Lessons from DEFCON 2016 – Bypassing Captive Portals

August 26, 2016 By Marc Laliberte

At this year's DEFCON, security engineer Grant Bugher gave a presentation on bypassing captive portals. Being DEFCON, the talk was geared towards individuals attempting to circumvent the portals, but IT professionals can use the information to help prevent this kind of behavior.

If you are unfamiliar with the technology, captive portals force users to authenticate or agree to an acceptable use policy (AUP) before connecting to the internet from a guest network. If you've ever connected to a hotel's Wi-Fi, you've probably been directed to a captive portal to input your room number and accept an AUP. Most captive portals work by intercepting the web browser's attempted connections and injecting a redirect, forcing the client to the captive portal web page. After the user completes the portal requirements, their connections are no longer redirected and they are allowed access to the internet.

Bugher's bypass methods take advantage of the fact that, while captive portals should prevent normal internet access until completed, most organizations don't block all outbound ports while waiting for a user to complete portal requirements. For instance, in order for the client's browser to be redirected to the portal, it first needs to attempt an HTTP or HTTPS connection to a web page. For this connection to succeed, the browser needs to resolve the destination web page's domain name. Most organizations keep

outbound DNS open on UDP port 53 to allow for this name resolution. Some organizations allow other ports out too for various reasons, such as SSH on TCP port 22 or web caching proxies on TCP port 3128.

Before diving in to the bypass methods, Bugher noted an important requirement during his presentation. In order for a client to bypass the captive portal, they need prior preparation. The client must have an external server pre-configured to act as an endpoint for their different bypass methods. I won't go into the details here on how to configure the server but for the rest of this article we can assume that the client has one available, whether it be their own or a public server.

Like any network security attack, the client's first step is to perform reconnaissance. After connecting to the guest network, a mischievous client will typically run a scan to find what ports are open (allowed) through the network's gateway. The client is looking for open ports matching the ones mentioned earlier, TCP/3128 for a web proxy, TCP/22 for SSH, or UDP/53 for DNS.

Once the client has identified ports which are allowed out without captive portal acceptance, they can start trying different bypass methods. If TCP/3128 is open, the client can configure their web browser to send connections through a proxy service on their pre-configured server. Proxy services are often legitimately used on corporate networks to cache responses from frequently visited websites, reducing outbound bandwidth from the network. A client attempting to circumvent a captive portal can proxy their requests through an external server, bypassing any restrictions on the default HTTP and HTTPS ports. IT administrators can protect against this bypass by simply restricting outbound access on TCP/3128 from networks protected by a captive portal.

Outbound proxy access on TCP/3128 is often blocked by network administrators. SSH (TCP/22) on the other hand is sometimes overlooked. SSH is normally used to securely access the command line of a remote server. Most SSH clients and servers have another feature though, which allows them to open an encrypted tunnel between the two hosts. This tunnel can be used just like the earlier bypass method to proxy outbound traffic, bypassing the default HTTP and HTTPS port restrictions. If the client finds in their network scan that outbound TCP/22 is allowed, they can simply create an SSH tunnel to their external server and then configure their browser to proxy connections through the tunnel. IT administrators can protect against this bypass by restricting outbound access on TCP/22, a port which shouldn't be required on captive portal networks.

Let's assume that the network administrator has done his or her due diligence by blocking all outbound ports except for DNS (to allow the initial browser connection). The client's port scan only returns UDP/53 as an allowed outbound port, meaning their web proxy and SSH tunnel bypass methods are not an option. The network admin should feel safe now right? Unfortunately, no. There exists an application called iodine which facilitates tunneling over DNS. With iodine set up on both the client's remote server and the client itself, the client can tunnel any and all traffic out over the DNS port. As an IT administrator, this bypass is more difficult to prevent. You can choose to block outbound DNS from the captive portal network, but the clients still need to be able to resolve domain names in order to hit the portal redirect. You could restrict outbound DNS to only be allowed from an internal DNS server, and then hand out that

internal DNS server's IP address to connecting clients using DHCP. Though this method requires the extra overhead of maintaining an internal DNS server and causes problems for any client with hard-coded DNS server addresses. WatchGuard customers can use the DNS Proxy to detect and block the abnormally large DNS queries used by DNS tunneling applications like iodine.

It's clear from Bugher's talk that IT and network administrators need to pay attention to which protocols they allow out before clients accept the captive portal requirements. Captive portals are critical whether they be for restricting access to only allowed guests, limiting liability with acceptable use policies, or obtaining revenue to offset the cost of network access. Prior to portal completion, there is little reason to allow anything but restricted DNS out of your network. By taking the time to secure your guest networks, you limit the opportunities for less-than-honest clients to circumvent your other connection requirements.
– Marc Laliberte

Filed Under: Editorial Articles
Tagged With: Hacking

Comments

Nobody says
August 18, 2017 at 12:41 pm

Last I checked, the WG DNS Proxy does not support DNSSec, since the packet sizes are larger than non secure DNS queries and it rejects them. This caused problems for an organization I know of, and they had to use a packet filter for DNS instead. If that's been fixed, that would be great to know.

Reply