Veiled
A Browser Darknet

Matt Wood & Billy Hoffman

matt.wood@hp.com billy.hoffman@hp.com

Web Security Research Group -- HP Software

What is a Darknet?

A private network where users can freely exchange ideas and content. Darknets typically have additional features like strong encryption, digital identities, or storage systems

Innovation and Turmoil Abound!

New Set of Browser Wars

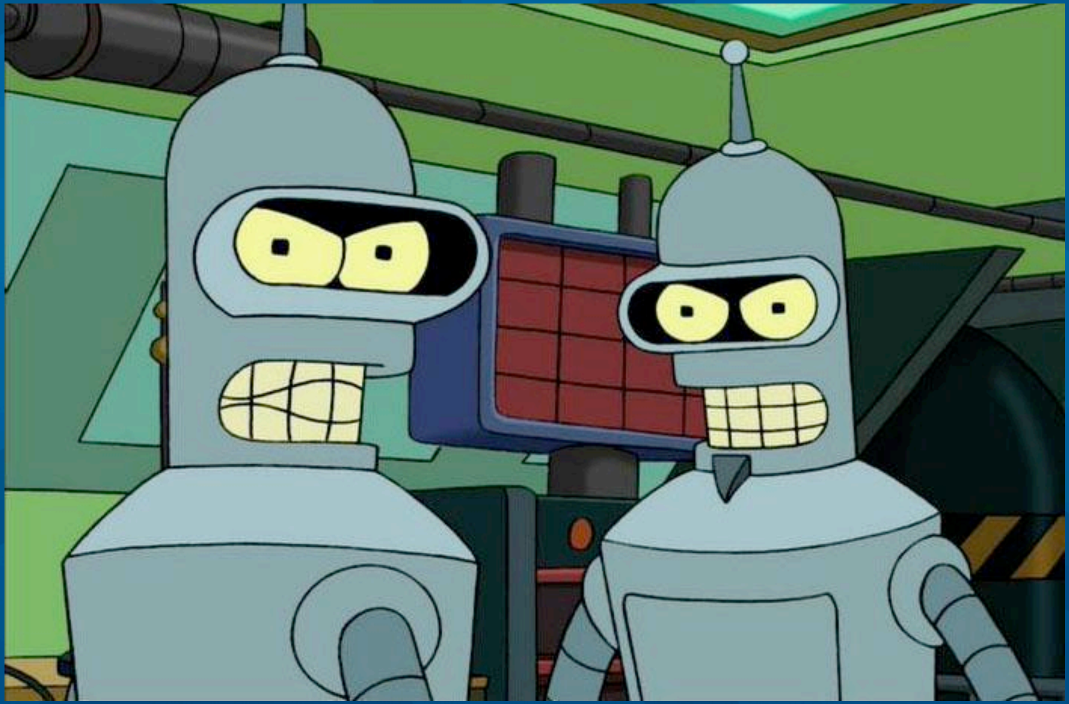
Desktop App <-> Web App

Server/Client <-> Peer-2-Peer

Browser Platform + Traditional Darknet + ??? = Profit

Original Darknets

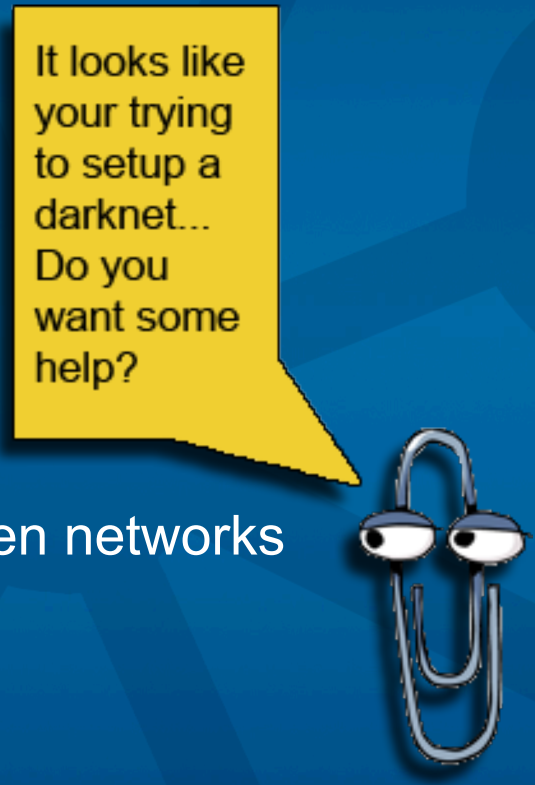
- Sneaknets
  - Yokel with floppy
- Freenet (1999)
  - Strong Crypto
  - Shared Storage
- WASTE (2000)
  - Private/sharing
- Gnutella(~2000)
  - P2P qualities
- Clones (2001-)



