CERIAS

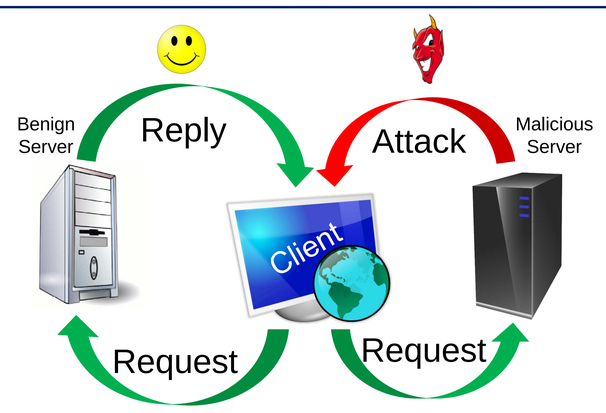
the center for education and research in information assurance and security

# Evading Client Honeypots

Jason Ortiz, Ankur Chakraborty, Pascal Meunier

## Client-side Attacks

- Client makes a request to a malicious or compromised server
- Server acts to infect the client, targeting some of many vulnerabilities in applications
- Increasing number of exploits due to better server-side security
- Large number of opportunities (apps, plug-ins…) are vulnerable to exploits or attacks




## Client Honeypots

- Client honeypots have detected and analyzed thousands of attacks
- Easily detect drive-by downloads and browser or application exploits
- Check of system changes (registry changes, process creation/termination)
- Actively seek to be attacked opposed to traditional honeypots

## Targeting Real Users by Hiding Malicious Intent

We observed that malicious websites employ social engineering to convince users of legitimate interaction.

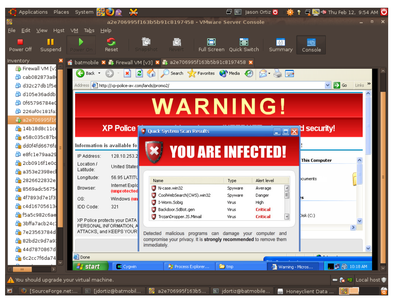
**Methodology**

We surveyed approximately 5000 websites using Honeyclient and manual navigation

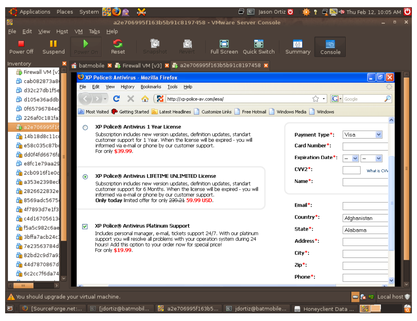
**Observations**

We found 43 attacks, many of them using a form of social engineering to gain user's trust

**Example** Rogue Applications




- Application appears as an *antivirus* software
- It is actually malware (usually spyware)
- A forged thread report informs the user their computer is infected with malware
- User is tricked into running a 'scan' of their system to find the infections

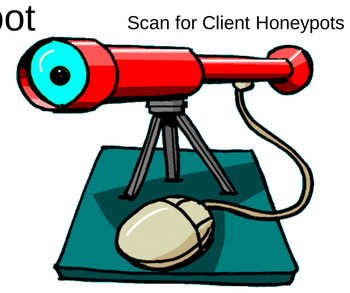


- 'Scan' finds hundreds of problems
- Application informs the user they must register and purchase the full version of the software to remove the threats
- User effectively pays for malware *and* disclosure of personal info

## Evading Honeypots by Hiding Malicious Content

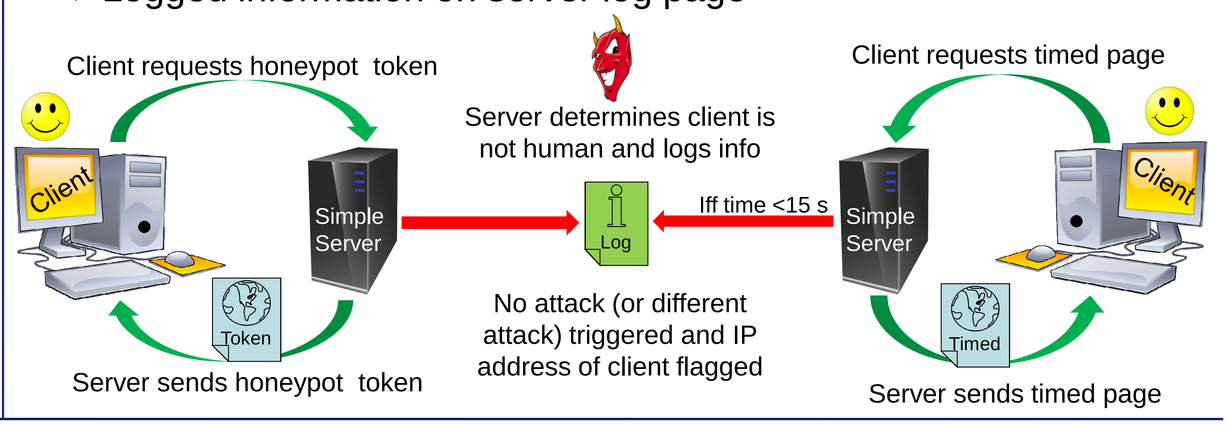
We hypothesize that malicious websites could employ several techniques designed to evade client honeypots.

- The client honeypot is unaware of the *existence* of the content
- Actively and passively detect a client honeypot
  - Detect presence of VM (active)
  - Detect automated clicks (active)
  - Detection using time restraints (active)
  - Use forms and CAPTCHA (passive)
- But how difficult is it?




## Proof of Concept

- Built a simple HTML Server
- Created a page designed to detect automation (clicking)
- Created a script which launched only after 15 seconds
- Logged information on server log page




## Future Plans: Preventing Evasion

- Bare-Metal Implementation of client honeypots
  - Traditional honeypots use virtual environments to contain attacks experienced
  - Malicious entities can detect presence of virtual environment (See proof of concept above)
- Requires computer we can reimage
- Requires remote logging of events
  - Data might get corrupted from attack
  - Attack might destroy its own trail

- Implementing a model of user interaction
  - Vary amount of time spend on a single page (time attacks not prevented but harder)
  - Parse for hidden links and avoid clicking them
  - Be able to fill out forms and defeat CAPTCHA

## References

MITRE Data Set, Kathy Wang

Detecting the Presence of Virtual Machines Using the Local Data Table, Danny Quist, Val Smith

Ask for others and more details!

PURDUE UNIVERSITY

CERIAS

Discovery Park e-Enterprise Center