WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attacks

Roberto Perdisci, Manos Antonakakis, Xiapu Luo, and Wenke Lee

Georgia Institute of Technology
Damballa, Inc.

DSN-DCCS 2009
# Introduction

- DNS Cache Poisoning
- Pros and Cons of Existing Solutions
- Our contribution: WSEC DNS

## WSEC DNS

- WSEC Query Process
- WSEC Caching System
- Analysis of Robustness

## Experiments

## Conclusion & Future Work
Kaminsky’s DNS Cache Poisoning Attack

- Attacker wants to poison the cache entry for www.example.com
Kaminsky’s DNS Cache Poisoning Attack

- Attacker wants to poison the cache entry for www.example.com
- Attacker can control/force a client to lookup 0001.example.com
Kaminsky’s DNS Cache Poisoning Attack

- Attacker wants to poison the cache entry for www.example.com
- Attacker can control/force a client to lookup 0001.example.com
- Respond with forged packets pretending to be the Auth. NS
Kaminsky’s DNS Cache Poisoning Attack

- Attacker wants to poison the cache entry for www.example.com
- Attacker can control/force a client to lookup 0001.example.com
- Respond with forged packets pretending to be the Auth. NS
- Needs to guess the TXID to succeed
Kaminsky’s DNS Cache Poisoning Attack

- Attacker wants to poison the cache entry for www.example.com
- Attacker can control/force a client to lookup 0001.example.com
- Respond with forged packets pretending to be the Auth. NS
- Needs to guess the TXID to succeed
- If attack does not succeed, the attacker can immediately retry
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)

  - Adds authentication and integrity by signing responses
  - Based on a chain of trust anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- DNSCurve
  - Adds authentication, integrity and some confidentiality
  - Elliptic cryptography used on the communication channel
  - Apparently, no implementation available
  - No RFC/Internet Draft available, no testing
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
Pros and Cons of Existing Solutions

DNS Security Extensions (DNSSEC)
- Adds authentication and integrity by signing responses
- Based on a *chain of trust* anchored at the root NS
- Currently not widely deployed: few isolated “islands of trust”
- Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
- “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)
Pros and Cons of Existing Solutions

- **DNS Security Extensions (DNSSEC)**
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- **DNSCurve**
Pros and Cons of Existing Solutions

- **DNS Security Extensions (DNSSEC)**
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- **DNSCurve**
  - Adds authentication, integrity and some confidentiality
Pros and Cons of Existing Solutions

- **DNS Security Extensions (DNSSEC)**
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- **DNSCurve**
  - Adds authentication, integrity and some confidentiality
  - Elliptic cryptography used on the communication channel
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- DNSCurve
  - Adds authentication, integrity and some confidentiality
  - Elliptic cryptography used on the communication channel
  - Apparently, no implementation available
Pros and Cons of Existing Solutions

- DNS Security Extensions (DNSSEC)
  - Adds authentication and integrity by signing responses
  - Based on a *chain of trust* anchored at the root NS
  - Currently not widely deployed: few isolated “islands of trust”
  - Privacy-related issues with NSEC RR (solved by NSEC3), and difficult key management
  - “Political” problems include establishing the ownership/control of root NS (would it be better if ICANN was under the influence of the UN instead of the US?)

- DNSCurve
  - Adds authentication, integrity and some confidentiality
  - Elliptic cryptography used on the communication channel
  - Apparently, no implementation available
  - No RFC/Internet Draft available, no testing
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**

  - The UDP source port is chosen at random.
  - Increases search space to make brute-force attacks harder.
  - Kaminsky's attack still works, although it takes more time (still feasible).
  - Many NAT/PAT devices may not preserve randomness, making it ineffective.

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., *Www.eXaMplE.CoM*.
  - Increases query entropy, and therefore the search space.
  - Level of protection is a function of the length of name.
  - Does not work well for domains such as *hp.com*, *msn.com*, *cnn.com*, etc.