Tor is *not* a Darknet!

Barriers to Darknet Adoption

- Technical Barriers
  - Installing/Configuring
    - Firewall
    - NAT
  - Not for Joe the Plumber
- Social Barriers
  - People are unaware they exist
  - Freenet/Gnutella/Kazaa are basically open networks
    - Difficult to set-up your own
    - Creating your own “scene”


It looks like your trying to setup a darknet... Do you want some help?

Web Ecosystem

- HTML 5 Features
  - Browser Storage
  - CORS/XDomainRequest
- JavaScript Advances
  - V8/TraceMonkey
- Lots of high quality JS libraries
  - UI/DOM
  - AES/RSA
- Ubiquitousnesslyosity
  - Everyone has it
  - Everyone can use it

A Browser Based Darknet

- Properties we want
  - Distributed Redundant File Storage
  - Some anonymity
  - Web in the Web
  - Communication entirely over HTTP
- Differentiating Properties
  - Zero-footprint Install
  - Web Clients are the new Web Server
    - P2P on top of HTTP
  - Simple to create/destroy/join
  - Focused on small to medium sized network/mesh

WHY?

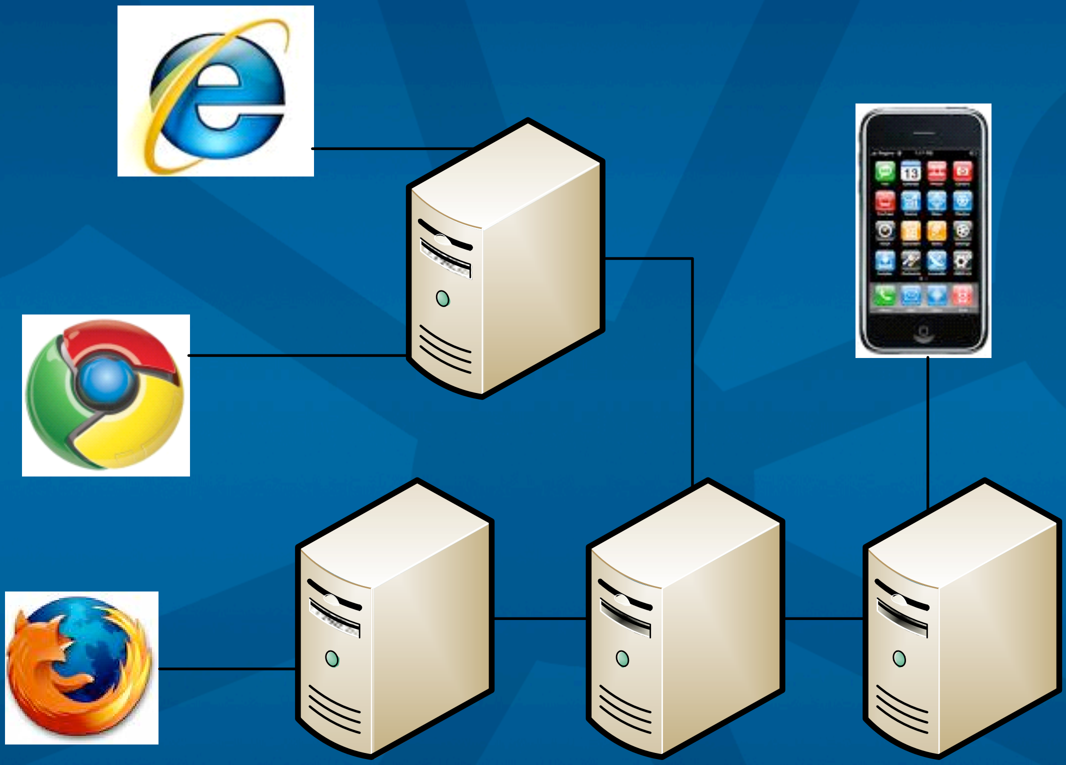
- Barriers are Bad!
- Removing Barriers is Good!
- Fewer Barriers allows more Participation!
- More participation allows Innovation!

