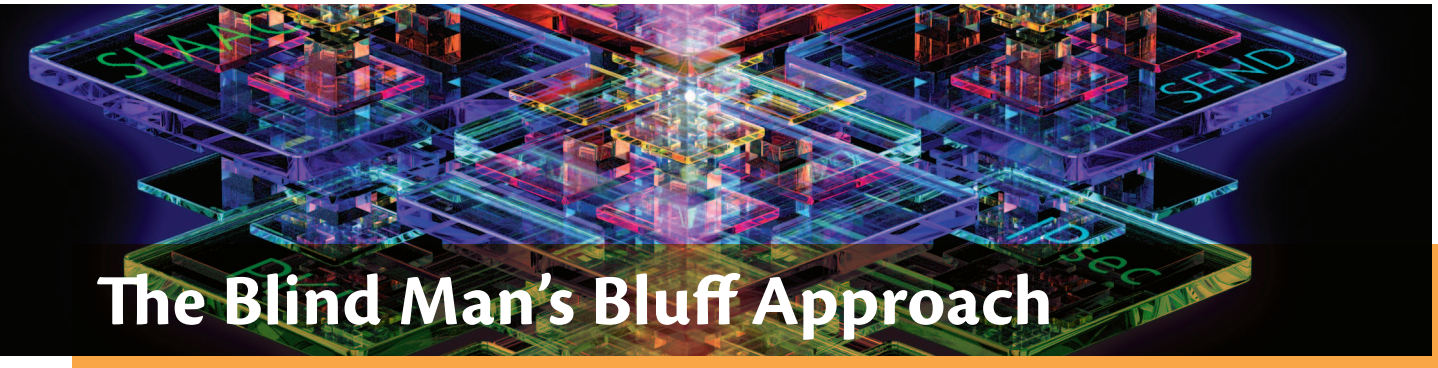

The Blind Man's Bluff Approach to Security Using IPv6

Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront | Virginia Tech

The problem with static defenses is that adversaries have unlimited time to circumvent them. A moving-target defense based on the Internet Protocol version 6 can dynamically obscure network- and transport-layer addresses and help prevent targeted attacks, host tracking, and eavesdropping.

Many security technologies help protect networks and communicating hosts. This task is more challenging owing to these technologies' static nature. Adversaries can continually launch attacks at these static defenses until a vulnerability is exploited. You can think of static security as a Whac-A-Mole game with only one mole hole. On the other hand, a moving-target defense "moves" the target, causing adversaries to exhaust their resources while attempting to locate the target.

We've developed a moving-target defense that leverages the vast address size of the Internet Protocol version 6 (IPv6) to present adversaries with more "mole holes" than are statistically feasible for them to test. Our technique, the Moving Target IPv6 Defense (MT6D), operates at the network layer and doesn't require any modifications to the existing IPv6 protocol, making it easy to integrate into networks and defense-in-depth strategies.

MT6D functions by dynamically obscuring both sender and receiver addresses (see the "Related Work in Obscuring Network Addresses" sidebar). Addresses can rotate at any time without disrupting ongoing sessions or requiring additional handshaking. The more often addresses rotate, the less time adversaries have to locate and detect a target host. In IPv6, finding a target address is statistically infeasible given that a single IPv6 subnet contains 2^{64}, or 1.8×10^{19} addresses.

MT6D greatly improves security, privacy, and anonymity. It enhances security because attackers will have difficulty finding the host, which makes host tracking and traffic monitoring more complicated, thereby increasing anonymity and privacy. This article describes how we achieve our moving-target defense.

IPv6 Addressing Problem

IPv6 designers developed a technique called stateless address autoconfiguration (SLAAC) to reduce the administrative burden of managing the immense IPv6 address space. Owing to most operating systems' current accepted definition of SLAAC, a node's IPv6 address's interface identifier (IID), or host portion, is deterministic across networks. For the last 64 bits, the node automatically configures an address on the basis of its network interface's media access control (MAC) address. Even operating systems that obscure addresses according to Request for Comments (RFC) 4941 contain a static IID used for neighbor solicitation.[1] These static IIDs can identify a particular node, even as the node changes networks.

Using Virginia Tech's campuswide IPv6 production network, which supports more than 30,000 IPv6 nodes

1540-7993/12/$31.00 © 2012 IEEE Copublished by the IEEE Computer and Reliability Societies

Related Work in Obscuring Network Addresses

Victor Sheymov developed a technique to dynamically obscure cybercoordinates to provide intrusion protection from certain network attacks.[1] However, unlike the Moving Target IPv6 Defense (MT6D), Sheymov's design doesn't provide anonymity because it uses the Domain Name System (DNS) to assign permanent names to devices. Attackers will have little problem correlating traffic using hosts' DNS names. Sheymov also used a management unit to distribute addresses. In MT6D, communicating hosts can calculate their own addresses independently.