R. Perdisci, M. Antonakakis, X. Luo, W. Lee
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder

R. Perdisci, M. Antonakakis, X. Luo, W. Lee
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., `www.example.com`
  - Increases query entropy, and therefore the search space
  - Level of protection is a function of the length of name
  - Does not work well for domains such as `hp.com`, `msn.com`, `cnn.com`, etc.
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., `Www.eXaMplE.CoM`
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., `Www.eXaMplE.CoM`
  - Increases query entropy, and therefore the search space
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., *Www.eXaMplE.CoM*
  - Increases query entropy, and therefore the search space
  - Level of protection is a function of the length of name
Pros and Cons of Existing Solutions

- **UDP Source Port Randomization**
  - The UDP source port is chosen at random
  - Increases search space to make brute-force attacks harder
  - Kaminsky’s attack still works, although it takes more time (still feasible)
  - Many NAT/PAT devices may not preserve randomness, making it ineffective

- **0x20-bit encoding**
  - Uses random combinations of upper and lower case characters: e.g., `Www.eXaMplE.CoM`
  - Increases query entropy, and therefore the search space
  - Level of protection is a function of the length of name
  - Does not work well for domains such as `hp.com`, `msn.com`, `cnn.com`, etc.
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
Overview of Wildcard SECur(e) (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., "x094mf.www.example.com"
- The attacker needs to guess random prefix: very hard!
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
- The attacker needs to guess random prefix: very hard!
- Protects RDNS from poisoning attacks, including Kaminsky’s
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
- The attacker needs to guess random prefix: very hard!
- Protects RDNS from poisoning attacks, including Kaminsky’s
  - No change to the protocol/format of DNS queries
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
- The attacker needs to guess random prefix: very hard!
- Protects RDNS from poisoning attacks, including Kaminsky’s
  - No change to the protocol/format of DNS queries
  - Requires RDNS software changes and configuration changes for Auth. NS, but no change at root and TLD name servers
Overview of Wildcard SEcure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
- The attacker needs to guess random prefix: very hard!
- Protects RDNS from poisoning attacks, including Kaminsky’s
  - No change to the protocol/format of DNS queries
  - Requires RDNS software changes and configuration changes for Auth. NS, but no change at root and TLD name servers
  - Complete backward compatibility, incremental deployment
Overview of Wildcard SECure (WSEC) DNS

- Based on existing features of DNS protocol: wildcard domains (e.g., *.example.com), TXT RR
- RDNS resolvers need to follow a new query process
- RDNS adds a random prefix string (nonce) in front of queried domain names, e.g., “x094mf.www.example.com”
- The attacker needs to guess random prefix: very hard!
- Protects RDNS from poisoning attacks, including Kaminsky’s
  - No change to the protocol/format of DNS queries
  - Requires RDNS software changes and configuration changes for Auth. NS, but no change at root and TLD name servers
  - Complete backward compatibility, incremental deployment
  - Transparent to users
WSEC DNS Requirements

R. Perdisci, M. Antonakakis, X. Luo, W. Lee
Configuration Requirements for Authoritative NS

- Add two TXT records: “*” and “_test_.wsecdns_” with value “|wsecdns=enabled|”

- Add two CNAMES for each domain, e.g. for “www” add “*.wsecdns_.www” and “*.test_.wsecdns_.www”
WSEC Query Process (Scenario 1: enabled)

Note: <rand> = random string, e.g.: g9ri5._wsecdns_.www.example.com
WSEC Query Process (Scenario 2: not enabled)

1 - www.example.com A?

2 - <rand>._test._wsecdns.www.example.com TXT?

3 - NXDOMAIN

4 - www.example.com A?

5 - 10.0.1.6

6 - 10.0.1.6

- Scenario 2: WSEC queries are not enabled
- the TXT query *does not* return ‘|wsecdns=enabled|’
- RDNS issues the original user’s query with no change and still returns the correct results
- This provides **backward compatibility**
Positive and Negative WSEC Caching

- If used this way we would double the traffic
Positive and Negative WSEC Caching

- If used this way we would double the traffic
- To reduce traffic overhead we use a WSEC Cache
Positive and Negative WSEC Caching

- If used this way we would double the traffic

- To reduce traffic overhead we use a WSEC Cache

- Once the RDNS establishes that a zone *is* or *is not* WSEC enabled, no other *test query (or handshake)* will be performed for a given time $T$ (e.g., $T=1$ day).
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.
- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.
- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.
- WSEC DNS avoids changes to root and TLDs:
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.
- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.
- WSEC DNS avoids changes to root and TLDs:
  - root and TLD NS are delegation-only name servers.
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.
- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.
- WSEC DNS avoids changes to root and TLDs:
  - root and TLD NS are delegation-only name servers
  - their IP addresses change very rarely
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.

- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.

- WSEC DNS avoids changes to root and TLDs:
  - root and TLD NS are **delegation-only** name servers.
  - their IP addresses change very rarely.
  - WSEC uses a **secure cache update** policy for TLDs.
Protecting the TLDs

- If root and TLD NS were WSEC enabled (config changes), no further action would be needed.
- However, changing root and TLDs is one of the main reasons why DNSSEC is not widely deployed.
- WSEC DNS avoids changes to root and TLDs:
  - root and TLD NS are delegation-only name servers.
  - their IP addresses change very rarely.
  - WSEC uses a secure cache update policy for TLDs.
  - if a change of IP is detected (by comparing with cache info), RDNS asks a root NS for confirmation, before overwriting the cache.
## Resisting Poisoning Attacks

Resistance to Kaminsky’s Attack for different Defense Scenarios

<table>
<thead>
<tr>
<th>Combination of Existing Solutions</th>
<th>BW = 1Mbps</th>
<th>BW = 10Mbps</th>
<th>BW = 100Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXID+src port(16)</td>
<td>21.74 days</td>
<td>2.17 days</td>
<td>0.22 days</td>
</tr>
<tr>
<td>TXID+src port(16)+0x20(6)</td>
<td>3.81 years</td>
<td>0.38 years</td>
<td>0.038 years</td>
</tr>
<tr>
<td>TXID+src port(16)+ANS(2)+0x20(6)</td>
<td>15.25 years</td>
<td>1.52 years</td>
<td>0.15 years</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effect of NAT/PAT (Port Derandomization)</th>
<th>BW = 1Mbps</th>
<th>BW = 10Mbps</th>
<th>BW = 100Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXID</td>
<td>29.08 seconds</td>
<td>2.87 seconds</td>
<td>0.25 seconds</td>
</tr>
<tr>
<td>TXID+0x20(6)</td>
<td>31.06 minutes</td>
<td>3.1 minutes</td>
<td>0.31 minutes</td>
</tr>
<tr>
<td>TXID+ANS(2)+0x20(6)</td>
<td>2.07 hours</td>
<td>0.21 hours</td>
<td>0.02 hours</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using WSEC (prefix length=5)</th>
<th>BW = 1Mbps</th>
<th>BW = 10Mbps</th>
<th>BW = 100Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXID+WSEC(\log_{2}36^5)</td>
<td>55.83 years</td>
<td>5.58 years</td>
<td>0.56 years</td>
</tr>
</tbody>
</table>

**Table:** Time needed to reach probability of attack success, \( P_{\text{succ}} = 0.5 \).

- BW = Attacker’s Bandwidth
- RDNS to ANS RTT = 100 ms
- Numbers between parenthesis represent additional bits of entropy

R. Perdisci, M. Antonakakis, X. Luo, W. Lee
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
- We compared the following software configurations
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
- We compared the following software configurations
  - Original PowerDNS implementation (pdns)
  - PowerDNS with 0x20-bit encoding (pdns+0x20)
  - 0x20 when length (d) ≥ 16, or WSEC when length (d) < 16 (pdns+WSEC+0x20(16))
  - WSEC DNS only (pdns+WSEC)
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
- We compared the following software configurations
  - Original PowerDNS implementation (pdns)
  - PowerDNS with 0x20-bit encoding (pdns+0x20)
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
- We compared the following software configurations
  - Original PowerDNS implementation (pdns)
  - PowerDNS with 0x20-bit encoding (pdns+0x20)
  - 0x20 when $length(d) \geq 16$, or WSEC when $length(d) < 16$ (pdns+WSEC+0x20(16))
Experimental Setup