Veiled : Agenda

- Architecture
- Demo
  - Tech Overview
- Features
  - Private Chat
  - Redundant Distributed File Storage
  - Web in the Web
  - Distributed JS Computation
  - Server Failover
- Threat Analysis
- Next Steps/New Tech

High Level Architecture

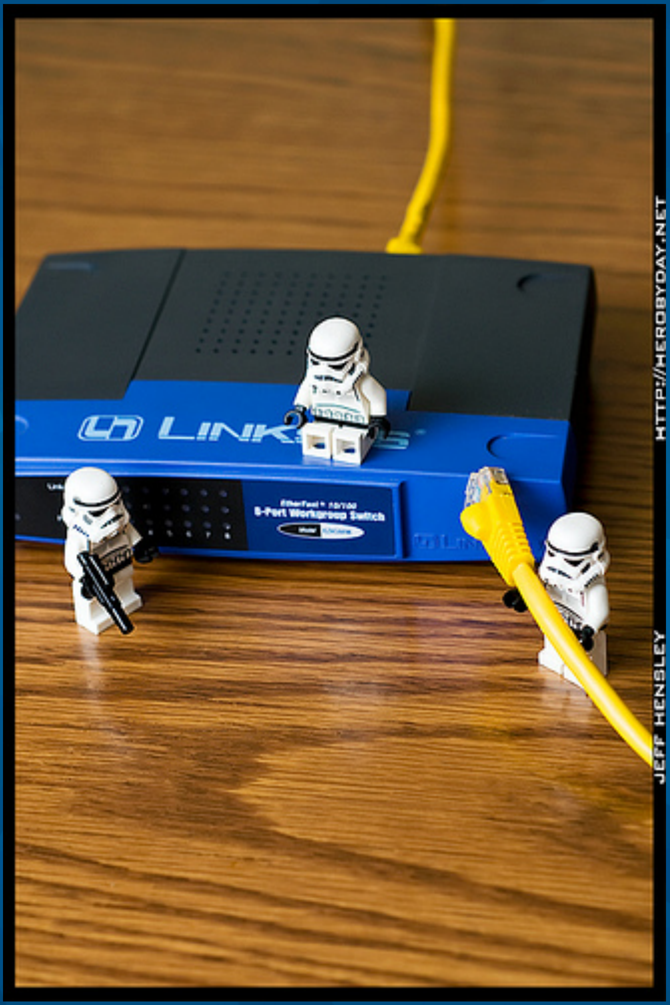
- Router is single server script file
  - Routers can peer
- Client is Pure JS and HTML
  - Clients connect to Routers
- Messages can traverse whole network



Veiled Demo

Veiled Router

- Single Dynamic Page
- Provides COMET connection to Client
- Provides Ajax Targets for Client
- Provides Peering hooks for other Routers
- Provides transient storage (server memory)