Russell Fink and colleagues also proposed a technique to dynamically obscure host addresses, called Adaptive Self-Synchronized Dynamic Address Translation (ASD).[2] ASD is similar to MT6D in that its objective is to hide communicating hosts' locations. It does this through a handshake process between a trusted sender and receiver enclave to assign source and destination addresses. Obscured addresses are selected from those available to the ASD enclave. MT6D improves on ASD by letting MT6D hosts communicate without needing to reauthenticate each time an address rotates. Reauthentication minimally gives away the communicating trusted enclaves' identities. In MT6D, authentication handshakes aren't necessary, which provides further protection for communicating hosts.

Two other proposals obscure addresses to achieve anonymity—privacy extensions and cryptographically generated addresses (CGAs). Privacy extensions were designed to protect Internet Protocol version 6 (IPv6) addresses that use stateless address autoconfiguration.[3] CGAs were designed to securely associate IPv6 addresses with public keys for use with the Secure Neighbor Discovery protocol.[4] Neither of these schemes dynamically obscures addresses. Once an address is assigned, it remains constant until the network session is terminated. A third party monitoring the connection can accomplish both address tracking and traffic correlation. These techniques also obscure only the source address. MT6D not only rotates addresses multiple times in a single session but does so for both the source and destination addresses.

References

1. V.I. Sheymov, *Method and Communications and Communication Network Intrusion Protection Methods and Intrusion Attempt Detection System*, US Patent 2010/0042513 A1, Patent and Trademark Office, Feb. 2010.
2. R.A. Fink et al., *Method and Apparatus for Providing Adaptive Self-Synchronized Dynamic Address Translation*, US Patent 7,043,633 B1, Patent and Trademark Office, May 2006.
3. T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC 4941, Internet Eng. Task Force, Sept. 2007.
4. M. Bagnulo and J. Arkko, "Cryptographically Generated Addresses (CGA) Extension Field Format," RFC 4581 (Proposed Standard), Internet Eng. Task Force, Oct. 2006.

daily, we were able to validate that IPv6 address tracking and monitoring are possible. We conducted testing with an Android mobile device using MAC-based IIDs to form wireless IPv6 addresses.

The first part of our test involved tracking the mobile device as it moved around campus. Geotemporal tracking was possible because the campus network contains different subnets that cover different geographic areas. We programmed a script that continually sent echo requests to the different subnets on campus. When we received an echo reply, we stored its time and location. Figure 1 demonstrates the results of a successful tracking attempt.

The second part of our test involved traffic monitoring. Our goal was to demonstrate that we could isolate a node, regardless of subnet, and collect all of its associated network traffic. We placed a sensor at the network border to collect all IPv6 traffic leaving the network. Using a packet sniffer, we successfully filtered the traffic related to the node in question across different subnets.

System Overview

MT6D lets hosts communicate with each other over the public Internet while maintaining anonymity from targeting, tracking, and traffic correlation. The system does this by tunneling IPv6 packets inside MT6D packets. The new header created by MT6D dynamically obscures the source and destination network- and transport-layer addresses for both communicating hosts. MT6D can also encrypt the entire tunneled packet to prevent attackers from analyzing payloads or header fields.

MT6D rotates addresses on the basis of a set of parameters known to only the two communicating hosts; there's no need for a trust model that extends beyond the communicating pair of nodes. In addition, MT6D doesn't require communicating hosts to exchange parameters prior to each communication session, meaning that dynamic addresses can't be linked to host identities. A unique feature of MT6D is that the dynamically obscured addresses can change in the middle of ongoing sessions without breaking the connection or requiring a new handshake. This feature allows addresses to change as often as the security posture dictates, rather than being constrained by ongoing network sessions. Figure 2 provides a simplified example of two hosts using MT6D over a network, as well as what this interaction might look like to attackers.

Threat Protection

MT6D is designed to protect against attacks targeted at specific hosts, regardless of whether the threat is internal or external to the trusted network. Although it wasn't designed to protect against unspecified attacks, it affords some measure of protection there as well.

Targeted Attacks

Targeted attacks are aimed at a particular host or group of hosts. These attacks can be passive or active.

