Cross-Site Search (XS-Search) Attacks

Hemi Leibowitz, OWASP AppSec IL 2015
Work By:

_Nethanel Gelernter:_
Head of the cyber research group at the Michlala LeMinhal.

_Professor Amir Herzberg:_
Head of the Secure Communication and Computing (`Cyber`) group at Bar-Ilan University
AGENDA

Extraction of private, sensitive data using cross-site vulnerabilities via XS-Search attacks

➢ Who, what, how?

➢ Demo

➢ Conclusions

* All experiments were performed ethically
VULNERABLE SITES AND DATA

Mail content, contacts...

Search history

Structured information

Relationships

And a lot more...
EXAMPLE SCENARIO

GET / POST request to Gmail
Browser receives the response and displays it
Cross-Site Attacks
XS-SEARCH: HIGH LEVEL VIEW

GET/POST request

Response

...<script>...

Gmail by Google
XS-SEARCH: HIGH LEVEL VIEW

...<script>...

Allowed

GET/POST request

Response
Timing Side Channel

We can’t read the response, BUT - we can measure how long it took
XS-SEARCH: HIGH LEVEL VIEW

...<script>...

Time {

GET/POST request

Response

}
PROBLEMS

1. **Noise** -
   a. Timing a response is inaccurate and influenced by many factors (Internet connection, Browser etc.)
   b. Very (very) short time differences between responses (even long ones) - especially when heavily compressed.

2. **Small window of opportunity** -
   a. User visits the page for a short term only
   b. Avoid detection mechanisms (anti-DoS)
These XS-Search attacks are impractical
**Dummy** - request that yields a short (fast) response

\[
q=\text{in}:\text{sent}\&\text{from}:\text{fdjakdhasd}
\]

**Challenge** - request that yields either long or short response

\[
q=\text{in}:\text{sent}\&\text{from}:\text{Alice}
\]
BASIC FLOW: ANSWER BOOLEAN QUESTIONS

\[ T(\text{Dummy}) \approx T(\text{Challenge}) \Rightarrow \text{False} \]

\[ T(\text{Dummy}) \ll T(\text{Challenge}) \Rightarrow \text{True} \]
XS-SEARCH: BASIC FLOW

![Bar chart comparing Dummy and Challenge times to load response](chart.png)
XS-SEARCH: BASIC FLOW

...<script>...

T(Dummy)  

GET \texttt{q=in:sent\&from:fdjakdhasd}  

Empty response

T(Challenge)  

GET \texttt{q=in:sent\&from:Alice}  

\texttt{? Unknown response}
DEALING WITH THE PROBLEMS

➢ Dummy / Challenge pairs
➢ Statistical tests
➢ Inflation techniques
➢ Divide and Conquer algorithms
STATISTICAL TESTS

Classical statistical hypothesis tests assume large samples.

In order to achieve good results using small samples:

➢ Ran each Dummy / Challenge pair a few times
➢ Tested and compared various statistical tests between the distributions

Main observation: lower values give better indication
INFLATION TECHNIQUES

Increase the difference of the response time between empty and full response

➢ Response-length inflation
  ○ Query fields are copied to the response

➢ Compute-time inflation
RESPONSE-LENGTH INFLATION
Abuses hard-to-compute ‘has not’ search terms

Short circuit ‘empty’ queries

Allows detection of information that appears only once!
COMPUTE-TIME INFLATION

- Abuses hard-to-compute ‘has not’ search terms
- Short circuit ‘empty’ queries
- Allows detection of information that appears only once!

**Dummy:**

\[ q=in: sent \& from: fdjakdhasd \& hasnot: \{ rjew +...+ iqejh \} \]

**Challenge:**

\[ q=in: sent \& from: Alice \& hasnot: \{ rjew +...+ iqejh \} \]
EFFICIENT TERM IDENTIFICATION

Which of \{T_1, T_2, \ldots \} appears in <data>? 
Naïve solution: check one by one...

Three efficient divide and conquer algorithms:

- Multiple Terms Identification (MTI)
- Optimized Multiple Terms Identification (OMTI)
- Any Term Identification (ATI)

Each of them sends queries for conjunction of terms from:michael+OR+dan+OR+\ldots Up to the URL limit
WHAT CAN WE EXPOSE WITH XS-SEARCH?

➢ Specific terms or from list of candidate terms
➢ By date, subject, folder, or other properties
➢ Structured information
  ○ Credit card numbers (xxxx-xxxx-xxxx-xxxx)
  ○ Phone numbers (xxx-xxxxx-xxx)
WHAT CAN WE EXPOSE WITH XS-SEARCH?

➢ Does the name of the user is Alice?
  ○ in:sent&from:alice

➢ Closely related to bob@gmail.com?
  ○ bob@gmail.com&st=100

➢ Is a client of SomeBank?
  ○ noreply@somebank.com

➢ Do have Bob as a friend in Google+?
  ○ from:bob&circle:friends

➢ Did Bob bcc Charlie about an amazing lecture?!
 WHAT CAN WE EXPOSE WITH XS-SEARCH?

Credit card numbers (xxxx-xxxx-xxxx-xxxx)

➢  \[x \in \{0,1\ldots 9\} \Rightarrow 10^{16} =\]

\[10,000,000,000,000,000,000\]

But, using XS-Search we only need to reveal xxxx

➢  Only \(10^4\) (= 10,000) possibilities!
PREVENTING XS-SEARCH?

We cannot complete the action at this time. Please try again using the search action above.
PREVENTING XS-SEARCH?

*Easy* - prevent any cross-site request.

BUT...

Many services wish to allow cross-site requests.

These services can *try* to:

- Restrict: limit requests rate, inflation ...
- Detect: anomalies, heuristics...
Thanks!

Any questions?

You can find me at:

leibo.hemi@gmail.com
Special thanks to all the people who made and released these awesome resources for free:

▷ Presentation template by SlidesCarnival
▷ Photographs by Unsplash