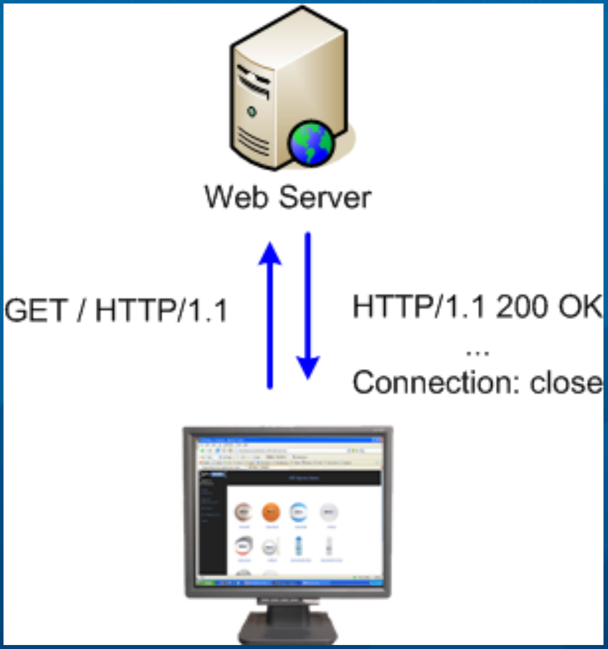


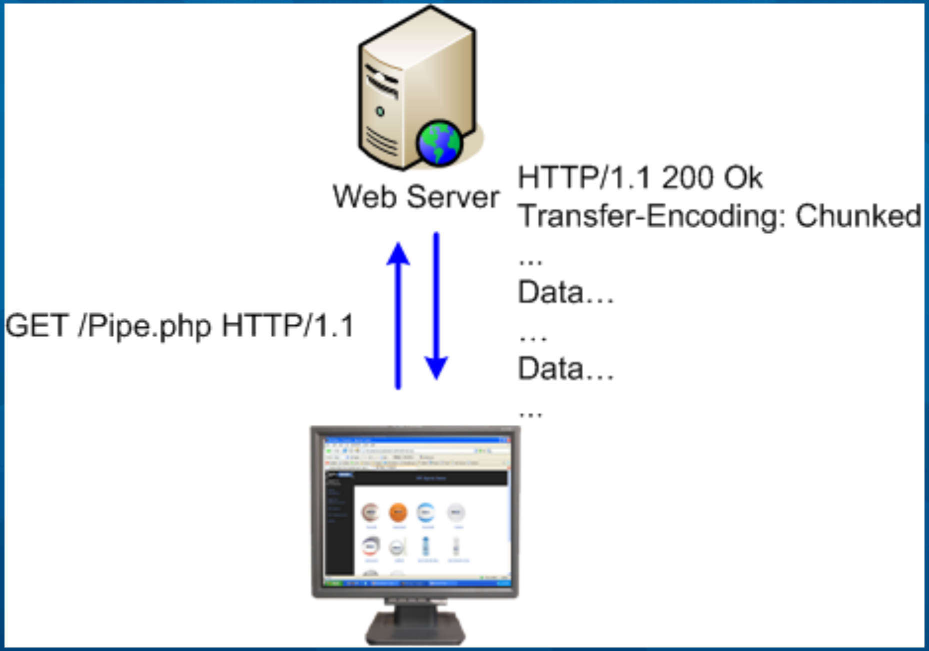
Veiled Client

- Pure JS/HTML
- Content served by a Router
- Provide callbacks for COMET data pipe
- Contains API to use browser storage
  - Whatwg_db/LocalStorage/Cookies/GlobalStorage
- Contains JS Encryption Routines
- UI

COMET?