Passive attacks. Adversaries passively targeting hosts are interested in discovering host identities and activities, including where hosts are and what they are doing there. Adversaries can determine a host's identity by observing authenticated network traffic or SLAAC addresses or by correlating network traffic over time. Discovering a host's identity is an attack against anonymity. Discovering a host's activities is an attack on privacy.

If adversaries know the targeted hosts' IPv6 addresses, they can determine host locations. Static addresses support the extended tracking of hosts because they remain constant over time. MAC-based IIDs are an example of addresses that keep a portion of the address static regardless of the subnet they connect to. MAC-based IIDs also make hosts more susceptible to traffic correlation because IIDs remain constant regardless of subnet. Even IIDs that don't remain constant over multiple subnets are typically still static for each individual subnet.

Active attacks. Adversaries interested in disrupting, intercepting, or modifying target hosts' network communications will launch active attacks. These attacks can come in the form of denial-of-service (DoS) attacks, with the goal of disrupting network communications, or man-in-the-middle attacks, with a goal of, for example, intercepting and modifying network communications. Hosts with static network addresses are especially susceptible to active attacks because attacks against a specific network address either persist or are easily renewed after the host changes subnets.

MT6D's responses. MT6D protects against targeted passive attacks in a few ways. First, it achieves anonymity by obscuring and frequently rotating host addresses. It also achieves privacy because adversaries can't link addresses to specific hosts at specific locations. Single sessions can be spread over multiple sender and receiver addresses, thus preventing adversaries from easily determining whether multiple packets belong to the same host or even the same session. Second, MT6D can encrypt entire packets prior to tunneling them, preventing both traffic correlation and observation of authenticated traffic.

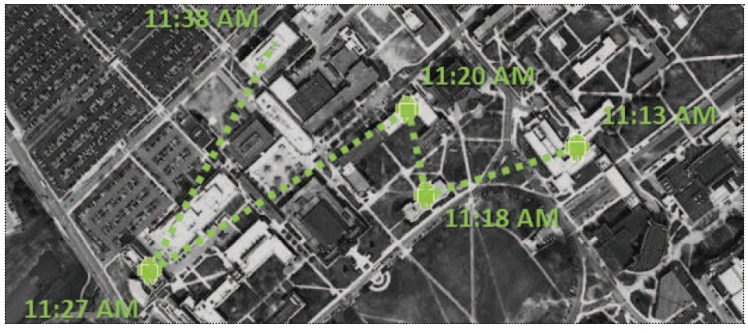

Figure 1. Geotemporal plot of a wireless node's movement in a campus network. The times on the figure indicate when and where the target host's interface identifier was detected as it moved through the campus-area IPv6 network.

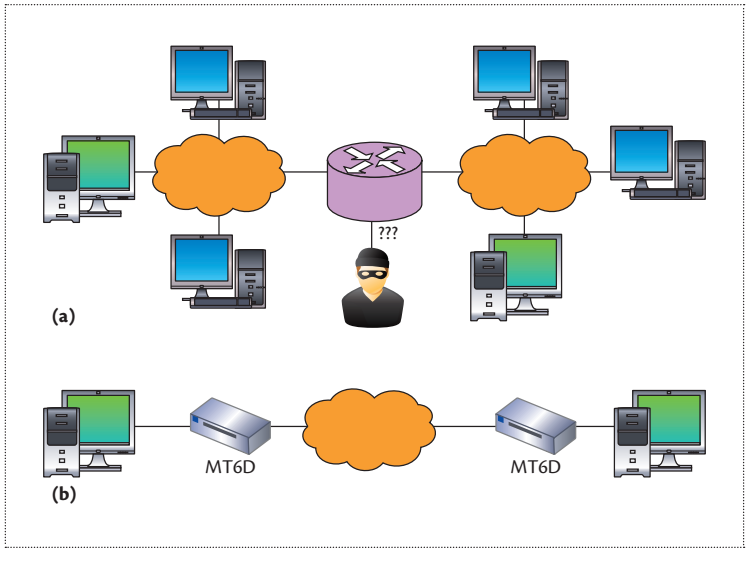

Figure 2. Two hosts using the Moving Target IPv6 Defense (MT6D) on a network. (a) The attacker sees communicating hosts appearing and disappearing on seemingly random addresses. (b) The actual network configuration.

MT6D protects against active attacks by providing dynamic addressing, which serves two purposes. First, adversaries can't easily locate the hosts they're trying to target owing to the obscured addresses. Second, adversaries who happen to locate a targeted host can only attack for, at most, the length of time between address rotations. After such time, adversaries are forced to relocate the target host.

