only include the probing technique and the exploitation used, but it will also allow the security administrator to watch over the attacker once he gains access to the system, unwittingly exposing his intentions and tools.

There is a non-profit organization known as The Honeynet Project (see *http://www.honeynet.org/*) that produces a great deal of intelligence and some easy-to-use tools designed to enable users to deploy high-interaction honeypots. They also provide excellent forensics-type tools to analyze the data collected during infiltrations into the honeypots.

Tip: the Honeynet Project (*http://www.honeynet.org/*) publishes a number of fantastic tools for use in deploying your own honeynets. We recommend paying particular attention to the Honeywall, Termlog, and Sebek tools. Likewise, the project team has also developed an excellent book on the psychology, tactics, and tools used by attackers as gleaned through honeynet technologies. The book, *Know Your Enemy,* which at the time of this writing is in its second edition, is available through the *honeynet.org* web site and proceeds from its sales are used to help fund honeynet research.

Recommendations for the Use of Honeynets

For research organizations or those with a lot of money and time to burn (do you know of any?), honeypots can be an invaluable tool, but we do not recommend utilizing honeypots inside the everyday enterprise. However, while not suitable for everyday use, when an innocuous piece of malicious software rears its ugly head and no sniffer or forensics tools help identify the problem to the extent that your administrator can solve it, a honeynet may be implemented on demand in order to establish communication by posing as a target the malicious software is expecting, thereby exposing enough information in order to adequately identify the attack. Another on-demand use is as a means to verify a suspected infiltration. Therefore, it should be another arrow in the security administrator's quiver.

One implementation worth mentioning is in use at one of the world's largest chipmakers. They have, throughout their network, Linux servers running VMWare, on top of which are running four virtual machines, one for each of the Windows OS varieties common within the enterprise – NT, 2000, 2003, and XP. Each is kept current with the standard corporate patch levels. The Linux OS monitors those for traffic and changes, as a means of detecting new worms (or other threats) that may circulate within the enterprise. They're essentially using this environment as a combination honeynet and IDS for worms. More details on this implementation may be found at *http://phoenixinfragard.net/meetings/past/200407hawrylkiw.pdf*

Implementing Sinkholes to Defend Against DDoS Attacks (Blackhole Routing)

Another novel use of sinkhole technology is as a defense tactic against (distributed) Denial-of-Service attacks. In the *Background and Function* section earlier in this article, the first example given was the simplest form of this blackhole routing technique. Once the exact target of an attack has been identified, the IP address being targeted was diverted to the discard interface at the edge of the network, before traversing the final link to the target. This freed the target network from total disruption through link saturation, but still likely impacted performance network-wide, especially for adjacent customers that shared some of the carrier's edge topology with the target network. Today, large telecom carriers have architected their networks and included sophisticated versions of this defense measure as part of their overall network design philosophy. In many cases, the carriers are now able to use a traceback technique in order to locate the ingress points of the attack and blackhole the malicious packets there (at the ingress points themselves) instead of allowing the attack to clog the carrier backbone all the way downstream to the target network's link. This traceback technique is largely unnecessary because the carriers' blackhole routes are customarily announced network-wide among their edge routers using a BGP community, thereby blackholing the malicious traffic at each ingress point, allowing them to blackhole attacks as they enter and (in many cases) avoid backbone and edge congestion all together. Some have even extended the control and automation of this capability to the end customer through what are known as customer-triggered real-time blackholes.

Triggered Blackhole Routing

As mentioned above, many large ISPs have implemented a distributed, automated system for *triggering* blackhole routing on targeted IP addresses. The trigger may be initiated by the ISP or by customers, either manually or automatically. Triggered blackhole routing utilizes the simple sinkhole described earlier in the section *Background and Function*. The sinkhole may be configured on all ingress (edge) routers within the ISP network where the ISP exchanges traffic with other providers or customers. When an

Table 1. *ICMP Packets*

ICMP Packets	Description
3.0	Network unreachable
3.1	Host unreachable
3.3	Port unreachable
3.4	Fragmentation required
3.5	Source route failed
3.6	Destination network unknown error
3.7	Destination host unknown error
3.10	Host administratively prohibited
3.11	Type of service network unreachable
3.12	Type of service host unreachable
3.13	Communication administratively prohibited
11.0	TTL expired during transit
11.1	Fragment reassembly timeout

TCP Packets	Description
RST bit set	TCP Reset

attack against a network target is identified, the ISP or the customer may announce the *attacked* prefix (or a more-specific prefix) into the BGP routing table. The attacked prefix is tagged with a next-hop that is statically routed to the discard interface on all edge routers, and propagated within the ISP's network via internal BGP (iBGP). Then, wherever the packets destined for the attacked prefix enter the ISP network (the ingress point), they are immediately sent to the discard interface on the closest router announcing the attacked prefix.