- Traditional HTTP “pulls” data
- COMET is a hack to make HTTP push possible
  - faster than long polling for lots of messages





Messaging in Veiled

- Client-Client Communication
- Client-Router Communication
- Router-Router Communication

Client-Client Messaging

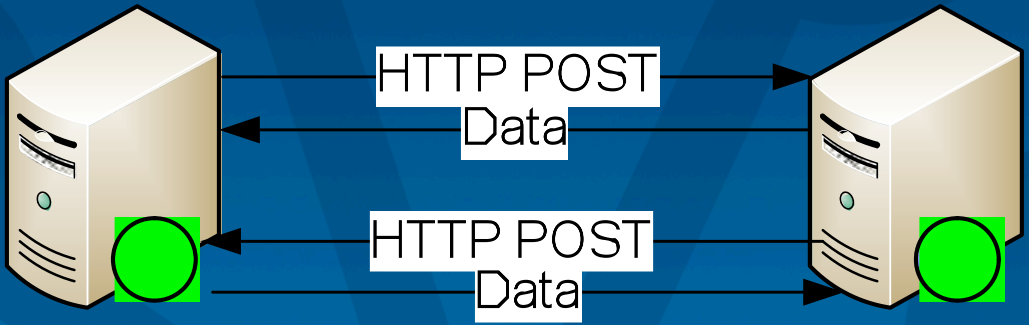
- Two Types of Messages
  - Multicast
  - Routed
- Occurs above HTTP Layer
- Message Format
  - Type
  - Action
  - Origin ID
  - Target ID
  - Data
  - Distance
  - Unique ID

Router-Client Communication

- Client initiates COMET connection
  - Hidden iframe
    - Easy
    - Messages streamed to client via individual <script> tag function calls
    - Higher bandwidth than long-polling
  - Times out after 2-5 minutes, refreshes
- Client uses AJAX to forward local messages/events to Router
- Can Use HTTP Auth
- Can Use SSL

Router-Router Peering

- HTTP/S Connection
  - Comet-y
  - No need for JS Tricks
  - Uses JSON for interop
  - PHP's fsockopen



- First connection contains connect back information
- Both Servers need to be mutually accessible
- Performs Routing
- Can be setup with HTTP Auth
  - to prevent vagabonds

Routing with Modified AODV

- AODV
  - Ad-hoc On-demand Distance Vector
  - Used for mesh networks
- Modified specifically to take advantage of protocol between clients
- Why?
  - Reduces Traffic
  - Minimizes clients receiving traffic not meant for them

Veiled Features

- Global Chat
  - More of a debugging mechanism
- Private Chat
- Redundant Distributed File Storage
- Web in the Web
- Distributed JavaScript Jobs
- Server Failover

Private Chat

- RSA Key Exchange
  - Keys generated by OpenSSL
  - Uses PidCrypt JS Library
  - Exchange AES 256 Key using RSA
- AES + CBC
  - AES Key generated from
    - Hash(RSA Priv Key/Time/Domain)
- Completely Encrypted on the Client
- Possible MITM
  - Verify public keys

Private Chat Protocol

- Request Peers from Darknet
  - Can opt-out
- Receive Client Aliases and Public Keys on Darknet
- Start Chat Session with Client
  - Initiator generates AES Key
  - Encrypt with Remote Client’s public key
  - Send
- AES Key Exchanged
- Begin chat encrypting with AES

Redundant Distributed File Storage

- Goals
  - Survive Clients leaving the network
  - Secure Upload/Download of content
  - Utilize browser storage
- Why
- How
- Challenges
  - JS has no access to local files
  - Two Options
    - Trust Router to distribute slices
    - Use Flash/Java to read local file and provide to JS

RDFS Protocol

- Upload
  - Select file and Submit HTML form
  - Router slices into 1k chunks
  - Multicast request for storage on darknet
  - Wait for slices to be “registered”
  - Send registered clients routed data packed
- Download
  - Multicast file identifier for retrieval
  - Client’s check if their data store contains file identifier
  - Send routed data packet if found