Unspecified Attacks

Adversaries might not be concerned with which specific hosts they attack. Unspecified attacks might aim to passively collect network traffic to determine the number of hosts on a network or the nature of network

communications. Adversaries might attempt to block any discovered hosts from using network resources. Because these adversaries aren't concerned with victims' specific identities, MT6D can't prevent these attacks. However, MT6D does limit the damage that unspecified attacks inflict. For example, adversaries can't accurately determine network density, owing to the number of new addresses they observe. They will even have difficultly determining the nature of network communications because a single session might be spread over multiple source and destination addresses. Unspecified active attacks are also limited. Because host addresses constantly rotate, active attacks against observed hosts are limited to the amount of time between address rotations.

System Design

Here, we examine MT6D's design, including dynamic addressing, IID lifetime, time incrementation, MT6D tunneling, symmetric keys, and architecture.

Dynamic Addressing

Dynamic addressing nondeterministically modifies both communicating hosts' source and destination network- and transport-layer addresses. Network-layer addresses are modified by a function obscuring the communicating hosts' IIDs using three components—a host's 64-bit extended unique identifier IID,[2] a symmetric key, and a time stamp. Of these three values, only the symmetric key must be kept secret. The three values are concatenated and hashed; the obscured IID is constructed from the leftmost 64 bits of the hash (bits 0–63) and has the form

$$IID'_{x(i)} = H[IID_x \| K_S \| t_i]_{0 \to 63}, \quad (1)$$

where $IID'_{x(i)}$ represents the obscured IID for host x at time t_i, IID_x represents the unobscured IID of host x, K_S represents the shared symmetric key, and t_i represents the time at instance i. The leftmost 64 bits of the hash value are denoted by $H[\cdot]_{0 \to 63}$. The MT6D IPv6 address is formed by concatenating the host's subnet with $IID'_{x(i)}$ as $IP'_{x(i)} = Subnet_x \| IID'_{x(i)}$.

In addition, MT6D obscures ports. If port numbers are left unobscured, attackers can use the collected packets' port numbers to correlate the packets with one another. MT6D includes two techniques to dynamically obscure the source and destination ports. The first technique lets hosts specify a port address range for MT6D use. Users can also configure MT6D to use common ports that more closely mimic normal network traffic or specify a port range that conforms to firewall rules.

The second technique obscures port numbers using a method similar to the IID obscuration in Equation 1. For example, obscured ports could leverage the unused bits of the hash calculation as follows:

$$Src\_Port_i = H[IID_{Src} \| K_S \| t_i]_{64 \to 79};$$

$$Dest\_Port_i = H[IID_{Dest} \| K_S \| t_i]_{64 \to 79}.$$

The source port, Src_Port_i, uses the next 16 bits of the source IID hash (bits 64–79), and the destination port, $Dest\_Port_i$, uses bits 64–79 of the destination IID hash. The MT6D header uses the obscured port numbers as its port numbers. Because the current MT6D implementation encapsulates all packets using the Unreliable Datagram Protocol (UDP), no other transport-layer header fields need to be obscured.

IID Lifetime

Hosts using MT6D rotate to the next dynamic address at every t_i increment. At each time increment, MT6D recalculates the source and destination network- and transport-layer addresses of both communicating hosts. MT6D purges the addresses for t_{i-1} to prevent any connection attempts or replay attacks from malicious third parties.

Each time a host recalculates its obscured IID, it must notify the local gateway device of its new IPv6 address so packets can be properly forwarded. This notification occurs through the Neighbor Discovery Protocol (NDP).[3] NDP serves two purposes. First, neighbor solicitation and advertisement messages perform duplicate address detection to verify that the new address doesn't conflict with a preexisting address on the subnet.[4] Second, NDP ensures proper notification of new MT6D IPv6 addresses to communicating devices.

At any given time, MT6D devices maintain multiple IPv6 addresses that correlate with a single obscured host, which minimizes latency and packet loss. Future obscured IPv6 addresses are precalculated and bound to the public-facing network interface controller (NIC) so that packets aren't lost during address transition periods. To accommodate this requirement, host x binds $IP'_{x(i+1)}$ at time t_i. Addresses for previous time increments are purged.

The host stores two obscured IPv6 address states—$IP'_{x(i)}$ and $IP'_{x(i+1)}$. The $IP'_{x(i)}$ state corresponds to the current computed obscured IPv6 address. This is considered the active state. The $IP'_{x(i+1)}$ state corresponds to the obscured IPv6 address that will be used at the next time increment. The state at t_{i+1} is stored but not used until t_i increments to the next time interval. It's precalculated to verify the validity of $IP'_{x(i+1)}$ in the subnet prior to time t_i. Again, the $IP'_{x(i+1)}$ state is purged from the

host. Any packets delayed past the active state will be discarded and handled according to the original packet's appropriate transport-layer protocol.

