Attacking Tor at the Application Layer

Gregory Fleischer (gfleischer@gmail.com)

DRAFT SLIDES

Updated slides will be provided after the talk.

Most importantly, the updates will include links to permanent location for all online demos.

Introduction

Introduction

- What this talk is about
  - identifying Tor web traffic
  - fingerprinting users
  - attacking at the application layers
- There is a heavy emphasis on the client-side, web browsers attacks and JavaScript

Introduction

- What this talk is NOT about
  - passive monitoring at exit nodes
  - network attacks against path selection
  - using application functionality to increase the likelihood of network attacks
  - breaking SSL

Introduction

- Software tested
  - The Tor Browser Bundle
  - Vidalia Bundle for Windows
  - Vidalia Bundle for Mac OS X
  - Firefox 2, Firefox 3.0 and Firefox 3.5 RC
  - Torbutton

Background

Background

- Brief overview of Tor
  - free software developed by The Tor Project
  - uses onion routing and encryption to provide network anonymity
  - can be used to circumvent local ISP surveillance and network blocking
  - can also be used to hide originating IP address from remote servers

Background

- Adversary model at the application layer
  - normal browsing, without Tor
    - local ISP
    - remote server

Background

- Adversary model when using Tor
    - remote server
    - exit nodes
    - remote server's ISP
    - exit node's ISP

Background

- Exit nodes as attack points
  - can inject arbitrary content into non-encrypted responses
  - but can also modify or replace non-encrypted requests
  - Tor users make attractive targets because they are self-selecting

Background

- Applications and Tor
  - only applications that are proxy aware can use Tor properly
  - network clients that don't know about Tor may leak the user's original IP address
  - user's IP address may also leak for applications that don't use proxy for name lookups

Background

- DNS requests over Tor
  - DNS queries are resolved by remote Tor node
  - resolution can be slow, so queries are cached locally for a minimum of 60 seconds regardless of TTL
  - makes traditional DNS rebinding attacks difficult

Background

- Application stack for Tor web surfing
  - web browser (most likely Firefox)
  - local HTTP proxy (Privoxy or Polipo)
  - Tor client as SOCKS proxy
  - remote web server

Identifying

Identifying

- Remote sites can easily detect Tor users' web traffic as a group
  - the list of Tor exit nodes is well known
  - for example, TorBulkExitList can be used to retrieve a list of all exit nodes
  - there are some alternative methods

Identifying

- Examine IP based on cached-descriptors
  - run a Tor client and track IP addresses
  - simple, passive
  - may be limited, not all exit IP addresses are published

Identifying

- TorDNSEL
  - DNS based look-up of exit node/port combination
  - uses active testing of exit nodes to determine actual exit IP addresses
  - used by https://check.torproject.org/

Identifying

- Request Tor specific HTML content
    - HTML request via: iframe, image, link, JavaScript, etc.
    - use hidden service (.onion)
    - use exit node syntax (.exit)

Identifying

- Problems with requesting Tor specific content
  - depends on resources outside of your control
  - there is an associated infrastructure cost
  - slow, may not always work
  - other options?

Identifying

- Use .noconnect syntax
    - internal Tor host name suffix that immediately closes connection
    - compare timing of resolving “example.example” and “example.noconnect”
    - can be performed in client-side script

Fingerprinting

Fingerprinting

- Browser fingerprinting using active testing
- Firefox and Torbutton
    - recommended by The Tor Project along with Torbutton
    - Torbutton hides user agent through setting modifications
    - Torbutton also disables plugins by default
- Other browsers not tested

Fingerprinting

- Anonymity set reductions through Firefox
- Firefox browser behavior changes
  - examine functionality differences between versions and platforms
  - iterate Components.interfaces
  - can “unmask” real user-agent information

Fingerprinting

- Look for installed/enabled Firefox add-ons
  - add-on content may remotely loadable if “contentaccessible=yes”
  - add-on may contain XPCOM components which are enumerable via Components.interfacesByID

Fingerprinting

- Generate and examine browser errors
  - some exception messages are localized and could be used to determine language
  - internal exceptions may leak system information
  - example, get local browser install location:
    - (new BrowserFeedWriter()).close()

Fingerprinting

- Enumerate Windows COM objects
  - Firefox exposes GeckoActiveXObject
  - can be used to load ActiveX objects
  - only whitelisted components are allowed
  - but different errors are generated based on whether the ProgID is located

Fingerprinting

- More anonymity set reductions through local proxies
  - Vidalia Bundle - uses Privoxy as proxy
  - Tor Browser Bundle - uses Polipo
  - examine proxy behaviors and content

Fingerprinting

- Local proxies may export specific content
  - RSnake demonstrated detecting Privoxy using Privoxy specific CSS
  - http://ha.ckers.org/weird/privoxy-test.html
  - circa 2006, but still works

Fingerprinting

- Local proxies may exhibit detectable behavior
  - Polipo filters a specific set of headers: “from”, “accept-language”, “x-pad”, “link”
  - can construct XMLHttpRequest requests that contain these headers and test for the filtering

Fingerprinting

- Exploit application interactions and defects
  - generate proxy errors using XMLHttpRequest
  - responses may include proxy version, hostname, local time and timezone
  - need to maintain same-origin to read response

Fingerprinting

- Use browser defects and edge cases
    - generate POST request without length
    - IPv6 host name: http://[example.com]/
    - malformed authority: http://x:@example.com/
    - requests with bogus HTTP methods: "* / HTTP/1.0"