Web in the Web

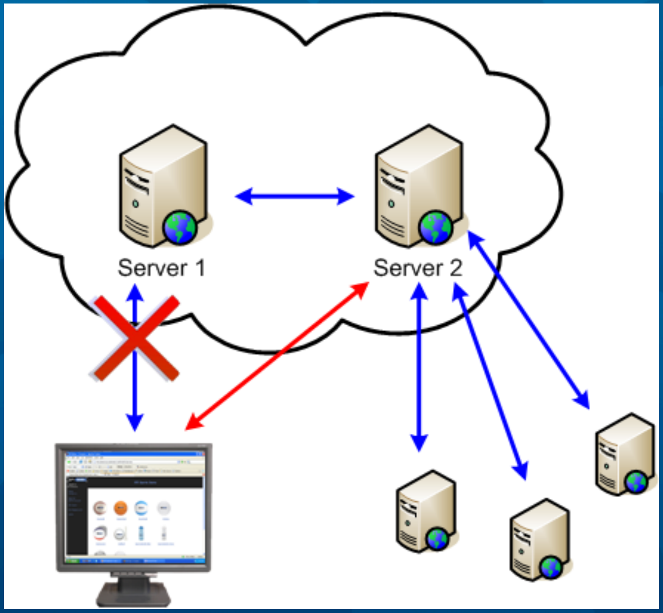
- Builds on top of File Distribution
- Retrieve Files from Magnet Hashes
- JavaScript API to Support
  - Embedded Images
  - Embedded (i)Frames
  - Rewriting Links to Magnet Hash

Distributed JS Computation

- Distribute JavaScript Jobs to Clients
- Inspired by JavaScript
- Provide Client API for:
  - Receiving jobs
  - Reporting results
- Challenges
  - Dangerous JavaScript/XSS
  - Threading/Blocking the UI
    - Worker threads (HTML5/Gears)
  - Execute jobs in sandbox
    - Gareth Hayes: JSReg?
    - Google’s caja too much

Server Failover

- Router Peering
  - Publicize connect-back details to local clients
  - Inform clients if peer goes down
- If Client COMET connection goes down
  - Retry
  - Connect to router peer



Challenges

- Debugging a “hidden” php connection blows!
  - COMET, distributes PHP files, “threading,” multiple clients
  - Pretty much left with printf() style debugging!
- Shared Memory Locking/Threading
- Over Reliance on Router
  - “Untrusted” Router
- Local Storage vs Global Storage
  - Domain restrictions

Threading in PHP

- Each connection is a “thread”
- Use flock(windows) or semaphore (sys-v) for locking
- Shared Memory message queue
  - In transient Memory
  - PHP’s shmop

Veiled Threat Analysis

Veiled External Threats

- Malicious Client
  - Disrupt/Inject faulty communications
    - HTTPS and HTTP Auth can defuse this mostly
    - Autodestruct network – new one needed
- Malicious Router Process
  - Rogue PHP script (since all are run by apache)
  - Modify Shared Memory
    - Alter Message Queue
    - Remove Messages
  - Not sure if there is a better was to secure shared mem…
- MITM
  - Mitigated with use of HTTPS

Veiled Internal Threats

- Malicious Client
  - Advertise false routes by sending spoofed packets
  - Saturate Network with Multicast Traffic
  - Send Bogus File slices during retrieval
- Malicious Router
  - Can MITM Private Chat RSA Key Exchange
  - Compromise Clients IP’s connected to it