Time Incrementation

Time T increments at t_i intervals. Time intervals vary for each communicating address pair. This is done for both network management and security and privacy purposes. From a network management perspective, varying address rotation times lessens the burden on networking equipment. A large number of MT6D users on a subnet all changing addresses at the same time have the potential to burden a router by forcing it to simultaneously bind every new address. Varying hosts' rotation times distributes the binding of new addresses. From a security and privacy perspective, rotating addresses makes identifying which hosts are using MT6D more difficult. For instance, if all hosts using MT6D change addresses at the same time, attackers can determine the number of hosts using MT6D by observing address changes at rotation times.

MT6D Tunneling

Rather than rewriting each original packet using the communicating hosts' obscured source and destination IPv6 addresses, MT6D encapsulates the original packet in a tunnel. Tunneling the original packet retains established end-to-end connections between the source and destination as well as flags specific to that session. Not only does the MT6D application become transparent to the host, but the MT6D connection can have different configuration settings.

An MT6D packet is formed by removing the source and destination addresses from the original packet. The Ethernet header is also removed to anonymize the MAC addresses. (This doesn't present an issue because it will be reconstructed at the destination MT6D device.) The entire packet is then prepended with an MT6D header that is formed using the dynamically obscured source and destination addresses.

Each packet is encapsulated using UDP to prevent Transmission Control Protocol (TCP) connection establishment and termination from occurring each time an MT6D address rotates. Encapsulating packets using UDP has minimal effect on the original packet's transport-layer protocol. Because transport-layer protocols are end to end, decapsulation will occur before the host processes the original packet. A session using TCP will still exchange all required TCP-related information. This information will simply be wrapped in an MT6D UDP packet. In addition, after a retransmission timeout, the end host will retransmit any lost packets that originally used TCP. (Fragmentation isn't an issue because it occurs at the source in IPv6.)

MT6D also handles any Internet Control Message Protocol version 6 (ICMPv6) messages from intermediate nodes. In IPv6, intermediate nodes generate many critical messages, including NDP messages and the "packet too big" message, which notifies the sender that the packet exceeds a node's maximum transmission unit somewhere along the physical link. Because intermediate nodes don't know the sender's actual address, the sender's MT6D device re-forms the packet for delivery to the sender.

Encrypted tunnel. By default, MT6D encrypts each original packet before appending it with the MT6D header. Original packet encryption prevents adversaries from gleaning useful information from the packet. For example, if the original packet is sent using TCP, the header is encrypted so attackers can't correlate network traffic using the TCP sequence numbers. In addition, the nature of the network traffic is also kept private.

Another benefit of encrypted tunnels is that hosts can authenticate traffic to one another while maintaining anonymity. Because the original authenticated packets are encrypted using a symmetric key, adversaries can't detect that the packets are authenticated. In addition, adversaries won't find identifiable information about the communicating hosts from captured packets.

Unencrypted tunnel. MT6D includes the option to tunnel unencrypted original packets. Because the source and destination addresses are stripped from the original packet header, address tracking isn't feasible. In addition, attackers can't determine which two hosts are communicating. However, unencrypted tunnels don't prevent traffic correlation because the remainder of the original headers and payloads stays intact. Traffic correlation requires deep packet inspection because any relevant header fields are embedded in the MT6D packet payload. Unencrypted tunnels might be preferable in environments where minimizing computational expense is more important than preserving privacy. In both encrypted and unencrypted tunnels, protection from targeted network attacks is provided.

Symmetric Keys

Symmetric keys in MT6D are preloaded, exchanged out of band, or exchanged in band. Of the three techniques, in-band key exchange is the least preferable because it's the most susceptible to eavesdropping and can expose the communicating hosts' IPv6 addresses. We don't attempt to solve key-exchange issues; we only point out the possible options for establishing keys. It's worth noting that even if malicious hosts learn a host's real IPv6 address, they can't match observed MT6D packets to that particular host.