The following steps are necessary for the ISP to implement the distributed blackhole mechanism:

- select a non-globally routed prefix, such as the Test-Net (RFC 3330) 192.0.2.0/24, to use as the next hop of any attacked prefix to be blackholed. Using a prefix of length 24 allows you to use many different IP addresses for specific types of blackhole routing. You may wish to differentiate between customer, internal, and external blackhole routes,
- configure a static route on each ingress/peering router for 192.0.2.0/24, pointing to the discard interface. For example: `ip route 192.0.2.0 255.255.255.0 Null0`,
- configure BGP and policy route-maps to announce a prefix to be blackholed as shown on listing 1.

In the example configuration, we are redistributing static routes into BGP that match *tag 199* (see below), setting the next hop to an IP address that is routed to the discard interface, setting the local preference to 50 (less preferred), and ensuring we do not leak these routes to any of our external peers (no-export).

Once this basic configuration is done, the trigger can be initiated by the ISP entering a static route for the attacked prefix (or host) to be blackholed, for example:

```
ip route 172.16.0.1 255.255.255.255
192.0.2.1 Null0 tag 199
```

The static route above is the *trigger* that kicks off the blackhole routing process. The router that this route is configured on will announce the route through iBGP to all internal routers, including edge routers. Any router with a static route to the discard interface for 172.16.0.1/32 will immediately blackhole traffic locally.

The ISP may wish to set up automated triggering through BGP as well, so a BGP customer could trigger the blackhole route independent of ISP intervention. This is the most powerful aspect of triggered blackhole routing. The configuration on the ISP side is slightly different in that communities and ebgp-multihop are used to properly receive and tag the routes learned from the customers. The basic configuration on the ISP side looks like on Listing 2.

The ISP already has the < *blackhole-ip* > statically routed to discard interfaces throughout the network, so as soon as the customer announces the prefix to blackhole, the ISP redistributes that internally and traffic to this prefix is blackholed at the *edge* of the ISP network.

The basic customer configuration looks like on Listing 3.

Once the BGP configuration is in place, the customer need only install a static route for the prefix # being attacked. With some very basic configuration in BGP, and the help of your ISP, you now have a very fast method to respond to Denial-of-Service attacks against a single host, or an entire prefix.

Note: be sure to check with your ISP's technical contact before implementing your blackhole-triggering solution as ISP implementations of this concept differ slightly.

Table 2. *Summary Checklist*

Step	Description
Understand how your ISP can help you during a DDoS attack.	Make an action plan for dealing with DDoS attacks that includes strategies that leverage your ISP's capabilities in the realm of real-time blackholing. Open dialogue between your organization and your ISP about enabling you to create customer-triggered real-time blackholes to protect yourself without spending precious time with their escalation procedures.
Consider implementing an internal darknet.	Remember, an internal darknet gives you the ability to catch worms earlier than your anti-virus vendor. Likewise, it exposes network misconfigurations that you'll be glad you knew about.
Consider implementing an external darknet.	External darknets can give you insight to what your network is being hit with from the outside and the tools you use with it may be easier on the eyes than a standard firewall log. The backscatter collected from an external darknet can give you intelligence about when your network is being implicated in an attack on a third party.
Explore using honeypots for research if you have the time and resources.	Thought most organizations won't see significant benefit from implementing a honeynet (outside of awareness), they are invaluable to information security researchers. Consider the implications of deploying a honeynet within your organization. Such consideration should include exploration of state laws that might have a bearing on your decision.

Backscatter and Tracebacks

In this section, we'll explore creative uses of decoy networks to detect attacks and spoofing and also to help track down the miscreant.

Backscatter

It seems fitting after all of this discussion on decoy networks and DDoS attacks to mention the notion of backscatter. For an entire semester during my freshman year in college, I wrote letters (yes, the physical kind) to various friends who were moving around a lot. Being the absent-minded individual that I am, I would consistently write the wrong return address on my envelopes. I'd forget to put my dorm suite number on them or it would be completely illegible (I had discovered beer). Occasionally, one of my friends that I wrote would have moved and the letter I'd sent them bounced back to me with a post office notification stating *return to sender.* Only, since my return address was written incorrectly, the bounce-back didn't go to me, it went to the resident office downstairs who called me and let me know (by matching my name) I had again written my return address wrong and there was a letter there waiting for me to pick it up and resend. That *return to sender* bounce-back is a form of backscatter. Of course, the backscatter indicated to the resident office that I had been sending mail (and to whom).

On the Internet, when party A intends to perform a Denial-of-Service attack against party B, but party A wants to conceal his identity, he normally writes the wrong source address on his attack packets (the IP headers are forged to look like they came from parties A-Z, for example, only A-Z in IPv4 is 2^{32} permutations). During such attacks, routers and other network devices along the path inevitably send back a variety of messages that range from connection resets to quench requests to unreachable notifications. Since these messages are *returned to sender*, and since the sender is forged, parties A-Z all receive them and thus gain knowledge of the attack on party B, just as the resident office gained knowledge of the mail I was sending. This is depicted in Figure 5.