Advances/Next Steps

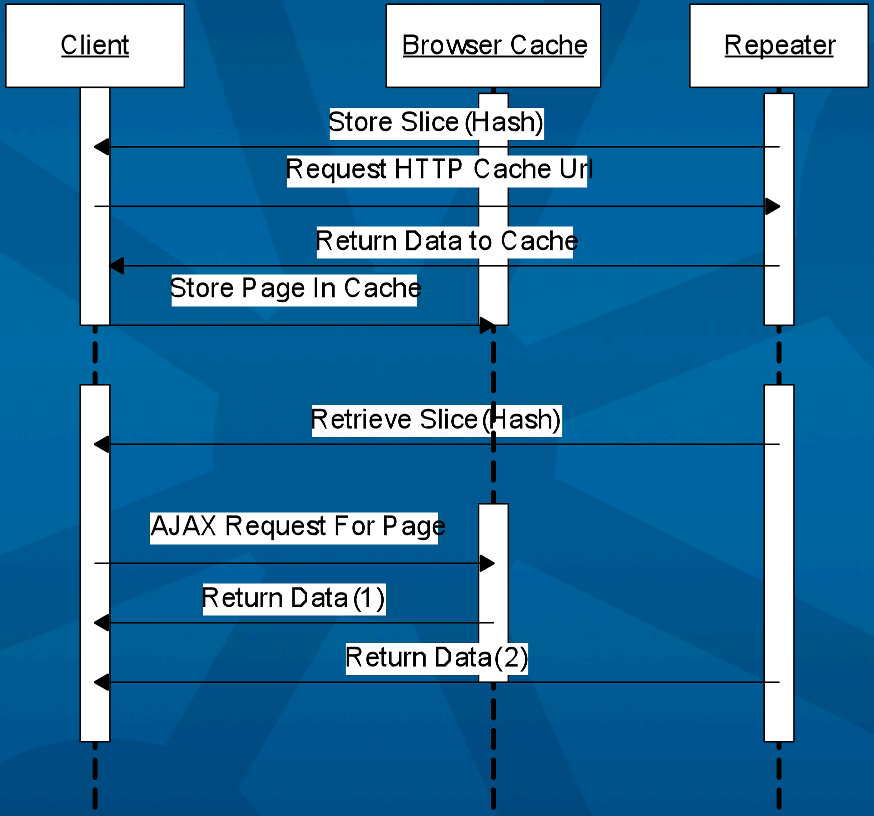
- NAT Busting
- File Storage with Browser Cache
- Using Clients as Routers
- Persistent-XSS as Com Port
  - Very low bandwidth
- Multiple Client-Router connections with CORS/ XDomainRequest
- Using Completely Untrusted/Public Routers
- Cloud Based Routers

NAT Busting with HTTP Request

- Server Behind NAT
  - Connection #1
    - Initiate connection to Remote
    - Send HTTP Request
    - Parse Response as it Arrives for Data
  - Connection #2
    - Initiate connection to Remote
    - Send “long” HTTP Request
    - Remote parses Request as it Arrives
- Vanilla PHP can’t inspect incomplete requests
  - Perl has this built in

File Storage with Browser Cache

- Use Browser Caching to Store File Slices
- Storage
  - Make hash from file hash and slice #
  - Router serves up page, client caches it
- Retrieval
  - Make hash from file hash
  - Ajax Request to router
    - Cached Response
    - Server Response



Using Clients as Routers

- Listen for messages on two routers
- Mediate Requests/Responses between them
- Benefits
  - Link inaccessible routers with a client
- Cons
  - Requires constant browser session/tab
  - Easier to MITM a network

Persistent-XSS as Shared Storage/Queue

- Use Persistent XSS's on the internet as storage.
  - Hundreds of online notepads, lots are vulnerable to pxss
- Decentralizes Network Further
- How?
  - Create JavaScript API to abstract PXSS as storage device
  - Use iframe communication from pxss to local window
  - JSONP if possible

Multiple Client-Router Connections

- Using CORS/XDomainRequest
- Benefits
  - Redundant connection to Darknet
  - Increased bandwidth
  - Stronger connected Mesh
- Challenges
  - Reduce/Identify duplicate traffic

Others…

- Using Completely Untrusted/Public Routers
  - Use strong encryption on top of all messages
  - However still allow routing somehow
- Cloud Based Router
  - Google App Engine Router
- One Way Communication Darknet

Sorry Wikileaks!

- Wikileaks is an amazing valuable service
- Made misinformed comments about system properties
- Wikileaks created their infrastructure from OS to Web Server to avoid the collection of any user data
- There is nothing to subpena
- You guys rock!

Ask Questions!

- Emails
  - Matt Wood -- matt.wood@hp.com
  - Billy Hoffman -- billy.hoffman@hp.com
- Twitterlicious
  - http://twitter.com/HP_AppSecurity