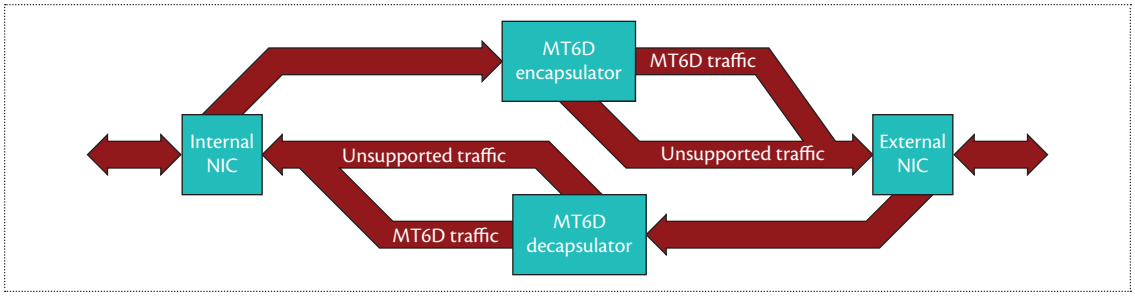

Figure 3. MT6D host architecture. The network interface controller (NIC) provides the link to the protected host while the external NIC connects to the public-facing network. The internal NIC directs outbound traffic to the encapsulator for packaging before forwarding it to the external NIC. The external NIC receives the inbound traffic, which is decapsulated, then delivers it to the protected host via the internal NIC. An optional unsupported traffic path facilitates communication with hosts that don't use MT6D, such as some webservers.

Users should periodically change symmetric keys to prevent key compromise. MT6D includes a means to periodically generate new symmetric keys in band using IPv6 destination options.[5] New symmetric keys exchanged in band in an ongoing MT6D session don't expose hosts' real addresses and are completely transparent to the communicating hosts.

Architecture

Figure 3 illustrates the architecture for a single MT6D host. The other end of the MT6D tunnel mirrors Figure 3. MT6D is designed to be virtually transparent to the user. Users encapsulate each packet and send it to the internal MT6D NIC. This can be a physical NIC, when MT6D is implemented in a separate device, or a virtual NIC, when implemented in the host's software. The internal NIC directs all incoming packets to the MT6D encapsulator.

The MT6D encapsulator transmits all outbound packets. Upon receipt of a packet, the encapsulator checks whether an MT6D profile exists for the sender/receiver pair. The encapsulator maintains a table of all valid MT6D destinations that a sender trusts, called *profiles*. Each profile includes the shared symmetric key that is valid between the host and each receiver. If no profile exists, the packet is treated as non-MT6D traffic, also called *unsupported traffic*. An optional unsupported traffic path facilitates communication with hosts that don't use MT6D, such as some webservers. Depending on the desired security level, unsupported traffic is either blocked or immediately forwarded to the nearest gateway device. Packets that match profiles are placed in an MT6D tunnel. The final step is to pass the packet to the external NIC and transmit it.

The MT6D decapsulator receives all inbound packets. The external NIC receives each packet and checks it for an MT6D profile. Those packets that don't match profiles are considered unsupported and optionally delivered immediately to the host. Packets that match MT6D profiles have the tunnel header stripped off. The packet is then decrypted, the source and destination addresses are rewritten to the original packet header, and the packet is delivered to the host via the internal NIC.

Configuration

Popular configurations include MT6D as integrated software on the host and as a separate stand-alone device. Both configurations adopt a trust model that assumes trust only between the sender and receiver. In a model in which all insiders are trusted, the stand-alone configuration can be expanded to the border of a trusted network.

Integrated Software

Integrating MT6D into the host device has several advantages, the biggest of which is mobility. With MT6D on the host device, MT6D can be implemented on handheld devices. Another advantage is cost. Because MT6D is loaded directly onto the host device, additional hardware isn't necessary. In addition, managing the configuration is easier and there's no need to transfer keys or preferences to a separate device.

Stand-Alone Device

Another option is configuring MT6D on a separate stand-alone device that is transparent to users. To use the stand-alone MT6D device, users just plug it in and connect network cables. This is especially useful if users have devices running different operating systems. In this configuration, MT6D is platform independent and computational complexity is offloaded to the MT6D device.

MT6D can also be used as a border device in a trusted environment, similar to a virtual private network (VPN). The main difference, which is an improvement

over traditional VPNs, is that addresses sent over the external network are constantly rotating. There are two benefits to configuring MT6D in this fashion: First, internal hosts can communicate without performance degradation. Second, network administrators can manage internal host activities, which would be otherwise obscured in a host-based MT6D implementation.

Test Configuration

We developed an MT6D prototype software implementation using the Python programming language. This initial version wasn't designed to maximize efficiency, but rather to prove the design's validity.