In today's packet filtering world, most of these backscatter messages are silently discarded by our firewalls because they are seen as responses to a message we did not send. But with an external darknet network implemented as explained earlier, we can look for these backscatter packets and determine when our address space is being implicated in an attack on another party. The following types of packets appearing in

On the Net

- *http://www.amazon.com/gp/product/0072259558/* – Extreme Exploits: Advanced Defenses against Hardcore Hacks, published by McGraw-Hill/Osborne Copyright 2005
- Internet RFCs 3330 (Special-use IPv4 Addresses) and 3882 (Configuring BGP to Block Denial of Service Attacks)
- *http://www.cymru.com/Darknet/* – The Team Cymru Darknet Project
- *http://www.tcpdump.org/* – The home of tcpdump and libpcap
- *http://www.qosient.com/argus/flow.htm* – The home of ARGUS
- *http://www.honeyd.org* – The home of Honeyd
- *http://www.honeynet.org* – The home HoneyNet Project
- *http://lcamtuf.coredump.cx/p0f.shtml* – The home of the p0f tool
- *http://www.secsup.org/Tracking/* – Chris Morrow and Brian Gemberling's article on ISP blackholing and backscatter analysis
- *http://phoenixinfragard.net/meetings/past/200407hawrylkiw.pdf* – Dan Hawrylkiw's presentation on honeynets
- *http://www.openbsd.org/faq/pf/* – A FAQ about the OpenBSD packet filter

About the author

Mr. Oppleman is an accomplished author, speaker, and teacher in the field of network security and a consultant to some of the world's most admired companies. Mr. Oppleman's open source software has been distributed to hundreds of thousands of computers worldwide and he holds US intellectual property patents in distributed adaptive routing and wireless consumer applications. Most of the content from this article has been taken from Mr. Oppleman's book, *Extreme Exploits: Advanced Defenses Against Hardcore Hacks*, which is published by McGraw-Hill/Osborne (Copyright 2005) and available at your favorite bookseller.

a darknet may be classified as backscatter and indicate your (darknet) address space is being implicated in an attack.

Traceback

Now that we have a handle on backscatter, how can we use it? In a network with multiple Internet transit gateways, it may be useful during a debilitating attack to locate the ingress point of the *bad packets*. This technique, known as a *traceback*, is useful in that once we identify the specific ingress point on our (or our ISP's) network, we may be able to drop the traffic there and reduce the load on our links, potentially even allowing *good* traffic to flow (through alternate gateways), unlike the simpler DDoS blackhole protection tactic discussed earlier. Traceback allows us to utilize the backscatter we collect in our darknet(s) as a means of finding the point where the attack is entering the network. Unfortunately, this is really only viable for ISPs or for far-reaching data networks with many Internet gateways. Some dependencies beyond that description include utilization of the blackhole defense mechanism at *every* Internet gateway. Since major ISPs do this along with a handful of global enterprise networks, it seems fitting to at least explain the process.

Assuming you have the network setup as described above, you can perform a traceback in the midst of a Denial-of-Service attack in three easy steps:

- identify the target and verify that the attack traffic is being spoofed (if it isn't, this traceback tactic will be fruitless),
- blackhole the route for the specific hosts (/32s likely) being attacked at each of your gateways. Exercise caution and follow best practice concerning the use of forwarding to the discard interface in lieu of using a packet filter to drop the attack packets. This blackhole operation will cause this gateway router to begin generating ICMP unreachable messages, which are (attempted to be) returned to the spoofed sources of the attack packets,
- inside your darknets, use your darknet tools you've put into place to look for the backscatter traffic (probably in the form of ICMP unreachables) with your gateway routers' IP address in it. Any IP addresses of your gateways you see as the source of these backscatter packets validate that those gateways are actually the ingress point(s) of the attack traffic. Voilá, you've found where the attack is entering the network. Even if you don't have your sophisticated darknet tools set up, a simple access list applied to the router interface of your darknet can do the trick for you as depicted below: `access-list 105 permit icmp any any unreachables log; access-list 105 permit ip any any`.

Then, if you enter terminal monitoring mode on this access list (or simply tail the log), you'll get a poor man's backscatter report that you can look inside for the IP addresses of your gateways.

The traceback tactic and the blackhole defense against DDoS attacks are useful in situations where the floods of malicious traffic have forged (spoofed) headers. This was the customary way of performing such attacks until recently. But with the proliferation of zombied machines and botnets, many attackers have stopped spoofing DDoS packets all together – there's no reason to forge headers if your army of attacking systems are everywhere. Likewise, spoofed DDoS attacks have declined significantly as a result of the wider deployment of uRPF and ingress filtering. ●