Fingerprinting

- Cause protocol errors from the server
  - serve valid content, but drop CONNECT requests
  - return nonsensical or invalid HTTP headers
  - anything in RFC 2616 that is specified as “MUST” is probably fair game

Attacking

Attacking

- Historical attacks of note
  - Practical Onion Hacking - FortConsult
  - HD Moore's Torment & decloak.net
  - ControlPort exploitation

Attacking

- ControlPort exploitation - Summer 2007
  - abused cross-protocol request to Tor ControlPort (localhost:9051)
  - Tor allowed multiple attempts to send AUTHENTICATE directive
  - attack via web page form POST with encoding of ‘multipart/form-data’
  - fixed by only allowing a single attempt

Attacking

- What else was big in Summer 2007?
- DNS rebinding:
  - Java applets could use ‘document.domain’ bypass to open raw TCP sockets
  - only protection was to set ControlPort password

Attacking

- Torbutton protections against scripts
  - restricts dangerous protocols (e.g., “resource://”, “chrome://”, “file://”)
  - masks some identifying properties
  - some of these are implemented JavaScript
  - but what’s done in JavaScript can be undone in JavaScript

Attacking

- Defeating Torbutton protections
  - use the "delete" operator or prototypes to access original objects -- mostly fixed
  - use XPCNativeWrapper to get reference to protected, original methods
  - use Components.lookupMethod to retrieve internally wrapped native method

Attacking

- Abusing active content and plugins
  - active content and plugins are dangerous
  - some people want to (or need to) use them
  - can sometimes force load of plugin content by directly including it:
    - <iframe src=“http://example.com/attack.swf”>

Attacking

- Example of Firefox 2 exploit
  - Torbutton behaves differently if it is set to Disabled when the browser is launched
  - by using nested protocol handlers, the content is loaded before Torbutton can block it
  - jar:view-source:http://example.com/x.jar!/attack.html
  - x.jar contains attack.html and attack.swf
  - attack.html loads attack.swf via iframe

Attacking

- Multiple browser attacks
  - The Tor Project suggests using two browsers; one for Tor, one for unsafe
  - the unsafe browser probably doesn’t have many of the restrictions or protections
  - content from the unsafe browser can potentially target local Tor resources
  - for example, use Java same origin bypass

Attacking

- External protocol handlers can launch applications that aren’t proxy aware
  - Windows telnet: protocol handler
  - Windows ldap: protocol handler
  - these may be automatically invoked unless the “Always ask” option is set

Attacking

- Add-ons may launch external programs
  - Microsoft .NET Framework Assistant
  - installed as system extension to support ClickOnce deployment
  - monitored for content that was returned with Content-Type: application/x-ms-application
  - re-requests content from external program, leaking the user's original IP address

Attacking

- Attacking saved content downloaded via Tor
  - any unencrypted content is vulnerable
  - any content downloaded over HTTP can be modified to be malicious
  - trojan content may wait to phone home
  - even “safe” content may not be so safe

Attacking

- Locally saved HTML content is not safe
  - any HTML content can be forced to be locally saved by specifying "Content-disposition: attachment"
  - may be saved with an HTML extension and opened later from the web browser
  - the "Open" option opens a local temporary file
  - in Firefox 2, local HTML can read any file

Attacking

- Vidalia bundles with Vidalia version 0.0.16
    - the ControlPort password was saved in clear text (even for random values)
    - locally saved HTML files could read this
    - if Java was enabled, same origin bypass could be used to authenticate to ControlPort using the password

Attacking

- Additional blended threats are possible
  - if plugin content is allowed, a locally saved file may be able to bypass restrictions
  - remote attacker sites can opt-in to allow plugin content to connect back (e.g., crossdomain.xml)
  - local HTML could use jar: protocol to load additional active content

Attacking

- New "Toggle" attacks against Torbutton
    - attempt to transition state information when user toggles Torbutton
    - use JavaScript setInterval as a timer
    - remotely detecting Torbutton banned ports
    - use returnValue from showModalDialog to transfer content between windows

Conclusions

Conclusions

- There is a large application attack surface
  - there are many attackable components between the user web browser, local HTTP proxy, Tor client and remote web server
  - new attack techniques are researched and refined all the time
  - many common web application attacks can be repurposed to attack Tor users

Conclusions

- Consider using an isolated environment
  - run web browser and Tor inside a VM
  - only install the software you need
  - create a restrictive egress firewall
  - only exit traffic that goes over Tor

Conclusions

- Remember safe web browsing habits
  - consider using isolated identities, and don’t mix and match user accounts
  - don’t trust content that was downloaded over unencrypted channels

Conclusions

- References:
  - https://www.torproject.org/
  - https://git.torproject.org/checkout/tor/master/doc/spec/address-spec.txt
  - https://www.torproject.org/torbutton/design/
  - http://exitlist.torproject.org/
  - http://www.ietf.org/rfc/rfc2616.txt
  - http://releases.mozilla.org/
  - https://developer.mozilla.org/En/DOM/Window.showModalDialog
  - https://developer.mozilla.org/En/Windows_Media_in_Netscape
  - https://bugzilla.mozilla.org/show_bug.cgi?id=412945
  - http://ha.ckers.org/blog/20061220/detecting-privoxy-part-ii/
  - http://www.fortconsult.net/images/pdf/Practical_Onion_Hacking.pdf
  - http://archives.seul.org/or/talk/Mar-2007/msg00131.html
  - http://decloak.net/

End