We used the integrated software configuration; Figure 3 illustrates the architecture for a single MT6D host. The internal NIC is virtually configured on each host to act as the interface between the operating system and MT6D. The physical NIC acts as the external NIC, transmitting MT6D encapsulated packets to and receiving them from the network. We installed the prototype software on two Dell Latitude D620 laptops, each containing an Intel Core Duo T2300 1.66-GHz processor with 2 Gbytes RAM running Ubuntu Linux 11.10. Both platforms contain a 10/100 Fast Ethernet NIC.

We conducted testing on a production IPv6 network. Virginia Tech has a fully functional IPv6 network, providing globally unique addresses through SLAAC to every wireless and wired node on the network. The production network provided us with results that account for the effects of actual network traffic on MT6D packets. Network traffic between the two MT6D-enabled platforms was routed through the core network, which routes traffic for more than 30,000 nodes.

Analysis of Results

Our goal was to demonstrate MT6D's functionality, so we tested MT6D's ability to successfully pass different traffic types. For ease of traffic analysis, we set a fixed address-rotation interval of 10 seconds. All testing was done with MT6D using encrypted tunnels, with 128-bit AES (Advanced Encryption Standard) encryption. To measure basic functionality, we sent 1,000 ping packets from the client to the server at a rate of one packet per second. To test MT6D under a high traffic volume of connectionless traffic, we sent a series of 10,000- and 50,000-packet ping floods from the client to the server. To test how MT6D handles connection-oriented traffic, we had one host, configured as a client, use HTTP over TCP to download files ranging from 500 Kbytes to 1 Gbyte from the other host configured as a server. We also examined how MT6D handles real-time traffic by testing two hosts' ability to communicate using voice over IPv6 (VoIPv6).

Table 1. Average transmission speed and percentage of retransmissions for connection-oriented network traffic using MT6D.

Network traffic type	Transmission speed (Mbytes/s)	Retransmissions (%)
TCP file transfer (500 Kbytes)	1.87	0
TCP file transfer (1 Mbyte)	1.96	0
TCP file transfer (10 Mbytes)	1.83	0.33
TCP file transfer (50 Mbytes)	1.86	0.44
TCP file transfer (500 Mbytes)	1.78	1.73
TCP file transfer (1 Gbyte)	1.79	1.80

Connectionless Ping Results

Our initial tests used a 1,000-packet ping. After 10 iterations, the average round-trip latency increase for an encrypted MT6D ping versus an unencrypted ping without MT6D was 5.7 milliseconds. The average packet loss was 0.28 percent. The increased latency was predominately caused by queuing during neighbor discovery after each address rotation.

Similar tests used ping flood to replicate a high traffic volume. For a 10,000- and 50,000-packet ping flood, the average round-trip latency increase was approximately 2.63 milliseconds. The percent packet loss was 0.02 percent. The reduced latency and packet loss occurred due to a decrease in the ratio of address changes to overall packets sent.

Connection-Oriented Results

To test connection-oriented traffic, we had one host download files from the other host. File sizes ranged from 500 Kbytes to 1 Gbyte. The average throughput for all the file downloads was just under 2 Mbytes per second. Table 1 illustrates the average download speeds as well as the percentage of retransmission that occurred during testing.

Real-Time Traffic Results

We tested MT6D's ability to handle real-time network traffic by setting up a VoIPv6 connection between the two test hosts. To establish the VoIPv6 connection, we used Mumble (www.mumble.com) because of its IPv6 support. In our tests, we set the voice quality to the maximum setting available, 96 Kbits per second. There was no noticeable delay between the two communicating hosts. The average latency during our testing was 3.34 milliseconds with 0.10 percent packet loss.

These results demonstrate that MT6D can successfully pass both connectionless and connection-oriented traffic. MT6D was able to rotate addresses without interrupting ongoing sessions between hosts using

MT6D and without requiring additional handshaking. Although some additional latency and packet loss occurred owing to the use of MT6D, it wasn't enough to significantly affect network performance. Even VoIPv6 communications weren't noticeably affected.

Limitations

MT6D still faces several limitations that potential users should keep in mind.

Attacks at Other Layers

Because MT6D is a network-layer defense, it doesn't prevent attacks from occurring at other layers. For example, MT6D can't prevent attackers from capturing MAC addresses because these attack types occur at the data-link layer. This isn't a major issue for two reasons: First, packets tunneled using MT6D aren't sent using the protected host's MAC address. However, the obscured MAC can't change as often as network addresses because changing a system's hardware address requires bringing down the adapter. Second, lower-layer attacks generally require attackers to be on the same network as the target host.