- POC implementation of 0x20 and WSEC on top of PowerDNS v3.1.7
- About 250 lines of C/C++ code (available online)
- Dataset of 100,000 DNS queries extracted from real DNS traffic towards the RDNS of a large enterprise network
- We compared the following software configurations
  - Original PowerDNS implementation (pdns)
  - PowerDNS with 0x20-bit encoding (pdns+0x20)
  - 0x20 when $\text{length}(d) \geq 16$, or WSEC when $\text{length}(d) < 16$ (pdns+WSEC+0x20(16))
  - WSEC DNS only (pdns+WSEC)
## Experimental Results

**Figure:** Cumulative Distribution Function of the RTT.

![Cumulative Distribution Function](image)

<table>
<thead>
<tr>
<th>DNS Server</th>
<th># queries to ANSs</th>
<th>Median RTT</th>
<th>DNS traffic</th>
<th>Max Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pdns</code></td>
<td>220,930</td>
<td>52 ms</td>
<td>28.67 MB</td>
<td>3.25 MB</td>
</tr>
<tr>
<td><code>pdns+0x20</code></td>
<td>229,157</td>
<td>73 ms</td>
<td>30.04 MB</td>
<td>3.25 MB</td>
</tr>
<tr>
<td><code>pdns+WSEC+0x20(16)</code></td>
<td>255,605</td>
<td>87 ms</td>
<td>37.22 MB</td>
<td>4.79 MB</td>
</tr>
<tr>
<td><code>pdns+WSEC</code></td>
<td>269,156</td>
<td>90 ms</td>
<td>41.11 MB</td>
<td>5.57 MB</td>
</tr>
</tbody>
</table>

**Table:** Comparison between PowerDNS, 0x20, and WSEC DNS.
Conclusion & Future Work

- We presented WSEC DNS, a new query algorithm that protects RDNS resolvers from poisoning.
Conclusion & Future Work

- We presented WSEC DNS, a new query algorithm that protects RDNS resolvers from poisoning.
- We compared it to previous work and discussed advantages and disadvantages.

WSEC DNS
Conclusion & Future Work

- We presented WSEC DNS, a new query algorithm that protects RDNS resolvers from poisoning.
- We compared it to previous work and discussed advantages and disadvantages.
- WSEC DNS is backward compatible and can be incrementally deployed.

R. Perdisci, M. Antonakakis, X. Luo, W. Lee
Conclusion & Future Work

- We presented WSEC DNS, a new query algorithm that protects RDNS resolvers from poisoning.
- We compared it to previous work and discussed advantages and disadvantages.
- WSEC DNS is backward compatible and can be incrementally deployed.
- WSEC is immediately deployable, because it does not need changes at the root and TLD name servers, but only configuration changes at the ANS level and RDNS software update.
Conclusion & Future Work

- We presented WSEC DNS, a new query algorithm that protects RDNS resolvers from poisoning.
- We compared it to previous work and discussed advantages and disadvantages.
- WSEC DNS is backward compatible and can be incrementally deployed.
- WSEC is immediately deployable, because it does not need changes at the root and TLD name servers, but only configuration changes at the ANS level and RDNS software update.
- Experiments show a “natural trade-off” between network/system overhead and increasing security while maintaining complete backward compatibility.
Conclusion & Future Work

- Optimize software to avoid unnecessary queries
Conclusion & Future Work

- Optimize software to avoid unnecessary queries
- Better study of the effect of DNS query “locality” on the RTT, RDNS cache size, and traffic increase
Conclusion & Future Work

- Optimize software to avoid unnecessary queries
- Better study of the effect of DNS query “locality” on the RTT, RDNS cache size, and traffic increase
- Measure effect of large-scale deployment (e.g., using PlanetLab)
Conclusion & Future Work

- Optimize software to avoid unnecessary queries
- Better study of the effect of DNS query “locality” on the RTT, RDNS cache size, and traffic increase
- Measure effect of large-scale deployment (e.g., using PlanetLab)
- Compare overhead introduced by WSEC DNS with overhead of DNSSEC
Thank You!

questions?
perdisci@gtisc.gatech.edu