Network Protocol

MT6D is designed to operate only on an IPv6 network. Although MT6D could feasibly be redesigned to operate on an Internet Protocol version 4 (IPv4) network, there isn't enough unoccupied address space to facilitate it. Because MT6D "hops" between different addresses, employing it on a densely populated subnet would cause numerous collisions.

However, this isn't a concern on IPv6 subnets because even heavily populated subnets won't occupy much of the available address space in the foreseeable future. The likelihood of an address collision on an IPv6 subnet can be written as $P_c = h/2^{64}$, where P_c represents the probability of a collision and h represents the number of other hosts on the subnet.

Communication Paths

Including an unsupported communications path in MT6D could result in attackers learning a host's actual address. However, in environments requiring a high security level, users can disable the unsupported communications path to mitigate this limitation. Disabling the unsupported communications path would be useful when MT6D is used on a classified government network that doesn't allow communication with the public Internet.

Scope of Protection

MT6D operates similarly to VPN technologies in that both endpoints require MT6D to communicate securely. This means MT6D can't feasibly protect the public Internet. However, this doesn't mean that MT6D can't provide secure communications with servers. Even public servers can be configured with an MT6D channel to handle nodes that use MT6D as well as those that don't.

It's worth noting that protection of the public Internet was never MT6D's goal. MT6D meets needs similar to those of Internet Protocol Security—neither was designed to be a public infrastructure scheme.

Network Administration

Implementing MT6D on individual hosts could hamper a network administrator's ability to monitor those hosts' behaviors. Applying MT6D at the network border could protect internal users while still letting administrators monitor internal traffic. However, with any security mechanism, there are trade-offs that administrators must weigh. For example, users might choose to use MT6D regardless of whether network administrators allow it. This is particularly true of malicious users. It's important that administrators are aware of technologies like MT6D and how they can affect networks.

Certain applications would greatly benefit from our technology. One potential application of MT6D is securing sensor networks, such as the smart grid. Attacks against sensors could deny critical information communications or expose sensitive information. MT6D prevents attackers from locating and subsequently targeting sensors. MT6D can also secure peripherals, such as printers and faxes. Typically, these devices aren't protected and are thus easy targets for attackers. MT6D can transparently protect these devices without requiring expensive modifications to the peripheral. Our technology can also provide secure communications for military or intelligence agents, corporate entities, or e-commerce. VPNs, which are susceptible to DoS at the endpoints, can benefit from MT6D as well: a VPN using MT6D will have a moving endpoint that attackers statistically can't target.

The next step in MT6D development is to optimize the design for inclusion on handheld devices and tablets. With migration to IPv6 on the horizon, it's important to prepare the next generation of IPv6 security solutions. MT6D is one of these solutions. ■

References

1. T. Narten, R. Draves, and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6," RFC 4941, Internet Eng. Task Force, Sept. 2007.
2. R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291, Internet Eng. Task Force, Feb. 2006.
3. T. Narten et al., "Neighbor Discovery for IP version 6 (IPv6)," RFC 4861, Internet Eng. Task Force, Sept. 2007.

4. S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862, Internet Eng. Task Force, Sept. 2007.
5. S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Internet Eng. Task Force, Dec. 1998.

Matthew Dunlop recently completed his doctoral degree in computer engineering at Virginia Tech. His research interests include IPv6 and network security. Dunlop has an MS in electrical and computer engineering from Carnegie Mellon University and an MS in engineering management from the University of Missouri at Rolla. He's a member of IEEE. Contact him at dunlop@vt.edu.

Stephen Groat is a doctoral student at Virginia Tech in computer engineering. His research interests include networking and network security. Groat has an MS in computer engineering from Virginia Tech. Contact him at sgroat@vt.edu.

William Urbanski is a network security advisor at Dell SecureWorks. His research interests include information security and online criminality. Urbanski has a BS in computer science from the University of Georgia. Contact him at will.urbanski@gmail.com.

Randy Marchany is the Virginia Tech information technology security officer and director of the Virginia Tech IT Security Lab. His research interests include Internet security vulnerabilities and denial-of-service attacks. Marchany has an MS in electrical engineering from Virginia Tech. Contact him at marchany@vt.edu.

Joseph Tront is a computer engineering and electronics professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. His research interests include integrated circuit design and testing, bio-microelectronic sensors, fault-tolerant autonomous computers, and space radiation's effects on integrated circuits. Tront has a PhD in electrical engineering from the State University of New York at Buffalo. He's a senior member of IEEE CS, the Special Interest Group on Design Automation, the ACM, and the Academy of Teaching Excellence. Contact him at jgtront@vt.edu.

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.